THE EXPERT'S VOICE® IN OFFICE

Pro Access 2010 Development

HARNESS THE POWER OF THE WORLD'S MOST PREVALENT DESKTOP DATABASE IN TODAY'S WEB-CONNECTED WORLD

Mark Collins

Apress®

www.it-ebooks.info

For your convenience Apress has placed some of the front matter material after the index. Please use the Bookmarks and Contents at a Glance links to access them.



Apress®

www.it-ebooks.info

Contents at a Glance

- About the Author xix
- About the Technical Reviewer xx
- Acknowledgments xxi
- Preface xxii

PART 1 ■ ■ ■ Introduction 1

- Chapter 1: Introduction 3

PART 2 ■ ■ ■ Defining the Database Schema 15

- Chapter 2: Defining and Relating Tables 17
- Chapter 3: Using Data Macros 53
- Chapter 4: Designing Queries 77
- Chapter 5: Creating PivotTables 103

PART 3 ■ ■ ■ Creating Forms and Reports 125

- Chapter 6: Standard Forms 127
- Chapter 7: Creating a CheckOut Form 153
- Chapter 8: Creating a Customer Admin Form 191
- Chapter 9: Enhancing Product Administration 221
- Chapter 10: Enhancing the User Experience 255
- Chapter 11: Branding with Themes and Styles 291
- Chapter 12: Reports 309

www.it-ebooks.info

PART 4 ■ ■ ■ Multiuser Considerations 359

■ **Chapter 13: Upsizing 361**

■ **Chapter 14: Distributing the Application 393**

■ **Chapter 15: Publishing to the Web 415**

PART 5 ■ ■ ■ Advanced Topics 455

■ **Chapter 16: Integrating Outlook 457**

■ **Chapter 17: Using External Data 489**

■ **Chapter 18: Miscellaneous Features 511**

■ **Chapter 19: Security 531**

■ **Appendix A: Northwind Web Database 551**

■ **Index 563**

www.it-ebooks.info

PART I

Introduction

www.it-ebooks.info

CHAPTER 1

Introduction

Microsoft Access is a unique platform that enables rapid development of database applications. It has been a long-time favorite of both developers and end users for creating single-user and small-scale office automation solutions. Recent improvements have made it a viable choice for enterprise-class applications as well. In this book, I'll show you how to take advantage of all the great features that Access 2010 has to offer. I think you'll find it surprisingly easy to build full-featured applications.

What Is Access?

For those that may be new to Access, a brief overview will be helpful before I get into the details. At its core, Access is a relational database engine that allows you to define tables to store data. You can define primary keys, indices, and table relationships much like you would with SQL Server or Oracle. You can also write queries to insert, update, retrieve, or delete data from your tables.

Defining the User Interface

However, this is where the similarities with other database engines end. Access is also an application development platform that you can use to define forms, menus, and reports.

With Access, you can design forms to display and update data. Because the form definition is integrally tied to the data schema, the form can easily enforce data integrity. For example, if you define a foreign key relationship between two tables, the form can automatically provide a dropdown list showing the permitted values.

Access provides a facility for designing reports that use tables or queries. Again, because the report writer is integrated into the database design, you can simplify the report definition. Access also allows you to create forms for navigating through the various forms and reports, so you can control the entire user experience. Finally, by using macros and Visual Basic for Applications (VBA), you can automate tasks and provide advanced features.

Choosing the Application Architecture

Because Access is both a database engine and an application development platform, there are several ways that these components can be configured. In this book, I will show you how to implement these configurations. The simplest scenario is to include both tables and forms in a single "database." The end user will open this file using the Access application. This is ideal for single user applications.

To support multiple users, you can split the UI elements (forms, reports, and macros) into a separate file. The tables are then referenced as *linked* tables in their own database. Each user can open their own copy of the UI file but reference a common database file.

www.it-ebooks.info

With Access, you can reference data from various sources such as Excel files, other Access databases, text and XML files, SharePoint lists, and Open Database Connectivity (ODBC) sources. It is possible to use Access solely as an application development platform using data from another source such as SQL Server.

With Access 2010, you can also publish your database to a SharePoint server. This gives end users access to your data and forms without needing a copy of the Access runtime. This also takes advantage of all the enterprise features of SharePoint such as security, concurrency and offline data caching.

Using This Book

This book is designed to show you how to use Access 2010 to build a complete application. You will create an Access database in Chapter 2 and each subsequent chapter will build upon the previous chapters adding features to your database. I recommend that you work through each chapter in order and, when you're done, you will have full-featured Access database application. You can download the final result from `www.apress.com` if you prefer. I will also create a copy of the interim application at the end of each chapter. So if you wanted to skip to Chapter 8, for example, you can download the database from Chapter 7 and start from there.

The Sample Application

The sample project that you will create throughout this book will support a lending library. It keeps track of all the inventory items including which items have been checked out and when they are due back. It also computes any overdue fees. It will send reminder e-mails when items are due and allow users to request an item to be held for them.

To give you a preview of where we're heading in this book, I'll briefly describe the end product. This is what you will create yourself as you work through the exercises in each chapter. When you first load the Access database, the initial view will look like Figure 1-1.

www.it-ebooks.info

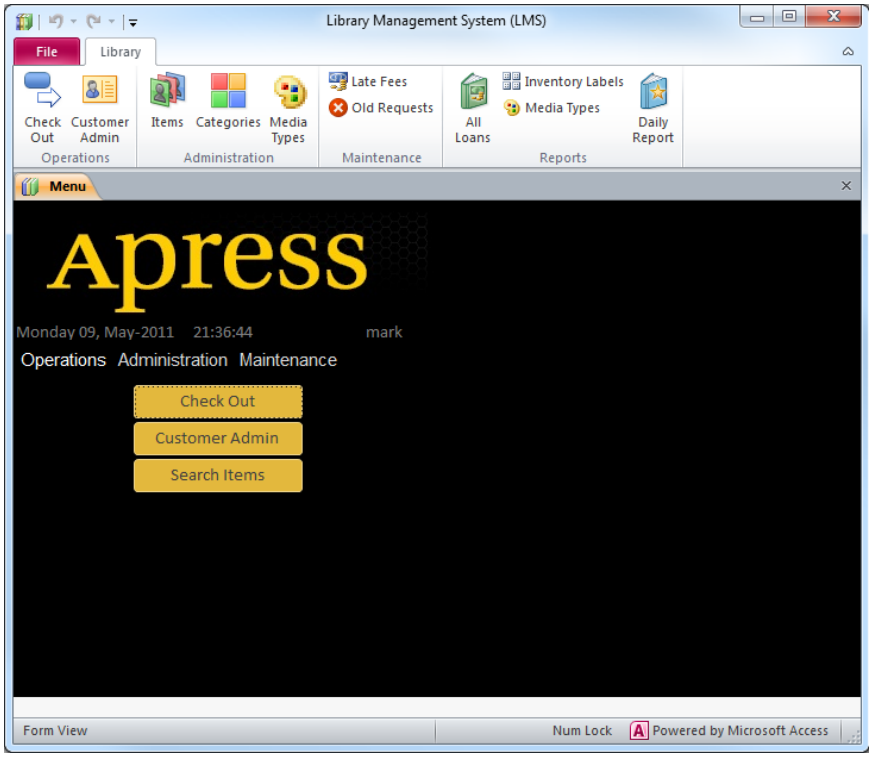


Figure 1-1. *The main* `Navigation` *form*

This is the `Navigation` form that is loaded automatically when the application starts. Notice that all of the standard Access menus and navigation are removed. In their place, a custom ribbon is used to reveal all of the forms, reports, and macros that are available to the users. Most of these features are also available through the tabbed dialog form. The application has been *branded* to mimic the `www.apress.com` website.

Figure 1-2 shows the first tab of the `Customer Admin` form, which is used to add or update a customer's profile.

www.it-ebooks.info

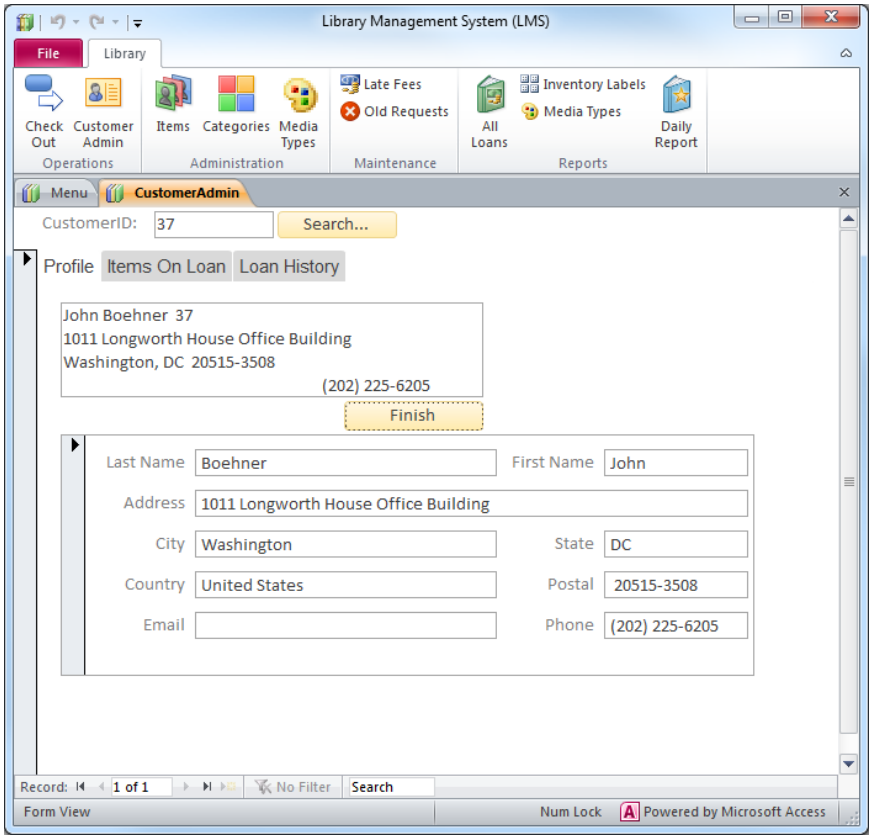


Figure 1-2. *The Profile tab of the* `Customer Admin` *form*

The next two tabs of this form show the items currently checked out and a complete history of all items that have been loaned to this customer. The "Items On Loan" tab is shown in Figure 1-3.

www.it-ebooks.info

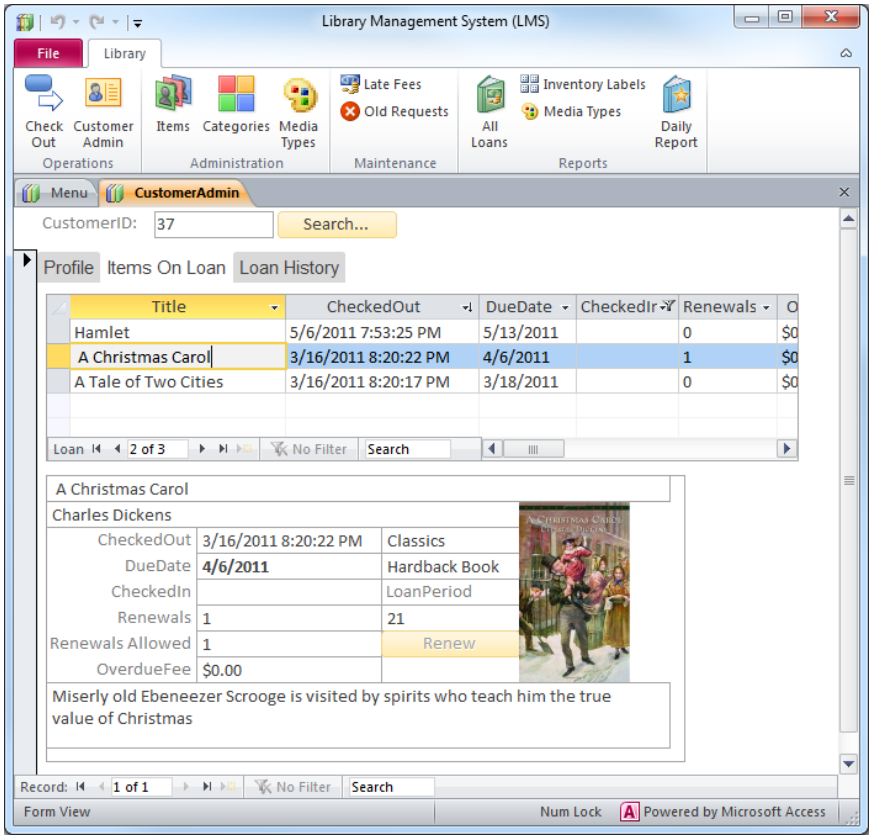


Figure 1-3. *The Items On Loan tab*

The top grid lists the items still checked out. The lower form shows the details of the selected item. If the item is overdue, the due date will blink to highlight that fact. Notice that an image of the item is included on this form.

The application provides a search feature for finding items based on author, title, or description. The `Item Search` form is shown in Figure 1-4.

www.it-ebooks.info

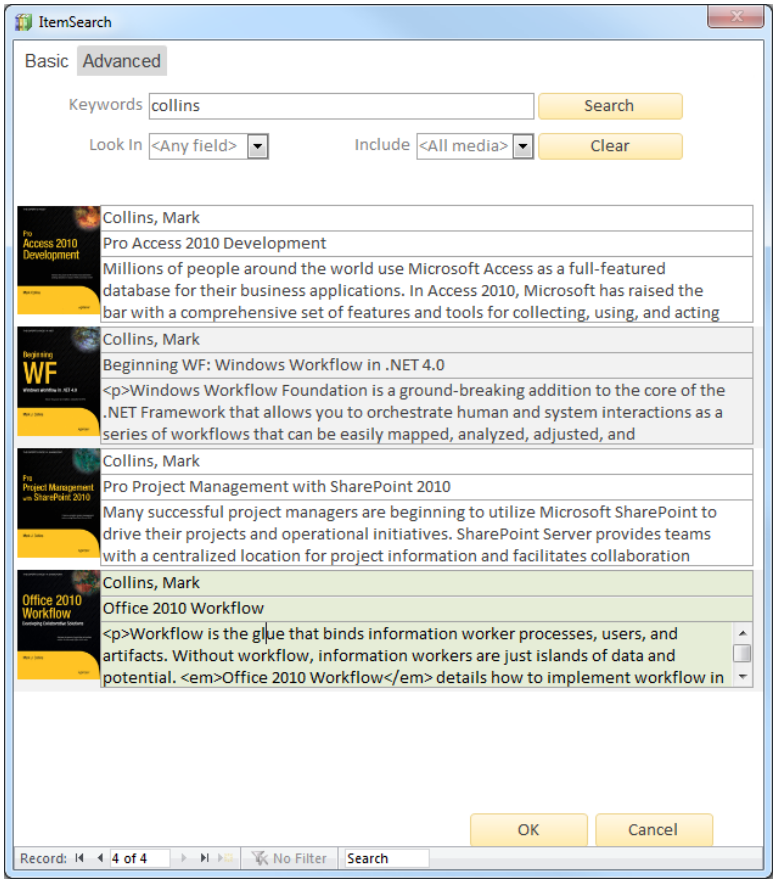


Figure 1-4. *The* `Item Search` *form*

The `Item Search` form uses a Continuous Form to allow more flexibility is formatting the records while still allowing multiple records to be displayed simultaneously. It also uses conditional formatting to highlight the current record. Once the item has been selected, it is displayed in the `Item` form, shown in Figure 1-5.

www.it-ebooks.info

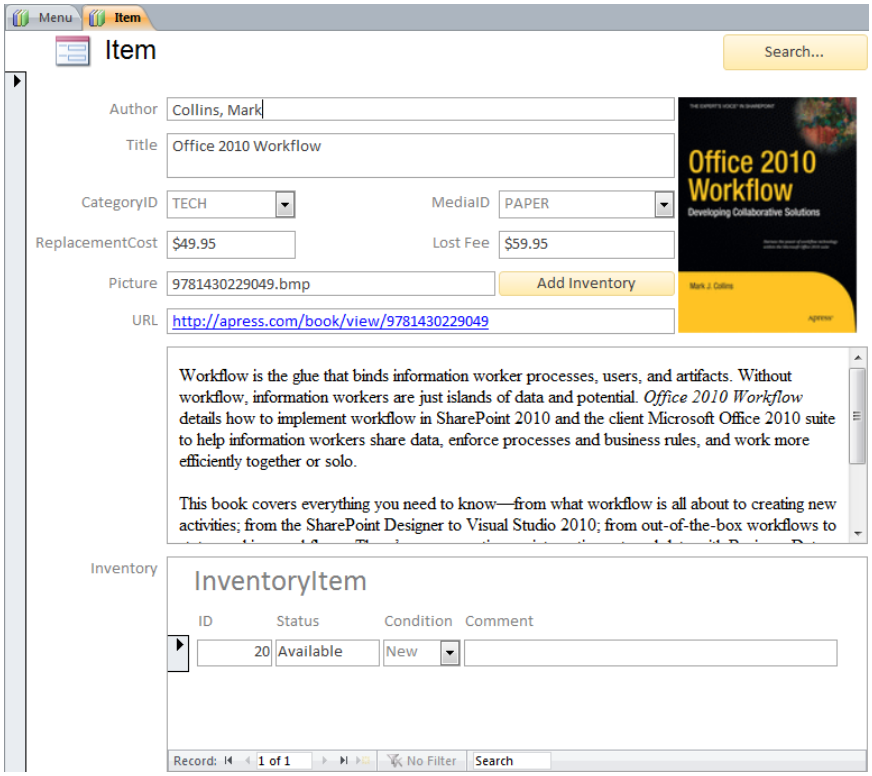


Figure 1-5. *The* `Item` *form*

The `Item` form is used to view and update the basic item information such as author, title, and description. The description is displayed in a Web Browser Control so the HTML tags will be displayed properly. It includes a subform that shows the status of the inventory (copies) of that item. You can also use this form to add additional copies and to update the status of existing items.

The primary use of this application is to allow items to be loaned (or checked out) to a customer. You can do this through the `CheckOut` form shown in Figure 1-6.

www.it-ebooks.info

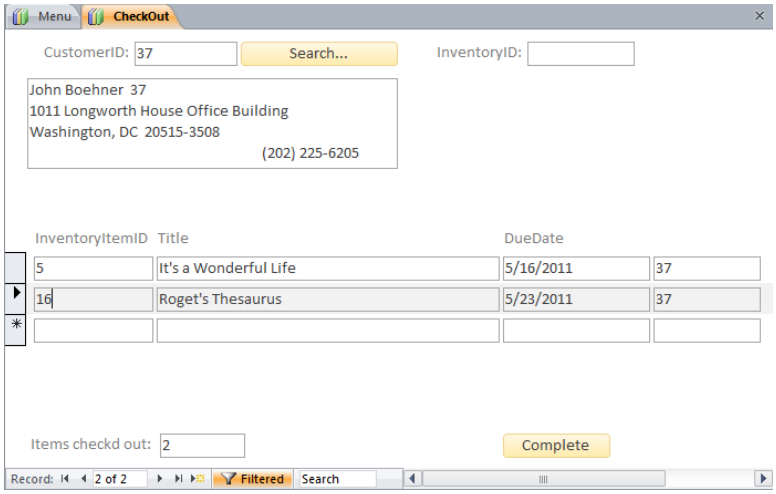


Figure 1-6. *The* `CheckOut` *form*

In the `CheckOut` form, you first select a customer by entering their ID or you can search for them using a custom search form. Then you enter (or scan) the inventory item IDs and the item is added to the form. The due date is automatically calculated based on the business rules defined by the selected item.

When the checkout process is complete, a receipt is automatically generated listing the items that are being loaned. A sample is shown in Figure 1-7.



Figure 1-7. *A sample* `CheckOut` *receipt*

You will create several reports that will demonstrate the various ways that reports can be designed. One of these, the `AllLoans` report is shown in Figure 1-8.

www.it-ebooks.info

AllLoans

Monday, May 09, 2011
9:56 PM

CheckedOut	DueDate	OverdueFee	Title	Author
CD Audio				
Seasonal				
4/9/2011 3:40:15 PM	4/16/2011	$11.50	White Christmas	Bing Crosby
4/9/2011 3:41:26 PM	4/16/2011	$11.50	White Christmas	Bing Crosby
Summary for 'CategoryDescription' = Seasonal (2 detail records)				
Sum		$23.00	7.17%	
Summary for 'MediaDescription' = CD Audio (2 detail records)				
Sum 2	6.25%	$23.00	7.17%	
DVD Video				
Classics				
1/8/2011 10:01:05 PM	1/15/2011	$60.00	Hamlet	William Shakespear
1/8/2011 10:02:15 PM	1/15/2011	$60.00	Hamlet	William Shakespear
5/6/2011 7:53:25 PM	5/13/2011	$0.00	Hamlet	William Shakespear
Summary for 'CategoryDescription' = Classics (3 detail records)				
Sum		$120.00	37.38%	
Seasonal				
1/8/2011 9:51:54 PM	1/15/2011	$60.00	It's a Wonderful Life	Frank Capra
3/14/2011 12:38:19 PM	3/21/2011	$0.00	It's a Wonderful Life	Frank Capra
3/14/2011 12:41:49 PM	3/21/2011	$0.00	It's a Wonderful Life	Frank Capra
3/14/2011 12:58:24 PM	3/21/2011	$0.00	It's a Wonderful Life	Frank Capra
4/9/2011 3:40:10 PM	4/16/2011	$23.00	It's a Wonderful Life	Frank Capra
4/9/2011 3:40:20 PM	4/16/2011	$23.00	It's a Wonderful Life	Frank Capra
4/9/2011 3:41:29 PM	4/16/2011	$0.00	It's a Wonderful Life	Frank Capra
5/9/2011 9:54:12 PM	5/16/2011	$0.00	It's a Wonderful Life	Frank Capra
Summary for 'CategoryDescription' = Seasonal (8 detail records)				
Sum		$106.00	33.02%	
Summary for 'MediaDescription' = DVD Video (11 detail records)				
Sum 11	34.38%	$226.00	70.40%	

Figure 1-8. *A sample* `AllLoans` *report*

The `Library` database will be published to a SharePoint site and many of its features will be made available using only a web browser. The `Customer` list, for example, is shown in Figure 1-9.

Library ▸ Options ▾ INTERNAL\mark ▾

Library Management System (LMS)

CustomerID	Last Name	First Name	Address	City	State P	ZIP Postal	Coun
1	Bishop	Rob	123 Cannon House Office Building	Washington	DC	20515-4401	Unite
2	Bishop	Sanford	2429 Rayburn House Office Building	Washington	DC	20515-1002	Unite
3	Bishop	Timothy	306 Cannon House Office Building	Washington	DC	20515-3201	Unite
4	Black	Diane	1531 Longworth House Office Building	Washington	DC	20515-4206	Unite
5	Blackburn	Marsha	217 Cannon House Office Building	Washington	DC	20515-4207	Unite
6	Blumenauer	Earl	1502 Longworth House Office Building	Washington	DC	20515-3703	Unite
7	Boehner	John	1011 Longworth House Office Building	Washington	DC	20515-3508	Unite
8	Bonner	Jo	2236 Rayburn House Office Building	Washington	DC	20515-0101	Unite
9	Bono Mack	Mary	104 Cannon House Office Building	Washington	DC	20515-0545	Unite
10	Bordallo	Madeleine	2441 Rayburn House Office Building	Washington	DC	20515-5301	Unite
11	Washington	George	3200 Mount Vernon Mem Hwy	Mount Vernon	VA	22309	Unite
12	Adams	John	1600 Pennsylvania Ave.	Washington	DC	20500	Unite
13	Jefferson	Thomas	931 Thomas Jefferson Parkway	Charlottesville	VA	22902	Unite
14	Ackerman	Gary	2111 Rayburn House Office Building	Washington	DC	20515-3205	Unite
15	Adams	Sandy	216 Cannon House Office Building	Washington	DC	20515-0924	Unite
16	Aderholt	Robert	2264 Rayburn House Office Building	Washington	DC	20515-0104	Unite

Category
Media
Customer
Item

Figure 1-9. *The* `Customer` *list retrieved from a SharePoint site*

www.it-ebooks.info

The `Item` web form, shown in Figure 1-10, is another feature that is exposed via the web.

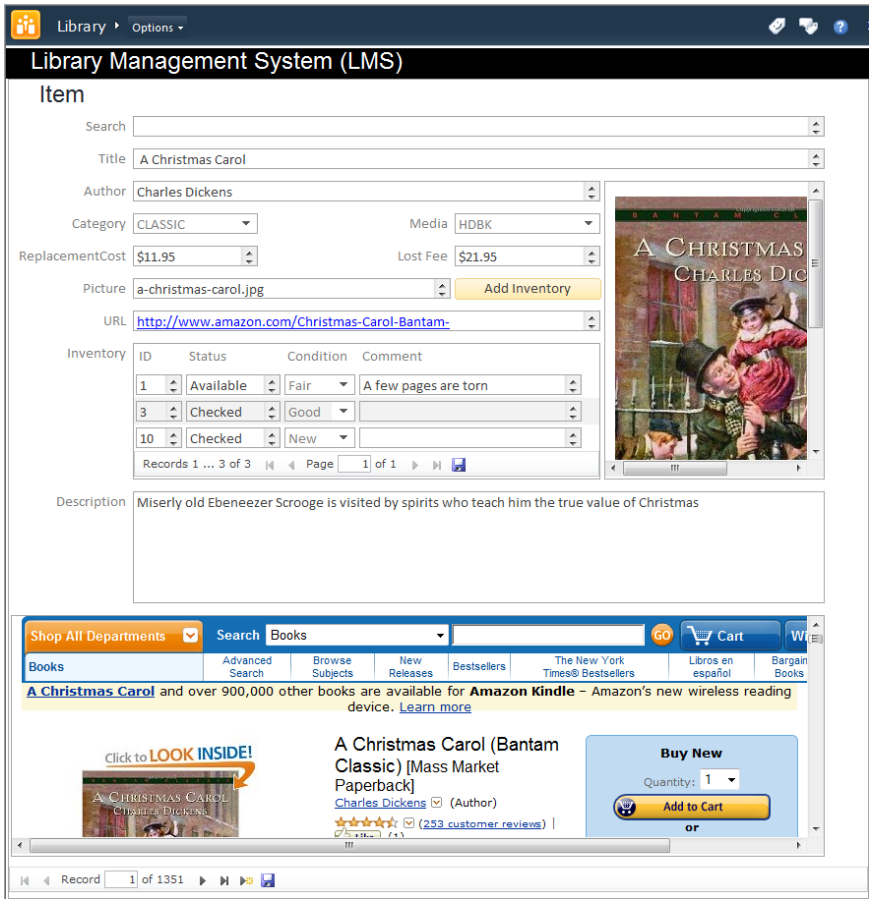


Figure 1-10. *The web-based* `Item` *form*

The `Library` database will also send out reminder e-mails when items are overdue. Figure 1-11 shows a sample e-mail that is generated using Outlook 2010.

www.it-ebooks.info

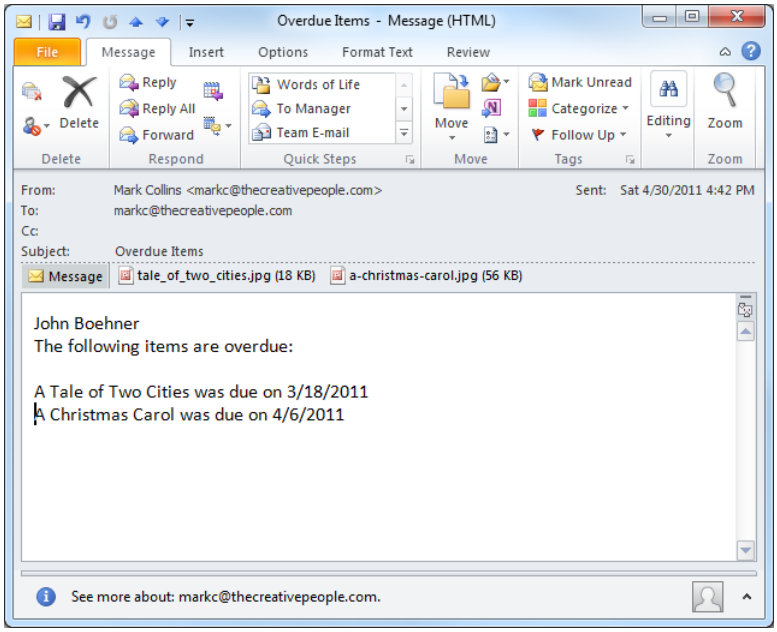


Figure 1-11. *A sample reminder e-mail*

The exercises in this book were carefully chosen to demonstrate how to take advantage of the core features of Access. I have also tried to implement some of the most common application scenarios, especially those that might seem more challenging. At the end of this book, not only will you have a working application but also lots of hands-on examples that you can reuse in your future projects.

Getting Started

The chapters in this book are organized in a logical progression that you would typically use when building an Access application. Part 1 includes chapters that show you how to define the data schema, including tables, queries, and data macros, which are similar to database triggers.

Part 2 explains how to create the UI elements such as forms and reports. Each exercise will demonstrate new features in Access and will provide new approaches for addressing typical application scenarios. You'll start by creating simple forms that require minimal implementation. Subsequent chapters will make heavy use of VBA and macros to implemented more complex forms and reports.

In Part 3 I'll demonstrate several deployments strategies that move the data to a central repository and allow the application to be distributed to each user. I'll also show you how to use the Access runtime so each user does not require an Office license. The repositories that you will use include:

- An Access backend database
- SQL Server

www.it-ebooks.info

- SQL Azure
- SharePoint

Finally, in Part 5, I'll cover a number of miscellaneous topics such as:

- How to link external data sources into your application
- How to integrate with Outlook
- Understanding the Access security features

In the next chapter, you'll create the `Library` database and define the tables that will be used for the rest of the book. This is how most good Access solutions begin, by providing a good data model at the start.

www.it-ebooks.info

PART 2

Defining the Database Schema

In Part 2, you'll create an Access database and define schema elements, such as tables and queries. Because Access makes it easy to add or modify tables, the database design is often done in ad-hoc fashion. Tables and columns are added as needed to support the form or report that is being developed. However, because the data schema forms the basis for the application, I recommend performing a thorough database design upfront.

In Chapter 2, you'll start by designing the tables. Access provides facilities for defining data integrity, such as foreign keys, unique constraints, and field validation. I encourage you to constrain the database design as much as possible. For example, if a column should not contain a negative value, make that an explicit rule in the table design.

In Chapter 3, you'll use data macros to further constrain the referential integrity. You'll also use them to propagate changes to other tables. For example, when a loan is created, the status of the associated item can be automatically updated. Data macros are also a great way to implement business rules, such as computing the late fee, if appropriate.

In Chapter 4, you'll design several queries that you will use later when implementing forms and reports. Action queries that insert, update, or delete records are a convenient way to perform batch processing.

In Chapter 5, you will implement a pivot, which is a useful tool for analyzing a large table.

www.it-ebooks.info

CHAPTER 2

Defining and Relating Tables

The logical starting point in designing a database application is to define the tables that will store the data. The table schema is the foundation that must support the entire application, so it is imperative that this be designed well. Because Access makes it so easy to build an application, it can be tempting to skip this important step, but don't.

In this chapter I'll explain some of fundamental principles that ensure a good data model. These techniques are universally applicable to most database engines. I will cover database normalization, defining primary and foreign keys and using constraints to ensure referential integrity.

I will guide you through the process of creating a base set of tables for your application. I'll show you step-by-step how to design the tables and use the many data modeling features in Access 2010. By the end of the chapter, you'll have a solid structure on which to build the rest of the application.

Considering Design Practices

Before starting the specific design for this project, I want to cover some basic principles that should be followed in any database design.

Using Primary Keys

While not required by Access, it is strongly recommended that you define a primary key for each table. A primary key uniquely defines a single record in the table. A primary key is needed when you define relationships between tables, and relating tables is a key part of defining a *relational* database.

There are two popular approaches to designing primary keys. The first is to create a single surrogate key field. Typically, this column has the same name as the table with an "ID" appended to it. For example, the primary key for an `OrderItem` table would be `OrderItemID`. The `AutoNumber` column type works well with this approach. When a record is inserted, the database engine assigns the next number to this field. This guarantees uniqueness of the key.

A second approach is to *inherit* the key from a parent table, adding an additional field for each level in the table hierarchy. For example, the primary key for the `Order` table would be `OrderID`, and for the `OrderItem` table, the primary key would be a composite of `OrderID` and `OrderItemID` columns. In this case, the `OrderItemID` only has to be unique within the specific `Order`. From a designer's perspective, this approach makes it clear that the `OrderItem` table has an aggregate relationship with the `Order` table. In practice, however, this is more difficult to implement, because it requires the application to generate the key values. Defining foreign key relationships in this situation is also more awkward, because multiple fields are required. With this design, some tables could end up having three or more primary key fields.

For the project in this book, I will use the former approach and give each table a single primary key field. I believe this to be the more prevalent approach, especially in Microsoft platforms such as Access and SQL Server. This will also simplify the implementation of this project.

www.it-ebooks.info

Normalizing a Database

A well designed database will have some level of *normalization*. Normalization is basically the process of removing redundant information. There is a series of definitions called normal forms that provide standard rules for determining normalization. These are successively more normalized. First normal form (1NF), for example, requires that there are no duplicate columns in a table and that each table has a primary key (a primary key can be comprised of multiple columns).

Second normal form (2NF) requires that each non-key field be dependent on the entire key, not just part of it. For example, in the `OrderItem` table described earlier, where the key is a combination of `OrderID` and `OrderItemID`, each non-key field must be dependent on both key fields. Fields such as `Customer` and `OrderDate` are dependent only on `OrderID` and not `OrderItemID`.

Third normal form (3NF) further requires that every column in a table be solely dependent on the primary key. For example, suppose you had a table of customer orders that included the customer's name and address. The name and address are attributes of the customer not the order. To satisfy 3NF, these columns should be removed and placed in a separate customer table. The order table would then include only a foreign key field, which is the primary key of the customer table.

Another way to look at this rule is that if a customer had more than one order, their name and address would be repeated on each order. Reducing this duplication of data is the goal of normalization. Another common example that violates 3NF is including a column that can be computed based on other columns. For example, including both `BirthDate` and `Age` would violate 3NF, since `Age` can be computed based on the `BirthDate`.

■ **Tip** For more information about database normalization, this set of slides presents a practical overview: `www.uncg.edu/ism/ism318/normalization.pdf`.

The goal of these formal definitions is to help you identify where there is redundant data. This redundancy should be eliminated for three primary reasons:

- Redundant data creates larger databases.
- Redundant data is much more difficult to maintain.
- Redundant data is more likely to have errors.

Storage is relatively cheap, so unless you're working with rather large databases, the extra space used by redundant data is probably not a big issue. The maintenance issue, however, can be significant. Consider, for example, the order table that includes the customer name and address. Suppose the customer moves; you would have to update every order record to store their new address. Also, for every new order, their name and address would need to be entered instead of simply looking them up in the customer table. In addition to increasing the data entry burden, it is also more likely that someone will enter something incorrectly. If someone needs to enter "Mark Collins" 50 times, they will likely fat-finger it at least once and end up with "Markl" or "Marl."

I recommend that the initial database design start out with at least third normal form. I realize there are times when you will want some amount of de-normalization for processing efficiency. You may choose to violate 3NF, for example, by adding an `Age` field in addition to `BirthDate`. However, add these columns later, and only if necessary. I also recommend, if you need to de-normalize, that you use data macros (explained in Chapter 3) to populate them. This will minimize the negative effects of de-

www.it-ebooks.info

normalization. You should never have to violate 2NF. Instead you can use a query to create a de-normalized view of one or more tables, which I'll explain in Chapter 4.

Database Constraints

I also recommend that the database be *constrained* as much as feasible. Adding database constraints allows you define rules for what data is considered valid. By providing these rules at the database level, you guarantee that these rules are followed, regardless of the source of the data. The database engine will not allow data to be added or modified if it violates any of these constraints. Fortunately, Access 2010 provides all the necessary features to allow you to do this.

Foreign key constraints ensure that references to other tables are (and remain) valid. For example, if you have a table with an `OrderID` field that references the `Order` table, by setting up a foreign key constraint the database will require that whatever value is entered in the field is found in the `Order` table. You will not be able to enter a value of 5, for example, if there is no `OrderID` 5 in the `Order` table. Once the data has been entered, the foreign key constraint will also prevent you from deleting `Order` #5 if another table has a reference to it.

There are several other types of constraints that can be defined as well. You can enter a validation expression to define what values are allowed. For example, use `>= 0` to ensure that negative values are not permitted. You can specify if a value is required and/or provide a default value if not specified by the user interface. I will cover each of these in more detail later in this chapter. In general, you should consider every column and provide the appropriate constraints.

Designing the Tables

This application will support a lending library. You'll need a table to store the items that you have in your inventory. The design will allow for multiple copies of the same item, so you'll have an `Item` table to define things like the title, author, and description, as well as an `InventoryItem` table to store details about a specific copy, such as status (available or checked out) and condition. You will also need a `Loan` table to record when an item is checked out and a `Customer` table to hold information about a borrower, such as name and address. This design will include a `Request` table, which will allow a customer to request an item to be held for them as soon as it becomes available. Finally, you'll need `Category` and `Media` tables to define available options when configuring the item's attributes. Figure 2-1 provides a high-level view of the basic structure.

www.it-ebooks.info

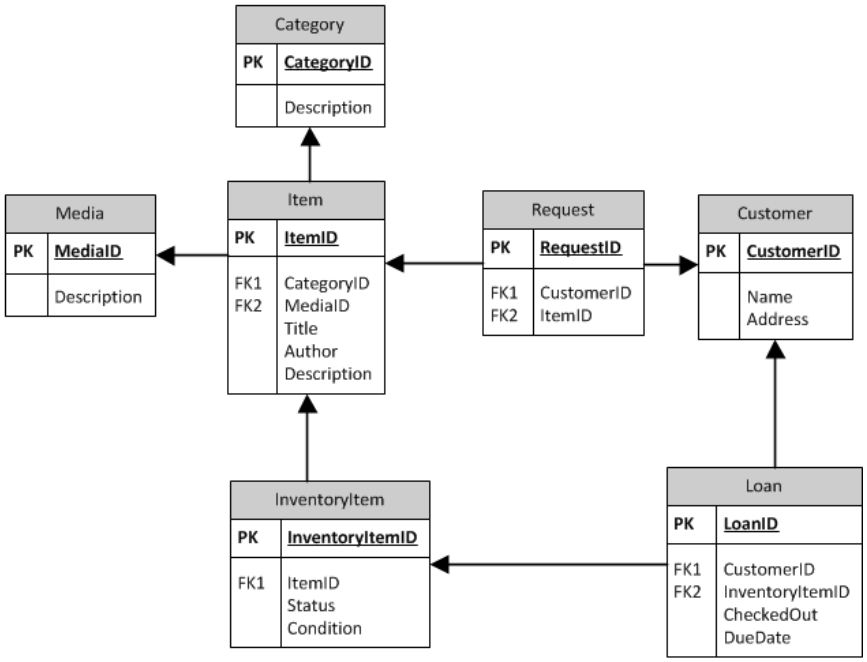


Figure 2-1*. Overview of the database schema*

I find it helpful to sketch out a diagram like this before actually creating the database objects. Seeing the tables in a visual presentation like this may help you spot missing entities or relationships. This diagram does not need to include every column, initially. Just start with enough columns to communicate the basic content of each table. Then flesh out the details as the design evolves. Figure 2-1 only contains the partial design; it's how a design might look when you're about halfway through it. When creating the actual tables in Access, you'll include other columns as well, which I will explain later.

I made this diagram using Microsoft Visio, but you can use any basic diagraming utility or just sketch it with paper and pencil. Once the design is entered in the Access database, you can produce detailed diagrams and documentation.

■ **Note** The `Customer` table in this design will not be normalized. To normalize this data, columns such as `City` should be put into their own table. I've chosen to leave this table de-normalized, as this will simplify the implementation and does not interfere with fulfilling the goal of this book.

When designing the schema, you normally start with the primary tables, like `Item` and `InventoryItem`. However, when creating them in Access it's more efficient to start with the peripheral tables, such as `Category` and `Media`. As you design a table, the tables that it is dependent on must exist before you can define the relationship between the tables. You can see from Figure 2-1 that the `Customer`,

www.it-ebooks.info

Category, and Media tables have no dependencies on other tables, so you should start with these. Then you'll create the Item and InventoryItem tables. Finally, you'll finish up with the Loan and Request tables.

Creating the Customer Table

Start the Access 2010 application and create a blank database named Library. Use the Microsoft Access 2007 database format, which should be your default option. This will create a Library.accdb file. The template automatically creates a single table named Table1 with an ID column.

Access provides two views that you can use to define the table. The default Datasheet View looks like an Excel spreadsheet. In this view, you add a column to the table by literally adding a column to the spreadsheet. I prefer to use the Design View, because it shows more details of each column. To display the Design View, just click the Design View button in the ribbon. You will be prompted to save Table1. In the Save As dialog box, enter Customer as shown in Figure 2-2.

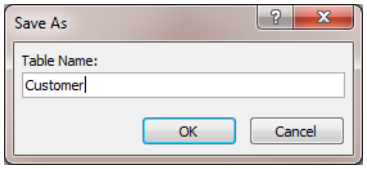


Figure 2-2. *Saving the* Customer *table*

The Design View, shown in Figure 2-3, is useful for defining the details of each column.

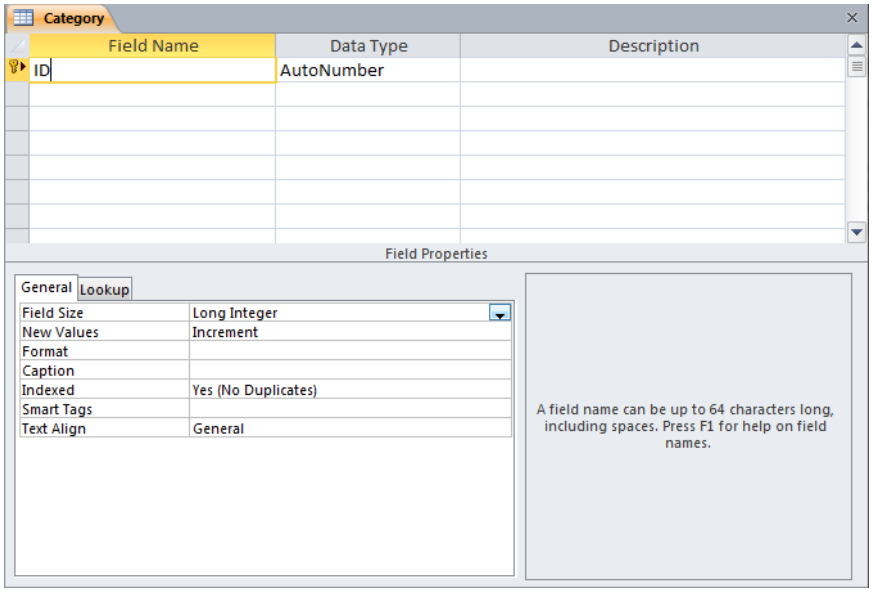


Figure 2-3. *The table Design View*

www.it-ebooks.info

The top portion lists all the columns that have been defined. The lower left pane displays all the properties of the selected column. The subset of properties that are displayed will vary depending on the data type of the column. The lower right pane provides information about the selected property.

In the Field Name column, replace `ID` with `CustomerID`. This column was set up as an `AutoNumber` field. Notice that the New Values property is set to Increment and there is a unique index on this column. This is also setup as the primary key.

Using Quick Start Fields

Access 2010 offers *quick start* fields that are preconfigured field definitions. These often contain multiple fields that are commonly used together. For example, the Name quick start field contains both a `LastName` and `FirstName` field. This is a convenient way to define a table using standard patterns.

Go back to the Datasheet View by clicking the View button in the ribbon and then select the Fields ribbon. It contains buttons that will add different types of fields. Click the More Fields button shown in Figure 2-4.

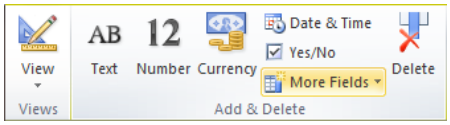


Figure 2-4. *The More Fields button*

This will display a list of additional data types that you can select from, which are grouped into categories. The Quick Start category, shown in Figure 2-5, shows the available preconfigured data types.

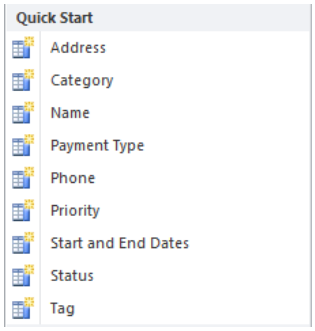


Figure 2-5. *Listing the Quick Start fields*

Select the Name quick start field, which should add the `LastName` and `FirstName` fields to your `Customer` table.

www.it-ebooks.info

■ **Caution** These fields will be added before the currently selected field. If the `CustomerID` column is selected, these fields will be added before `CustomerID`. Make sure the last column is selected, is the one labeled "Click to Add," before adding the Name field so these will be added after the `CustomerID` field. If you need to change the field order, keep in mind that if you rearrange columns in the Datasheet View by dragging the column headings, this only affects the view and not the underlying table. To rearrange the actual field order, you'll need to go to the Design View, select a row, and then move it to the desired position.

In the same way, select the Address quick start field. This will add the following fields:

- `Address`
- `City`
- `StateProvince`
- `ZIPPostal`
- `CountryRegion`

These fields as well as `LastName` and `FirstName` were added as not required and the "Allow Zero Length" property is set to No. Depending on how your library will be used, you probably want to make some of these required. I suggest that you set the Required property to Yes on at least `LastName` and `FirstName` fields. I will explain more about required fields later in this chapter.

Using the quick start fields you have defined most of the fields in the `Customer` table. Now you'll need to add fields for the phone number and e-mail address.

Adding Search Fields

The Phone quick start field adds several different phone number fields (business, home, mobile, and fax). You probably don't need all of these in your table. Save the table definition and then go to the Design View.

After the `CountryRegion` field, add the following fields:

- **PhoneNumber:** `Text`, size **15**
- **E-mail:** Text, size **70**

Both of these fields could be useful for searching for a customer, so you should create an index for them to speed up the search. Go to the Indexed property and select Yes (Duplicates OK) for both fields. You can leave the Required property as No, but change the "Allow Zero Length" property to No for both of these fields.

www.it-ebooks.info

Using an Input Mask

For certain fields, like `PhoneNumber`, you can assign an input mask that will assist the end user in formatting the data correctly. Select the `PhoneNumber` field, select the Input Mask property, and then click the button to the right of the field. This will display the Input Mask Wizard shown in Figure 2-6.

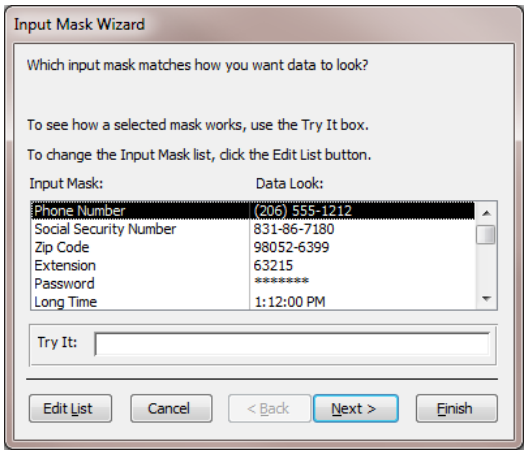


Figure 2-6. *Selecting an input mask*

Select the Phone Number mask and click the Next button, and the wizard will give you an option to test the input mask. Notice that the "_" place holder is replaced as you type each character, as shown in Figure 2-7. The formatting characters like "(" and "-" are automatically skipped as the phone number is entered.

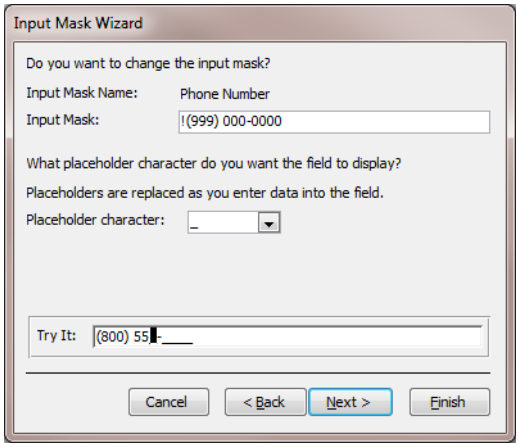


Figure 2-7. *Testing the input mask*

www.it-ebooks.info

After you click the Next button, the wizard will then ask if you want the formatting characters included or just the raw text. Select the first option, which is to store the formatting characters, as shown in Figure 2-8.

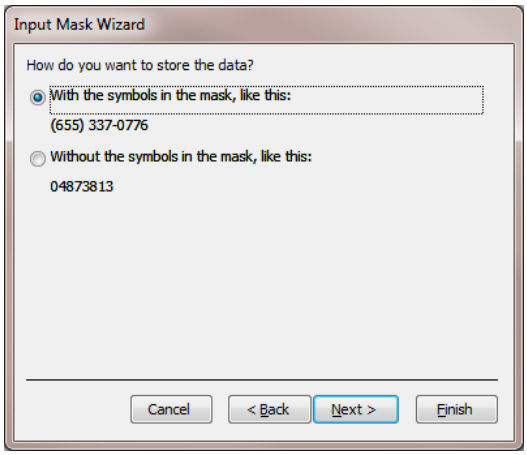


Figure 2-8. *Selecting how the formated value should be stored*

The completed properties for the `PhoneNumber` field should look like Figure 2-9.

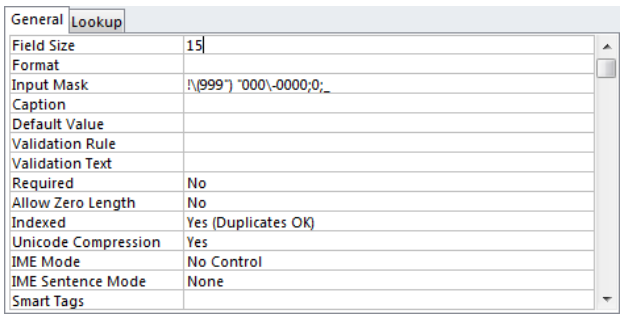


Figure 2-9. *The completed properties for the* `PhoneNumber` *field*

Adding Customers

Open the `Customer` table in the Datasheet View by clicking the View button in the ribbon. The first record will have the text "(New)" in the `CustomerID` field. Tab to the `LastName` field and enter a name. As soon as you start typing in the `LastName` field, the next ID is filled in the `CustomerID` field. Enter data in the remaining columns. When you get to the `PhoneNumber` field, notice that the input mask is displayed as demonstrated in Figure 2-10.

www.it-ebooks.info

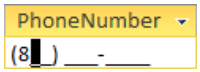


Figure 2-10. *The* `PhoneNumber` *input mask applied*

Add several customers into this table so you'll have some data when checking out items.

Creating the Category Table

The `Category` table simply defines the allowed categories and will be used to populate a dropdown list when configuring an item. Categories are used to group similar items. They can be based on topics or genre, or whatever organization makes sense based on the type of library that will use this database. By putting categories in a table, the end user can determine the set of values that work best for them.

To create a new table, select the Create ribbon. The Table button will create a table in the Datasheet View and the Table Design button will use the Design View. Click the Table Design button. In the Design View add the following columns:

- **CategoryID:** `AutoNumber`
- **CategoryCode:** `Text`, size **20**
- **CategoryDescription:** `Text`, size 50

Select the `CategoryID` column and then click the Primary Key button in the Design ribbon. This will make this the primary key field. The `Code` field will be used like a short description.

■ **Tip** For the field names, you could have used the more generic names of `ID`, `Code`, and `Description`. You can infer from the context that these attributes refer to a category. It may seem somewhat redundant to include the table name as part of the field name. However, when you start designing queries that join several tables, it could be confusing when every table has an `ID` and `Description` field. I'm not suggesting that you include the table name in every field name; you only need to do this for commonly used names. A good rule of thumb is that a field name should communicate what data is stored in the field without relying on contextual information.

Making a Field Required

There are some fields that need to be supplied or the record just isn't usable. For example, what good would it do to set up a category without supplying the code and description? To avoid getting bad data in your database, you can make critical fields required. The end user who is adding the category, would then be required to supply these fields before they can save the record. To make a field required, select the Required property and choose Yes from the dropdown list. Do this for the `CategoryCode` field.

For Text fields there is also an "Allow Zero Length" property. Access makes a distinction between a null value and an empty string, as do most database engines. In practice, however, we seldom care about that distinction. If the "Allow Zero Length" property is set to Yes, the end user could enter a blank string and still satisfy the Required constraint. Set this property to No to ensure some data is entered in this field.

www.it-ebooks.info

Go to the CategoryDescription field and also set its Required property to Yes and the "Allow Zero Length" property to No.

■ **Caution** Unless you have a good reason not to, you should always set "Allow Zero Length" to No when Required is set to Yes. If you try to publish your database to SharePoint, which I'll explain in a later chapter, you'll get a compatibility error if both of these properties are set to Yes.

Adding a Unique Constraint

There are times when we need to ensure certain values are unique. For example, if two categories were created that used the same code, the code would be ambiguous. In Access you can ensure unique values by adding an index and selecting the No Duplicates option. Go to the CategoryCode field and select this option for the Indexed property. The properties of the CategoryCode field should look like Figure 2-11.

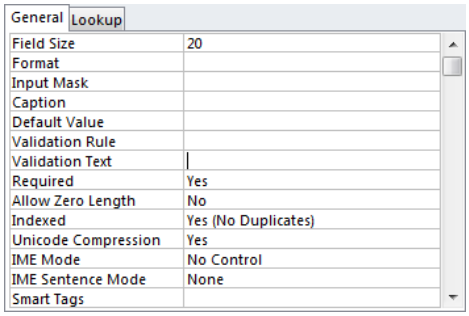


Figure 2-11. *The CategoryCode field properties*

Click the Save icon at the top of the window to save your changes to the database.

Defining Categories

Open the Category table in the Datasheet View. The first record will have the text "(New)." Tab to the CategoryCode field and enter a code. Tab to the CategoryDescription field and enter a description. Click the Tab key and the record will be saved. You may want to change the column widths, which you can do by dragging the gridlines. Add several more records to the table. The Datasheet View should look like Figure 2-12.

www.it-ebooks.info

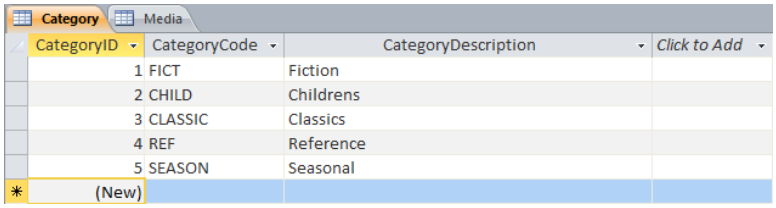


Figure 2-12. The Category table

Creating the Media Table

The Media table, like the Category table, defines attributes that will be applied to each item. Whereas the Category table is used for organizing topically, the Media table is used to define formats such as book, video, and audio. The Media table will also define business rules, such as how long an item can be checked out and what is the charge for each day an item is overdue. This will enable you to limit videos to be kept for only a week, for example, and allow a longer period for books.

To create a new table, click the Table Design button in the Create ribbon. In the Design View add the following columns:

- **MediaID:** AutoNumber
- **MediaCode:** Text, size **20**
- **MediaDescription:** Text, size **50**
- **LoanPeriod:** Number, size Integer
- **RenewalsAllowed:** Number, size Integer
- **OverdueFee:** Currency

Select the MediaID column as the primary key.

■ **Note** When you added the MediaID column as an AutoNumber field, it was initially setup with a non-unique index, which means that duplicate values were allowed. When you selected this as the primary key, the index was automatically changed to not allow duplicates. That is how you want this to be defined.

Click the Save icon. When prompted, enter **Media** for the table name. Just like you did for the Category table, make the MediaCode and MediaDescription fields required and don't allow zero length strings. Also, for the MediaCode field, select No Duplicates for the Indexed property. This will prevent the same code being used twice.

www.it-ebooks.info

Adding Field Validation

Select the `LoanPeriod` column. This will define the number of days an item can be checked out. You'll add validation logic to ensure that a reasonable value is entered. Let's assume that this must be at least 7 days but not more than 21. Select the Validation Rule property and enter **>=7 And <=21**. Notice that when you select this property, a button with ellipses appears to the right of the value field. You can click this button to display the Expression Builder, shown in Figure 2-13.

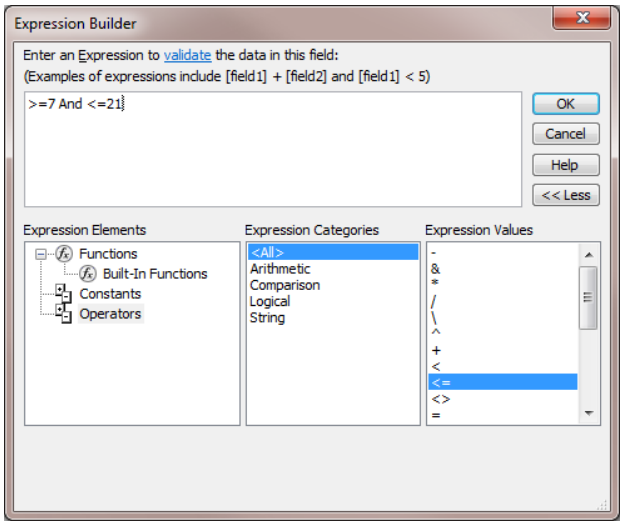


Figure 2-13. *The Expression Builder*

Expressions in Access use the Visual Basic syntax. If this is not familiar to you, the Expression Builder is a useful tool for looking up built-in functions and operators.

■ **Tip** Validation rules can include other fields in the logic. For example, the `LoanPeriod` defines the number of days for the initial loan and `RenewalsAllowed` specifies how many times it can be renewed. Suppose to you wanted to ensure that an item was never checked out for more than 60 days. You could add a validation rule to the `RenewalsAllowed` field as `<=60/[LoanPeriod]`.

If you enter a validation rule, you should also specify the validation text. This text will be displayed when the user tries to update a field and its validation rule fails. Select the Validation Text property and enter

```
The loan period must be between 7 and 21 days
```

Set the Required property to Yes. The Decimal Places property defaults to `Auto`. Because you are expecting only whole numbers, change this to 0. Notice that when you change the Decimal Places

www.it-ebooks.info

property a lightning bolt appears next to it. This is known as the Property Update Options button. If you click this, you'll see the options shown in Figure 2-14.

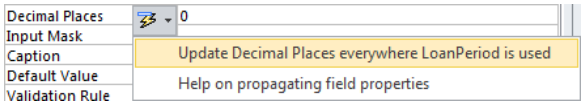


Figure 2-14. *Propagation options*

If you choose to update every place that this field is used, a dialog box will appear listing the fields that will be updated. Currently, there are no other fields called `LoanPeriod` so you can ignore this.

The completed properties should look like Figure 2-15.

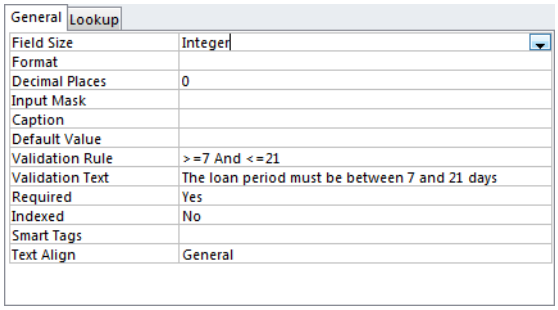


Figure 2-15. *The completed* `LoanPeriod` *properties*

Entering a Field Caption

When a field is displayed on a form or report, by default the Field Name will be used as the label for this field. In many cases this is appropriate. However, if you specify the Caption property, this will be used instead of the Field Name. Select the `RenewalsAllowed` field, select the Caption property, and enter

```
How many times can a loan for this type of item be renewed?
```

Change the Decimal Places property to **0**.

Using Default Values

For fields that are likely to be used in an expression, you should ensure that they have a value. If you don't, then any expression that uses this field needs to handle a null value. It's easier to simply set the Required property to Yes and supply a Default Value. By setting both of these properties you resolve the null issue without adding any extra data entry work. The user only has to enter a value if they want something other than the default.

I realize that there are some situations where you need to take a different action when there is a null value. In this case, make sure the Required property is set to No and there is no Default Value. You'll then need to handle this case in the expression that uses this field.

www.it-ebooks.info

Select the RenewalsAllowed field and set the Required property to Yes and enter a Default Value of **0**. Also, enter **>=0** for the Validation Rule. For the Validation Text property enter **Negative values are not allowed**.

Select the OverdueFee field and set the following properties:

- Default Value: **0**
- Validation Rule: **>=0**
- Validation Text: **The fee cannot be negative**
- Required: **Yes**

Save the changes to this table.

Defining Media Types

Open the Media table using the Datasheet View by clicking the View button in the ribbon. The MediaID field should be selected with the default value of "(New)." Tab to the MediaCode field and enter a code and then enter a description. For the LoanPeriod, enter an invalid value such as 3. When you tab off that field your validation text should be displayed as shown in Figure 2-16. Click the OK button and then enter a valid value.

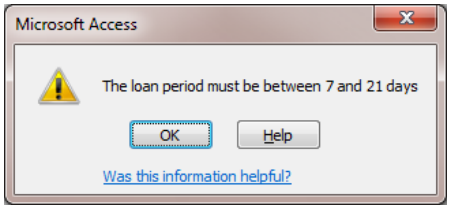


Figure 2-16. *A field validation prompt*

You should get similar results if you try to enter negative values for the RenewalsAllowed or OverdueFee fields. If you try to save a record without entering a LoanPeriod, you'll see the prompt shown in Figure 2-17.



Figure 2-17. *Required data prompt*

Enter several media types so you'll have some data available when defining items. The Media table will look similar to Figure 2-18.

www.it-ebooks.info

Media

MediaID	MediaCode	MediaDescription	LoanPeriod	How many tin	OverdueFee	Click to Add
1	DVD	DVD Video	7	1	$1.00	
2	CD	CD Audio	7	2	$0.50	
3	HDBK	Hardback Book	21	1	$0.25	
4	PAPER	Paperback Book	14	1	$0.10	
5	PERIOD	Periodical	14	0	$0.10	
(New)				0	$0.00	

Figure 2-18. *The* `Media` *table with data populated*

Creating the Item Table

The `Item` table will store information about the items that are available in your library. Create a new table using the Table Design button and enter the following columns:

- **ItemID**: `AutoNumber`, primary key
- **Title**: `Text`, size **255**

For the `Title` field, set the Required property to Yes and the "Allow Zero Length" to No. Save the table and enter the name as **Item** when prompted.

Creating Lookup Columns

As I mentioned earlier, you'll need to assign a category and media type to each item. To do that, you'll add `CategoryID` and `MediaID` columns, which will reference the associated record in the corresponding tables. Adding the columns to the `Item` table merely provides a place to store the values, but does not define a relationship between the tables.

■ **Note** In previous versions of Access, you would first create the tables with the appropriate columns and then define a relationship between the tables. While creating the relationship, you would indicate the primary and foreign keys in each table. You can still define relationships this way in Access 2010; however, they may not be compatible if you want to publish your database to SharePoint. Instead use the Lookup Wizard, which will add the column and create the relationship.

From the Design View, add a field named **CategoryID** and select Lookup Wizard for the data type. This will start the Lookup Wizard shown in Figure 2-19.

www.it-ebooks.info

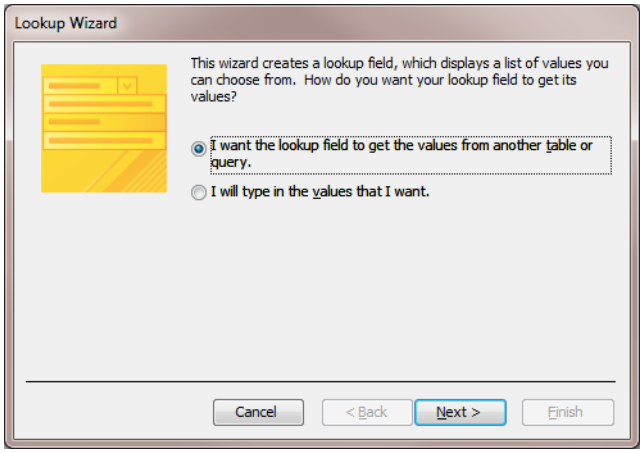


Figure 2-19*. Selecting the type of lookup*

The Lookup Wizard can either get values from another table or you can specify a fixed set of values to use. Select the first radio button and click the Next button. Since you selected to get the values from a table or query, the next dialog box, shown in Figure 2-20, allows you to select the table.

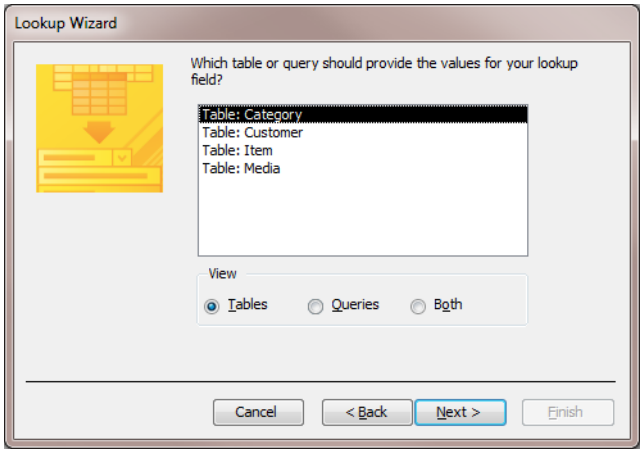


Figure 2-20*. Selecting the referenced table*

Select the `Category` table and click the Next button. In the next dialog box, you'll select the applicable fields from this table. In this case, click the ">>" button. All three fields should be listed in the Selected Fields list as shown in Figure 2-21.

www.it-ebooks.info

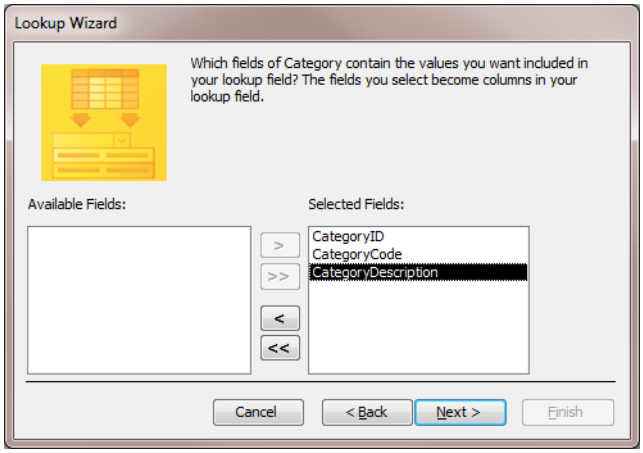


***Figure 2-21**. Selecting the fields to be used*

The last statement in this dialog box is a little bit misleading. It says "The fields you select become columns in your lookup field." All of the selected fields are used when looking up a value in the referenced table. However, only the key field, `CategoryID`, is added as an actual column to the `Item` table.

Click the Next button to display the next dialog box, which allows you define how the choices will be sorted. Select the `CategoryDescription` column as shown in Figure 2-22 and click the Next button.

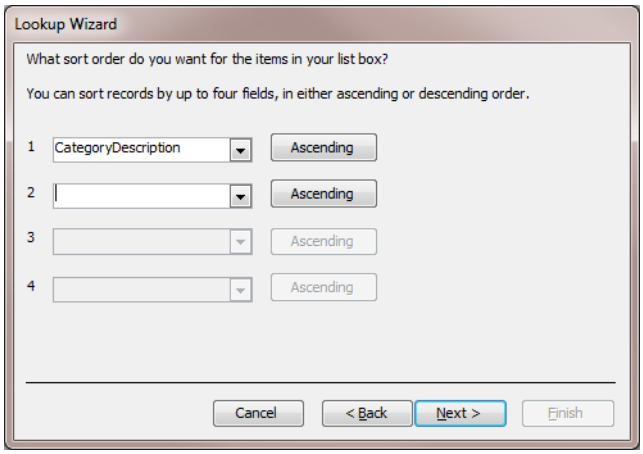


***Figure 2-22**. Defining the sort order*

The `Category` table that you created earlier will be used to populate a dropdown list in the `Item` table. The next dialog box, shown in Figure 2-23, shows a preview of what the dropdown list will look like. You can resize the column by dragging the grid lines.

www.it-ebooks.info

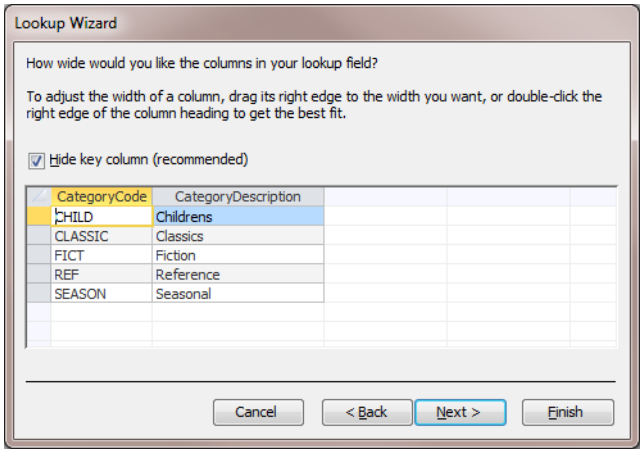


Figure 2-23. *Configuring the dropdown list*

The "Hide key column" check box controls whether the key column, `CategoryID`, is included in the dropdown list. Typically, the ID is auto-generated and not usually meaningful to the end user. The code and description, as shown here, is usually sufficient. If the ID would be helpful, you can un-select this check box and it will be included as a separate column in the dropdown list.

Click the Next button to display the final dialog box shown in Figure 2-24.

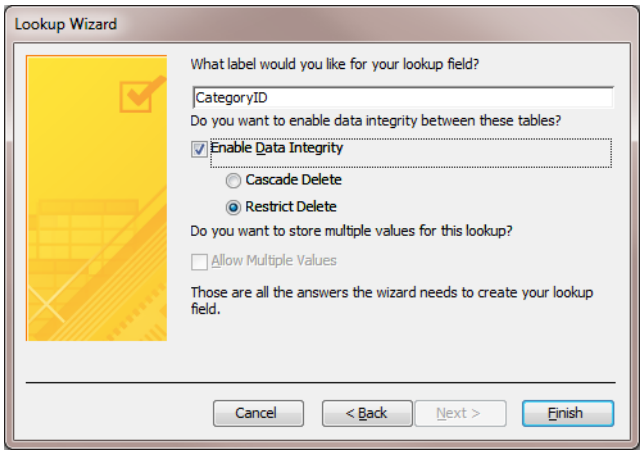


Figure 2-24. *Specifying the data integrity rules*

The last step in defining the relationship is to specify the data integrity rules. You can choose to disable data integrity, but I recommend you use it to help maintain valid data. There are two ways that data integrity can be enforced.

The first is called Cascade Delete. In this mode, if the record being referenced is deleted, the records that reference it are also deleted. You use this mode when you have an aggregate relationship; that is,

www.it-ebooks.info

when the two tables have a parent-child relationship. For example, an `Order` table would be the parent of an `OrderItem` table. In this case, if an `Order` record is deleted; all `OrderItem` records that reference it would be automatically deleted as well.

In our case, however, the Item and `Category` tables do not have that kind of relationship. Instead, use the Restrict Delete option. This will prevent someone from deleting a `Category` record if any Item record references it. If you want to delete the `Category` record, you must first either delete these items or change them to reference a different category.

Select the Restrict Delete option and click the Finish button. You will then see the prompt shown in Figure 2-25.

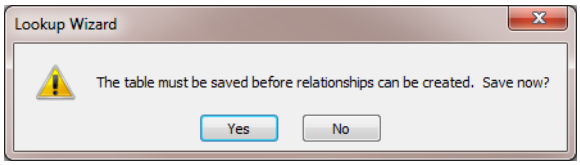


Figure 2-25. *Saving the table changes*

The table has been modified by the wizard to include the `CategoryID` column. This must be saved before the wizard can create the relationship. Click the Yes button. Then click the Save icon to save all the changes made by the wizard. You may see the dialog box shown in Figure 2-26.

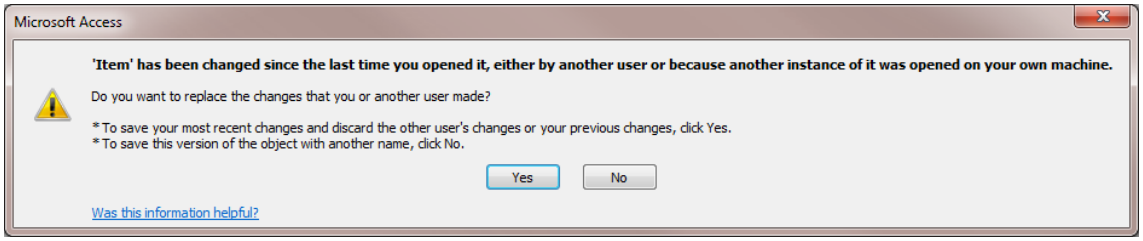


Figure 2-26. *Message that the `Item` table has been modified*

In my opinion, this is a very confusing dialog box. The wizard has made some changes to the `Item` table and this message is letting you know that someone (or something) other than you has modified an entity that you have open for editing. Essentially, the version that you have in memory is different from what is stored on disk. If you choose No, you'll be able to save your current version (in memory) to disk with a new name. This will keep both versions. The text does not make it clear, however, what happens when you choose Yes. Since you don't have any changes that you need to worry about losing, click the Yes button.

■ **Caution** Select the `CategoryID` field and check the Lookup tab on the Field Properties pane. The "Limit to List" property should be set to Yes. The wizard should be setting this field, but I have found instances where it did not. This will give you a compatibility issue if you try to publish this database to SharePoint.

---

www.it-ebooks.info

Adding the MediaID Column

Now, in the same way, you'll add a reference to the `Media` table. Add a new field named, `MediaID` and select the Lookup Wizard for the data type. This will take you through the same set of dialog boxes that you used to setup the `CategoryID` field.

- In the second dialog box, select the `Media` table.
- In the third dialog box, select the `MediaID`, `MediaCode`, and `MediaDescription` fields.
- In the fourth dialog box, sort by the `MediaDescription` field.
- In the sixth dialog box, select the "Enable Data Integrity" checkbox and the Restrict Delete option.

Viewing the Relationships

In addition to adding the foreign key fields, the Lookup Wizard has also created a relationship between the tables. Select the Database Tools tab of the ribbon and then click the Relationships button. You may need to click the All Relationships button to refresh the view. The relationships should look like Figure 2-27.

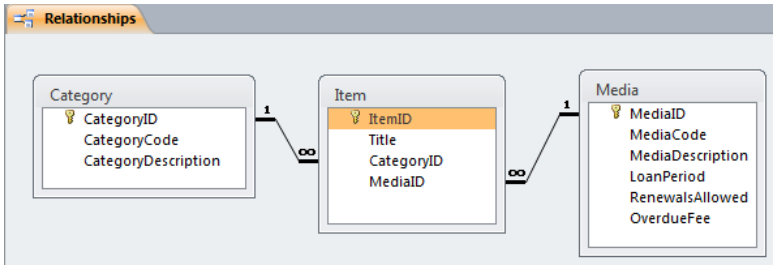


Figure 2-27. *The relationships view*

Adding the Remaining Columns

Close this view and display the `Item` table using the Design View. Add the following columns to this table:

- **Author:** `Text`, size **255**
- **Description:** `Memo`
- **ReplacementCost:** `Currency`

Make the `Author` field required and set the "Allow Zero Length" property to No. You can leave the `Description` field with the Required property set to No. For the `ReplacementCost` field, enter **>=0** for the Validation Rule property and enter the Validation Text as **Negative values are not allowed**.

www.it-ebooks.info

Adding a Calculated Field

A calculated field is computed based on an expression that can use other fields. You'll use a calculated field to define the `LostFee`. The `ReplacementCost` field specifies the actual cost to purchase a single copy of the item. The `LostFee` is what the customer is charged when an item that they have checked out is lost or damaged beyond repair. The fee that you charge the borrower could be the `ReplacementCost` plus a flat re-stocking fee. Or it could be based on a percentage, say a 15% handling fee. For our project we'll add a flat $10 to the `ReplacementCost` field.

In the Design View, enter the Field Name as **LostFee** and select `Calculated` for the Data Type. This will display the Expression Builder. Double-click the `ReplacementCost` field to add it to the expression, then add **+ 10** and click the OK button to close the dialog box. For the Result Type property, select `Currency` from the dropdown list. The completed properties should look like Figure 2-28.

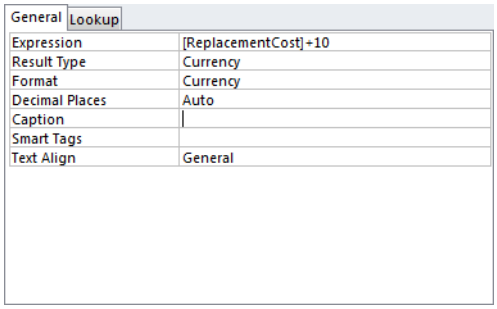


Figure 2-28. *The `LostFee` properties*

Adding Items to the Database

Open the `Item` table using the Datasheet View and enter some items into your database. Notice that when you select a category or media type, the dropdown list displays both the code and description. Once you select a value, only the code is displayed in the table. For lookup columns, the table automatically displays the first column from the dropdown list.

Also, as soon as you enter a value in the `ReplacementCost` field, the `LostFee` is automatically calculated. The `LostFee` is also read-only; you are not allowed to modify it because it is computed based on the formula you entered. The Datasheet View will look like Figure 2-29.

Item

ItemID	Title	CategoryID	MediaID	Author	Description	ReplacementCost	Lost Fee
1	A Christmas Carol	CLASSIC	HDBK	Charles Dickens	Miserly old Ebenee:	$11.95	$21.95
2	A Tale of Two Cities	CLASSIC	PAPER	Charles Dickens	A moving story abou	$7.99	$17.99
3	White Christmas	SEASON	CD	Bing Crosby	Seasonal favorites s	$7.50	$17.50
4	It's a Wonderful Life	SEASON	DVD	Frank Capra	George Bailey learn	$8.75	$18.75
(New)							

Figure 2-29. *The Datasheet View of the `Item` table*

www.it-ebooks.info

Creating the ItemInventory Table

The `Item` describes the titles that are available in your library. However, you will need to track specific copies of these titles. To do that you'll create an `ItemInventory` table. This will provide specific information about each copy such as status and condition and reference an `Item` record for generic details.

From the Create ribbon, click the Table Design button. Enter the Field Name **InventoryItemID** with an `AutoNumber` Data Type. Click the Primary Key button and save the table. Enter the name **InventoryItem** when prompted.

Adding a Lookup to the Item Table

Add another field named `ItemID` and choose the Lookup Wizard for the Data Type. You'll configure this like you did the other lookup columns, `CategoryID` and `MediaID`.

- Select the `Item` table in the second dialog box.
- In the third dialog box, select the `ItemID`, `Title`, and `Author` fields as shown in Figure 2-30.

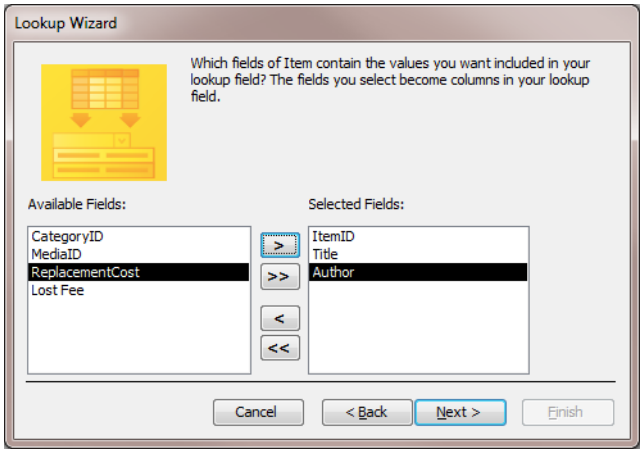


Figure 2-30*. Selecting the fields to be included in the lookup*

■ **Tip** You don't have to explicitly include the primary key, `ItemID`. If you don't include this field, the wizard will add it for you. Also, the order in which you add the fields to the Selected Fields list will determine the order they are displayed in the dropdown list. For example, if you wanted the `Author` shown before the `Title`, then add it to the Selected Fields first. You can also adjust the field order in the fifth dialog box, shown in Figure 2-31. Just select a column and drag it to the desired location.

---

www.it-ebooks.info

In the fourth dialog box, select `Author` for the first sort field and then `Title` for the second sort field. This will group the items by author in the dropdown list.

As I mentioned, for lookup columns, the table displays the first column of the dropdown list. So you can control which column is displayed. If you wanted the description shown instead of the code, just make sure the description is the first column in the dropdown list. For relatively short lists, like `Category` and `Media`, displaying the code or description works well.

For the `Item` table, which could have thousands of records, you'll find that displaying the `ItemID` will be more useful. Keep in mind that you will be developing forms for the end user; they will not typically look at the table directly. Developers and maintainers will use the table and they'll want to know which specific `Item` record is being referenced by this `InventoryItem` record. Titles are often non-unique. You could have three different items with the title "White Christmas."

To display the `ItemID`, unselect the "Hide key column" checkbox, as shown in Figure 2-31.

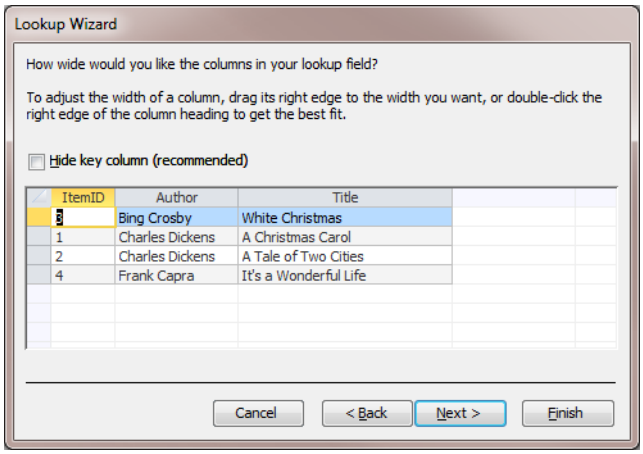


***Figure 2-31**. Configuring the dropdown list*

Depending on how the fields were selected and configured, you may see the dialog box shown in Figure 2-32. This allows you to specify the field that will be stored in the `InventoryItem` table, and it needs to be the primary key, `ItemID`.

www.it-ebooks.info

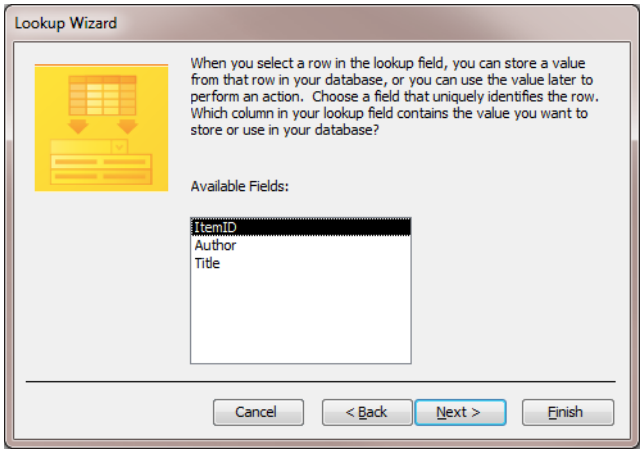


Figure 2-32. *Selecting the key field to be stored in the* `InventoryItem` *table*

In the final dialog box, select the "Enable Data Integrity" checkbox. In this scenario, the `InventoryItem` table is a child of the `Item` table and you could choose the Cascade Delete option. If you did this, when an `Item` record is deleted, all of the associated `InventoryItem` records are also deleted. I think this is dangerous. I believe it would be better to force the user to first remove the inventory items before allowing them to remove an item. To do this, select the Restrict Delete option. Then click the Finish button to setup the relationship.

■ **Caution** If the `Item` table is open in another tab, you will probably get an error because the wizard is not able to modify the `Item` table. If you do, close the `Item` table. You will need to re-run the Lookup Wizard. Change the Data Type from `Number` to `Lookup Wizard`, which will re-start the wizard.

Go to the Lookup tab of the Field Properties pane and make sure the "Limit to List" property is set to Yes.

Adding a Lookup with Fixed Options

You'll now add a `Status` field, which should be selected from a list of standard values. In the Design View add a new field named **Status** and select Lookup Wizard for the Data Type. In the first dialog box, select the second option, which lets you specify the values to use, as shown in Figure 2-33.

3
www.it-ebooks.info

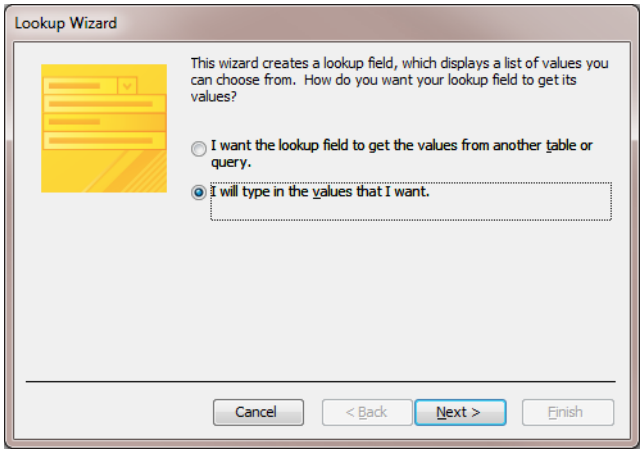


Figure 2-33*. Selecting the lookup source*

Then click the Next button, which will display the second dialog. In addition to the actual status value, you can add one or more columns to provide information in the dropdown list. You may want to include a comment with each value that explains when it should be used. In this case, the status values are fairly self-explanatory so a single column will be sufficient.

Enter the following values as shown in Figure 2-34 and click the Next button.

- **Available**
- **Checked Out**
- **On Hold**
- **Damaged**
- **Lost**

www.it-ebooks.info

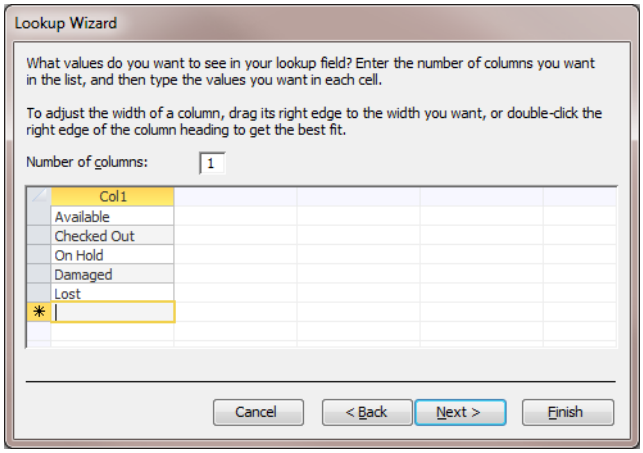


Figure 2-34. *Entering the allowed values*

In the third and final dialog box, select the "Limit to List" checkbox, as shown in Figure 2-35. This will prevent the user from entering a value that is not in the predefined list.

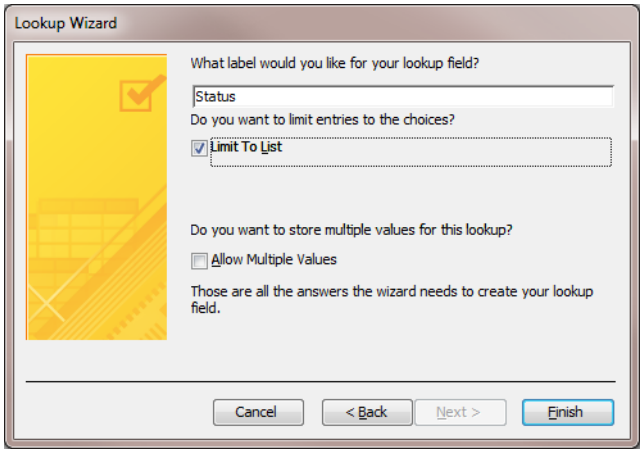


Figure 2-35. *Setting the "Limit to List" option*

Click the Finish button to complete the setup of this field. Select the Lookup tab in the Field Properties pane, which is shown in Figure 2-36.

www.it-ebooks.info

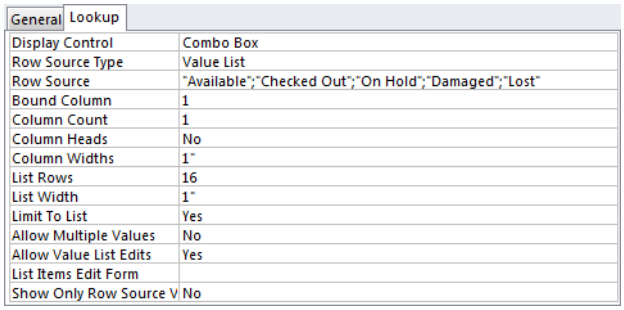


Figure 2-36. *Displaying the Lookup properties*

The Lookup Wizard sets these properties using information entered in the dialog boxes. You could enter these properties yourself manually if you prefer.

Go back to the General tab. Enter **"Available"** for the Default Value property, set the Required property to Yes, and set the "Allow Zero Length" property to No.

Specifying the Item's Condition

You'll need two more columns in the `InventoryItem` table, which you'll use to record the condition of this item. The first will be a dropdown list with standard values such as New, Good, and Fair. The second will be a text field that you can enter freeform comments.

From the Design View add a new field named **Condition** and select `Lookup Wizard` for the Data Type. Configure this just like you did for the `Status` field except enter the following values:

- **New**
- **Good**
- **Fair**
- **Poor**

After the wizard has configured the field, enter **New** as the default value. As with the `Status` field, set the Required property to Yes, and set the "Allow Zero Length" property to No.

From the Design View, enter a new field named **Comment** with a Data Type of `Text`. You can leave all of the default properties, including leaving Required as No.

Adding InventoryItem Records

Open the `InventoryItem` table in Datasheet View. Click the dropdown icon in the `ItemID` column. Then select an item from the list as shown in Figure 2-37.

www.it-ebooks.info

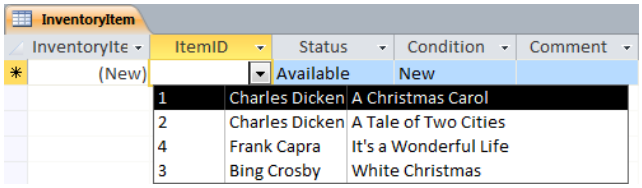


Figure 2-37. *Selecting an* `Item` *from the dropdown list*

The remaining required columns all have default values so the `ItemID` is the only one you'll need to enter. You can pretty quickly enter a couple on `InventoryItem` records for each `Item` that you defined earlier. The `Status` should be Available for all records, since you have not yet loaned anything out. You can change the `Condition` to some other values for a few of the records. When you're done, the Datasheet View should look like Figure 2-38.

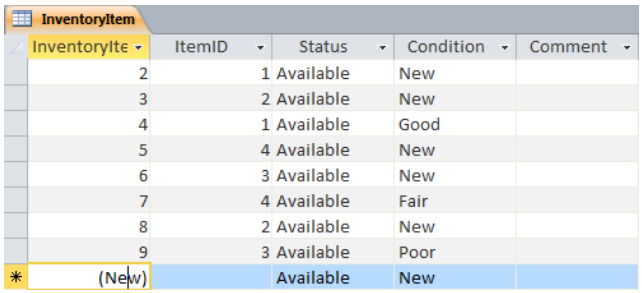


Figure 2-38. *The* `InventoryItem` *table populated with data*

Creating the Loan Table

The `Loan` table will be used to record each time an item is checked out. It will reference an `InventoryItem` record, so you'll know the specific copy that was lent out. It will also reference a `Customer` record to track who checked it out.

Create a new table in the Design View by clicking the Table Design button in the Create tab of the ribbon. Enter a new field named **LoanID** using the `AutoNumber` Data Type. Click the Primary Key button and save the record. Enter the table name **Loan** when prompted.

Referencing the Customer Table

From the Design View, perform the following steps:

1. Add a new field named **CustomerID** and select the `Lookup Wizard` Data Type.
2. In the second dialog box select the `Customer` table.
3. In the third dialog select the following columns to be used in the lookup:

www.it-ebooks.info

- `CustomerID`
- `LastName`
- `FirstName`
- `Address`

4. In the fourth dialog box, sort by `LastName` and then by `FirstName`.
5. In the fifth dialog box, unselect the "Hide key column" checkbox, as shown in Figure 2-39. You should also resize the columns so the data will fit in each column.

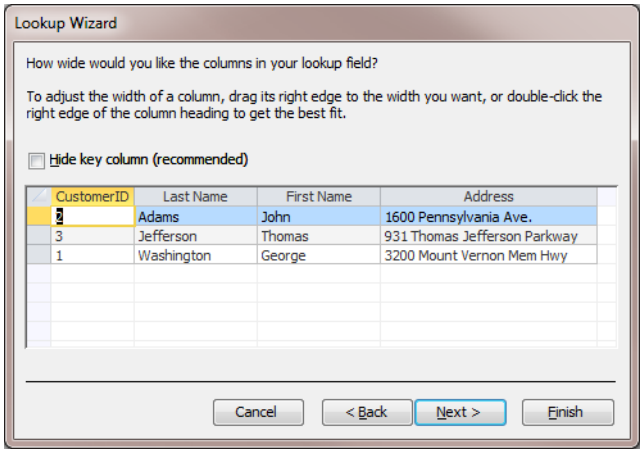


Figure 2-39. *Configuring the dropdown list*

6. In the next dialog box, select the `CustomerID` field to be stored in the `Loan` table, as shown in Figure 2-40.

www.it-ebooks.info

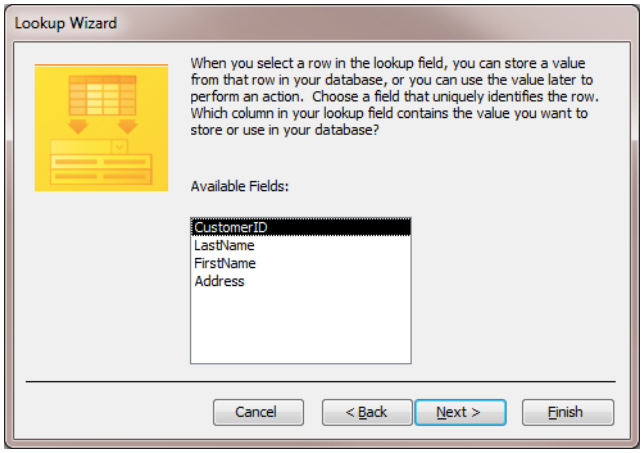


***Figure 2-40**. Selecting the value to be stored in the Loan table*

7. In the final dialog box, select the "Enable Data Integrity" checkbox and choose the Restrict Delete option, as shown in Figure 2-41.

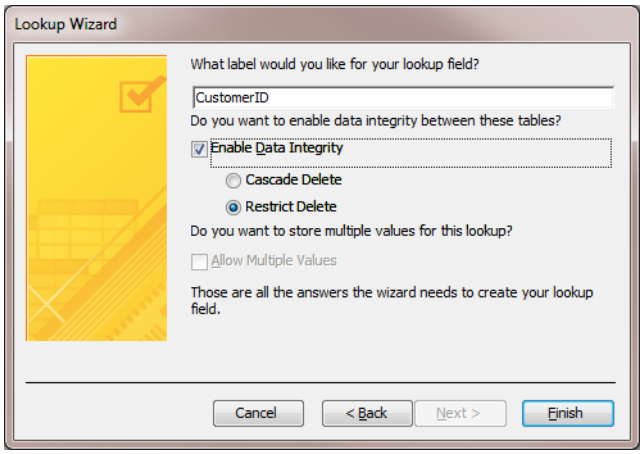


***Figure 2-41**. Enabling data integrity*

Referencing the InventoryItem table

Add a new field name `InventoryItemID` and select the `Lookup Wizard` Data Type. You'll configure this like you did with the other lookup columns.

- In the first dialog box, select the first option, which is to get the values from another table.

www.it-ebooks.info

- In the second dialog box, select the `InventoryItem` table.

The `InventoryItem` table doesn't have any columns that would be useful in a dropdown list. The `Title` is in the `Item` table, not the `InventoryItem` table, and the Lookup Wizard is not able to use referenced tables. Normally, when checking out an item, rather than looking up the item, you would scan or key the `InventoryItemID` from a barcode attached to the item. So you'll just use the `InventoryItemID`.

- Add only the `InventoryItemID` field to the Selected Fields list in the third dialog box.
- In the final dialog box, select the "Enable Data Integrity" checkbox and choose the Restrict Delete option.
- After the wizard has finished, go to the Lookup tab of the Field Properties pane and make sure the "Limit to List" property is set to Yes.

Adding the Remaining Fields

Now you'll add a few more fields to record when the item was checked out, when it is due back, when it was actually returned and if there are any overdue fees associated with this loan.

1. From the Design View add the following fields:
   - **CheckedOut**: `Date/Time`
   - **DueDate**: `Date/Time`
   - **CheckedIn**: `Date/Time`
   - **Renewals**: `Number`, size `Integer`
   - **OverdueFee**: `Currency`
2. Select the `CheckedOut` field. For the Default Value property, enter **Now**(). This is a Visual Basic function that returns the current date/time. Set the Required property to Yes, and for the "Show Date Picker" property, select Never.
3. For the `DueDate` field, select Short Date for the Format property. The due date should be a date that does not include the time portion.
4. For the `CheckedIn` field, you can leave all of the default properties.
5. For the `Renewals` field modify the following properties:
   - Decimal Places: **0**
   - Caption: **How many times has then loan been renewed?**
   - Default Value: **0**
   - Validation Rule: **>=0**
   - Validation Text: **Negative values are not allowed**
6. For the `OverdueFee` field, modify the following properties:

www.it-ebooks.info

- Default Value: **0**
- Required: **Yes**

We'll wait until Chapter 3 to start entering data in this table.

Creating the Request Table

The `Request` table is used to store a request by a customer for a particular item. When a copy of that item becomes available, it will be placed on hold and reserved for that customer. The `Request` table will reference an `Item` record, which specifies the item that is being requested. It will also reference a `Customer` record to store the customer who made the request. It will also reference an `InventoryItem` record that indicates the specific copy that is being held for the customer.

Create a new table in the Design View. Add a field named **RequestID** and select the `AutoNumber` Data Type. Click the Primary Key button and save the table. Enter the name **Request** when prompted.

- Add a new field named **CustomerID** and select `Lookup Wizard` for the Data Type. Configure this exactly the same as you did for the `CustomerID` field on the `Loan` table.
- Add another field named **ItemID** and select the `Lookup Wizard`. Configure this field exactly the same way as you did for the `ItemID` field on the `InventoryItem` table. Make sure that both `CustomerID` and `ItemID` have the Required property set to Yes.
- Add another field named **RequestDate** using the `Date/Time` Data Type. For the Default Value property enter **Now**(). Set the Required property to Yes and the "Show Date Picker" property to Never.

Defining the Status Field

Add another field named **Status**. Use the Lookup Wizard to specify a list of valid values like you did for the `Status` field on the `InventoryItem` table. In this case, you'll add an extra column to provide more details. In the second dialog box, change the number of columns to 2. Then enter the following for the list of allowed values:

- **Pending**: Waiting for an item to become available.
- **Ready**: The item is ready for the customer to pick up.
- **Complete**: The customer has picked up the item.
- **Cancelled**: The request has been cancelled.

The dialog box should look like Figure 2-42.

www.it-ebooks.info

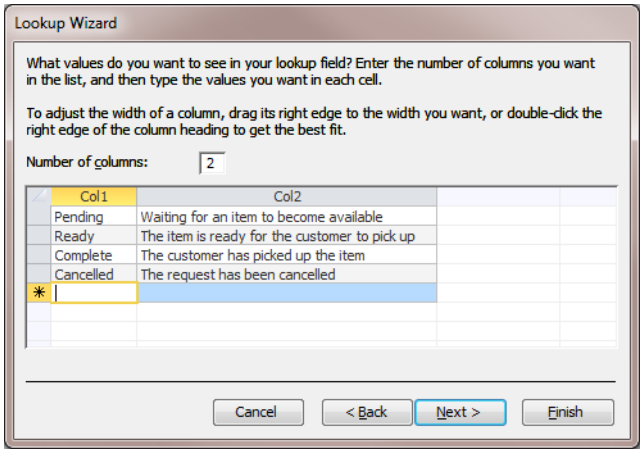


Figure 2-42. *Entering the* `Status` *values*

Because there are multiple columns, you'll need to specify which one is stored in the `Request` table. In the next dialog box, select `Col1` as shown in Figure 2-43.

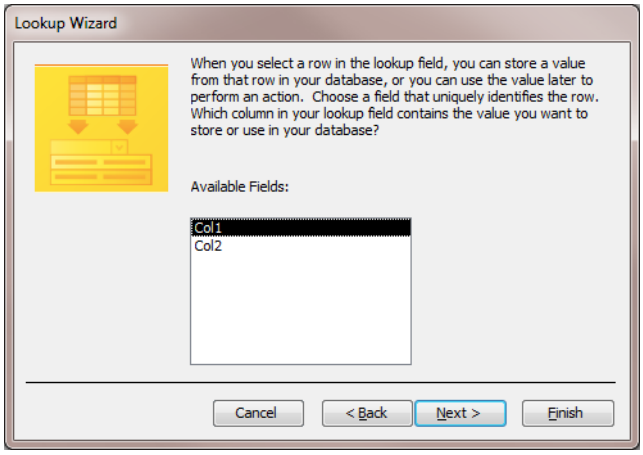


Figure 2-43. *Selecting the column to be stored in the* `Request` *table*

After the Lookup Wizard is finished:

- Set the Default Value property to Pending
- Set the Required property to Yes
- Set the "Allow Zero Length" property to No

www.it-ebooks.info

Adding the InventoryItemID Field

Add another field named **InventoryItemID** and use the Lookup Wizard to configure this exactly like you did for the `InventoryItemID` field on the `Loan` table. The only difference, however, is that the Required property should be set to No. This value will not be assigned until later, when an item becomes available.

For all three lookup fields, `CustomerID`, `ItemID`, and `InventoryItemID`, go to the Lookup tab of the Field Properties pane and make sure the "Limit to List" property is set to Yes.

Viewing the Relationships

As you have been creating tables and defining their fields, the Lookup Wizard has been creating table relationships. You can view these in a graphical presentation. From the Database Tools tab of the ribbon, click the Relationships button. You may need to click the All Relationships button to refresh the view. Drag the tables around to simplify the connecting lines. After some re-arranging, the view should look like Figure 2-44.

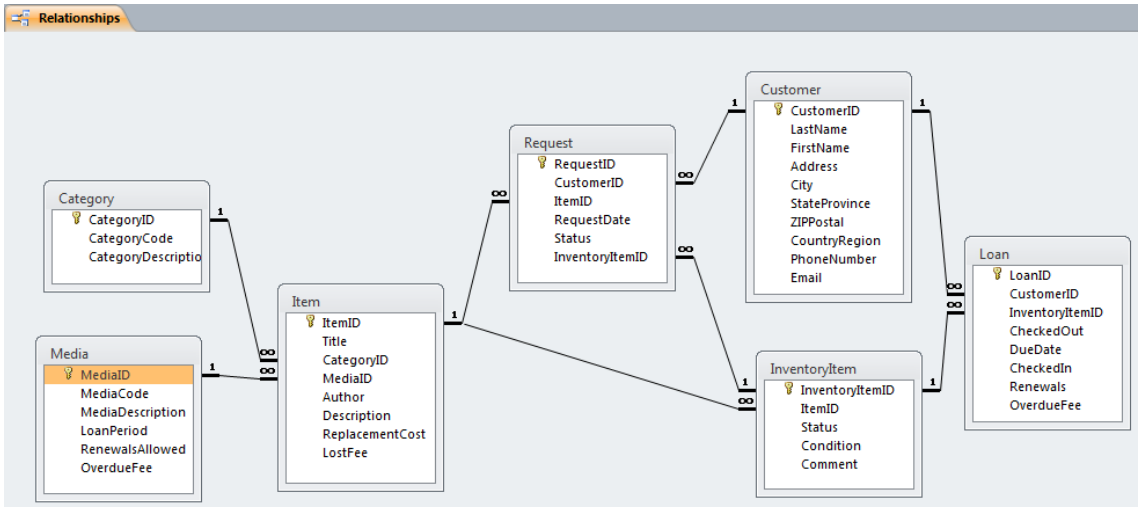


Figure 2-44. *Viewing the relationships*

Summary

In this chapter, you created an Access database and defined a set of tables, which will form the foundation for the remainder of this book's project. You used the Lookup Wizard to define the foreign keys and the table relationship. The tables are well constrained, which will help ensure data integrity. The advanced features that you used include the following:

www.it-ebooks.info

- Using quick start fields
- Using an input mask
- Creating a unique constraint
- Using default values
- Defining field validation
- Using calculated fields
- Defining fixed value lookups

In the next chapter you'll use data macros to further enhance the database design

www.it-ebooks.info

CHAPTER 3

Using Data Macros

Data macros are arguably the best new feature in Access 2010. In the previous chapter, I showed you how to use foreign key and other constraints to ensure data integrity. Data macros will allow you to take this to a whole new level. Constraints keep the user from doing bad things, while data macros can do good things to keep the data in sync. For example, when a `Loan` is created, data macros can be used to automatically update the `InventoryItem` to show it has been checked out. Similarly, when the `Loan` is updated to show it has been checked back in, the `InventoryItem` is updated as well.

In earlier versions of Access, this type of logic had to be implemented with VBA code as part of the UI implementation. You'll see as you read through this chapter that this logic fits more naturally in the data layer. This also frees up the UI implementation to focus on the user experience.

In this chapter, I'll show you how to use data macros to implement business rules and maintain data integrity as a natural extension of data modeling.

Understanding Data Macros

A data macro is logic that you can execute when certain data events are raised. Data macros are similar to triggers in SQL Server. The events that are supported are shown in Figure 3-1. Each data macro is associated with a specific table. To add or view a macro, you'll need to first open the desired table (in Datasheet View) and select the Table tab in the ribbon.

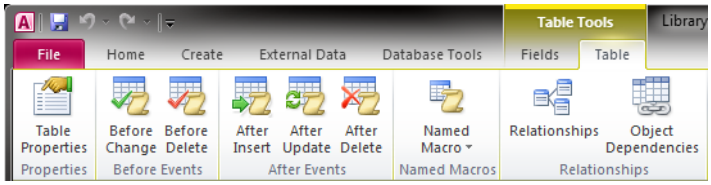


***Figure 3-1**. Data macro events*

These events are grouped into two categories: *before events* and *after events*. As you might expect, before events are raised before the record is updated (or inserted or deleted) and after events are raised after the change has been made. There are two before events: Before Change and Before Delete. The Before Change event is raised both before an insert and before an update. In your macro, you can use the `IsInsert` property to determine which action generated the event. In contrast, a separate after event is raised for each possible action (insert, update, or delete).

www.it-ebooks.info

Identifying Data Macro Limitations

Apart from this, there are some fundamental differences in before and after events and what you can do with them. The following are the important limitations to keep in mind:

- Before events can only modify the data in the current record; you cannot modify other records in this (or other tables).
- After events can update other records and other tables, but they cannot update data in the current record.
- Before events can raise an error, which will cancel the initial change that triggered the macro.
- After events can raise an error, but this will not halt the update; it has already been committed.
- After events can log records to the application log table, but before events cannot, since that is another table.
- After events can call a named macro, but before events cannot.

Creating Your First Data Macro

I think this will start to become clearer as you implement a macro. For the first macro, you'll update the status of an `InventoryItem` record when it has been loaned out. You'll use the After Insert event on the `Loan` table. When a `Loan` record is created, the macro will update the associated `InventoryItem` record, changing its Status value to Checked Out.

1. Open the `Loan` table using the Datasheet View, select the Table tab, and click the After Insert button. This will display the Macro Editor. A blank macro is displayed with a dropdown list that you'll use to add an action to the macro, as shown in Figure 3-2.

www.it-ebooks.info

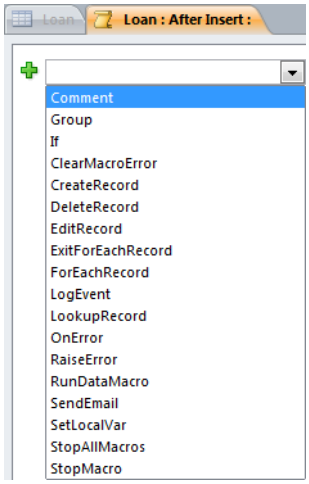


Figure 3-2. *The available actions*

2. Select the `Comment` action and a Textbox will appear. Enter the following text:

```
Update the associated InventoryItem record to show it has been checked out
```

3. Then, in the dropdown list, select the `LookupRecord` action, which loads a record based on the where clause that you'll provide. The editor will expand this action, showing the parameters that need to be supplied, as shown in Figure 3-3.

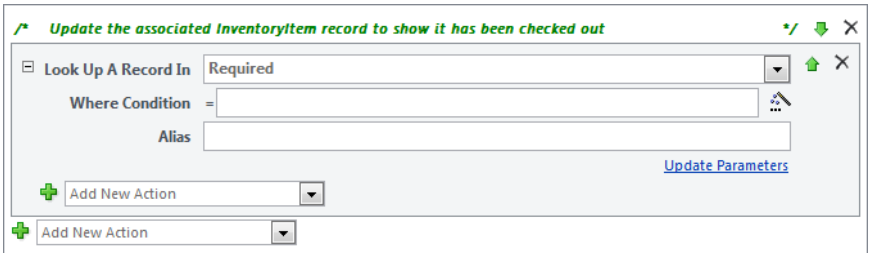


Figure 3-3. *The `LookupRecord` action parameters*

4. For the Look Up A Record In parameter, select the `InventoryItem` table.

■ **Tip** Most of the fields in the Macro Editor support IntelliSense. Just start typing and the appropriate choices will be displayed for you to select.

---

www.it-ebooks.info

5. For the Where Condition, enter

```
[InventoryItem].[InventoryItemID]=[Loan].[InventoryItemID]
```

6. The Alias will default to `InventoryItem`; you can leave the default value, for now. (I will explain aliases later.) Notice that there are now two Add New Action dropdown lists. One is inside the `LookupRecord` block and the other is outside the block. In the first one (inside the block), select the `EditRecord` action, which will allow you to make changes to this record. This action will be expanded as shown in Figure 3-4.



Figure 3-4. *The `EditRecord` action*

7. Leave the Alias field blank. Notice that there are now three Add New Action dropdown lists. The Macro Editor does a pretty good job keeping this straight by indenting appropriately and highlighting the selected block. For the one inside the `EditRecord` block, select the `SetField` action, which modifies a single field of the selected record. The expanded action is shown in Figure 3-5.



Figure 3-5. *The `SetField` action*

8. For the Name property, enter **InventoryItem.Status** and for the Value property, enter **Checked Out.** The completed macro is shown in Figure 3-6.

www.it-ebooks.info

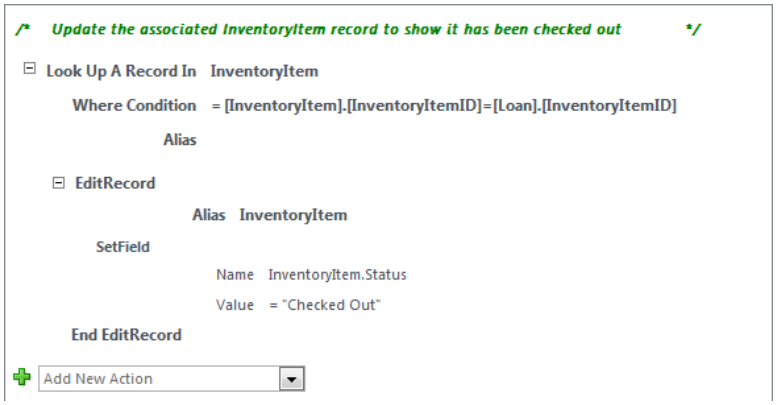


Figure 3-6. The completed macro

9. Click the Save button to save your changes and then click the Close button to close the Macro Editor.

Notice that the After Insert button in the ribbon is highlighted, which indicates that there is a macro defined for this event as demonstrated in Figure 3-7.

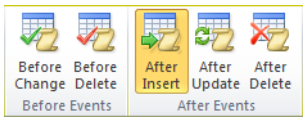
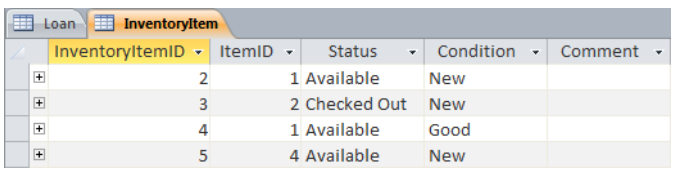


Figure 3-7. The highlighted After Insert button

Testing Your Macro

Now let's try it out. Enter a new record in the `Loan` table. Just select a customer and inventory item from the dropdown lists and save the record. Then go to the `InventoryItem` table. You should see that the one you selected in the `Loan` record now has a status of Checked Out as shown in Figure 3-8.



InventoryItemID	ItemID	Status	Condition	Comment
2	1	Available	New	
3	2	Checked Out	New	
4	1	Available	Good	
5	4	Available	New	

Figure 3-8. The `InventoryItem` record is checked out

www.it-ebooks.info

Exploring the Macro Editor

You'll be creating several more data macros, but before you do that, I want to explain some of the features of the Macro Editor. As you've noticed, you don't write code; instead, you design a macro by adding actions and filling in the appropriate parameters. The Action Catalog, shown in Figure 3-9, provides a good overview of the available actions.

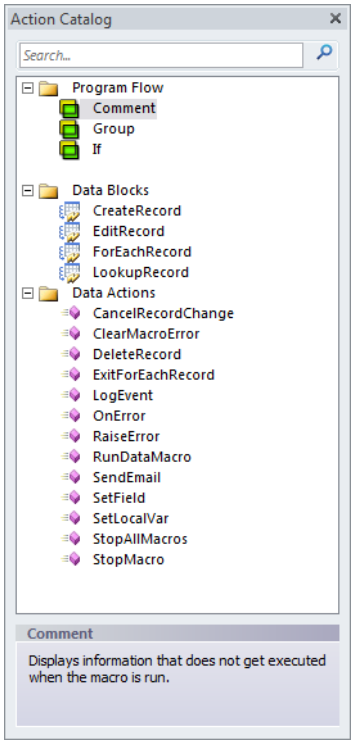


Figure 3-9. *The Action Catalog*

There are three types of actions: Program Flow, Data Blocks, and Data Actions. The `Comment` action, which you have already used, simply adds a line of text to explain something about the macro. You'll use the `Group` and `If` actions later. The `Group` action allows you put a set of actions in a group, which can then be collapsed. This is roughly equivalent to the `#region` blocks in .NET code. The `If` action lets you add conditional logic in your macro.

Understanding Data Blocks

Accessing and updating data is always done inside a data block. When you add a data block action to a macro, the editor indents the block and highlights that area of the macro. It does this to help you keep track of the scope of that block.

www.it-ebooks.info

For example, you used an EditRecord action, which is a data block action. You then added a SetField action to update a specific field of that record. The EditRecord action defines the record that is to be updated. The SetField action has to be inside the scope of the EditRecord action. If you were to place them outside of that scope, the Macro Editor would not know which record to update.

The LookupRecord action works the same way. It looks up a single record. You can only access fields from that record while inside the scope of that data block. Likewise, you can only edit that record while inside that data block. That's why the EditRecord action is inside the scope of the LookupRecord action.

There are two other data block actions. The CreateRecord is used when you want to insert a new record. You add SetField actions inside this data block to specify the field values. The ForEachRecord action works like the LookupRecord action, except that it allows for multiple rows to be returned. You can then process each one in a for-next loop.

Using Aliases

An alias is used to name and then reference a data block. For example, the LookupRecord action was assigned an alias. The EditRecord action uses the same alias so the Macro Editor knows to update the record that was just returned by the LookupRecord action.

Using Default Aliases

The Macro Editor does its best to avoid the need for you to deal with aliases. The default alias that the macro editor assigns is the table name. In most cases you can ignore the Alias field, as you did in the macro that you just implemented. If you need to work with two records from the same table, for example, the default alias would be ambiguous. In this case, you'll need to change the default alias for at least one the records. For those times that you need to specify an alias, you should understand how they work. First, I'll explain how the default aliases work.

In a data macro, the macro keeps track of the records that are being used. These records are referred to as *data contexts*. Initially, a macro is executed on behalf of a record that is about to be or has been modified. A data context is added to the macro for this record, and it's given an alias using the table name. In your macro this was the Loan table, and you access its fields by putting "Loan." in front of the field name.

Whenever you add one of the data block actions (CreateRecord, LookupRecord, or ForEachRecord – EditRecord is a special case, which I'll explain later), a new data context is added, and its default alias is determined by the table name. The LookupRecord action added a new data context with the InventoryItem alias. So inside the LookupRecord data block, there are now two data contexts, and their aliases are Loan and InventoryItem.

The default data context is the last one to be added to the macro. Think of this like a stack. Initially the Loan data context was added. It's on the top of the stack and is the default data context. When the LookupRecord action is executed, it adds the InventoryItem data context on top of the Loan data context. Inside the LookupRecord data block, the InventoryItem data context is the default context; it's the last one that was added. Once the macro exits this data block, the InventoryItem data context is popped off the stack and the Loan data context is now the default.

The Macro Editor will show the default in the Alias field, which is based on the name of the table. If you don't change it, when you tab off that field, the Macro Editor changes it to an empty field. This signifies that the default alias is being used.

www.it-ebooks.info

Using the EditRecord Action

Unlike the other data block actions, the `EditRecord` action does not create a data context. Rather, it must be assigned one to use, which is done by entering the appropriate alias. If no alias is supplied, it will use the default data context. Since we're inside the `LookupRecord` data block, the `InventoryItem` data context is used. If you were to add the `EditRecord` outside of the `LookupRecord` data block, the default data context would be the initial `Loan` record.

As you might have already guessed, using default alias and data contexts can leave room for error. It is also a little more difficult for someone else to understand how the macro is supposed to work. For these reasons, I recommend that you always explicitly supply an alias. You can use the default name, when appropriate, but you'll need to enter it in the field so the editor doesn't display a blank value. Figure 3-10 shows your first macro with explicit aliases.

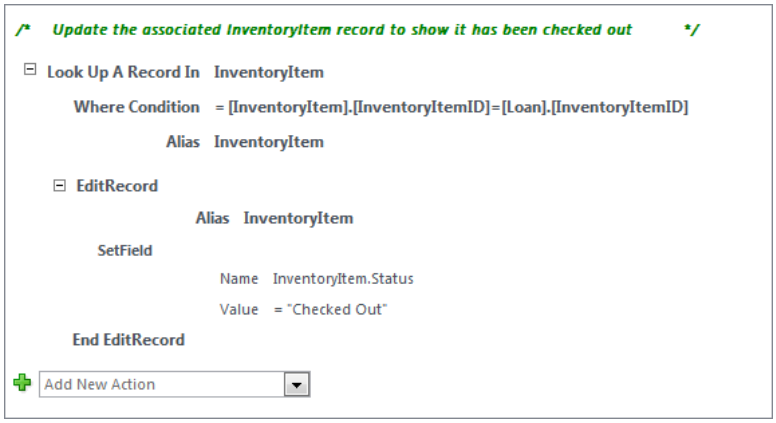


Figure 3-10. *The macro updated with explicit alias references*

When to Use an Alias

There are times when you need to define an alias instead of using the default alias. For example, if you're writing a macro for the `Loan` table. The default data context is given the alias `Loan`. Now suppose you wanted to look up the loan for the last time this item was checked out. You would add a `LookupRecord` action and the appropriate expression for the Where Condition. You would now have two records, both of which are from the `Loan` table. To avoid ambiguity, you would need to specify a different alias, such as `PriorLoan`. In subsequent expressions, you would specify `Loan` for the current record and `PriorLoan` for the previous loan.

Using Data Actions

The last section of the Action Catalog lists the available data actions. I will demonstrate many of these in the remainder of this chapter. Some of these are only available in certain situations. For example, `SetField` can only be used inside of an `EditRecord` or `CreateRecord` data block. The Add New Action dropdown list in the Macro Editor will list the subset of actions that are allowed based on the context.

www.it-ebooks.info

Navigating the Macro Editor

After you have added actions to your macro, the Macro Editor allows you rearrange them. Figure 3-11 shows these editing controls.

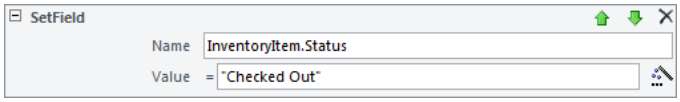


Figure 3-11*. The Macro Editor controls*

When you select an action, there is a minus "-" button at the top left, next to the action name. Collapsing actions can make it easier to read your macro, especially with more complex macros. If the action is already collapsed, there with be a plus "+" button instead, which will expand the action.

At the top-right of the action there are green up and down arrows. Use these to change the order of the actions. If you select a data block action such as `LookupRecord`, the entire block will be moved. There is also an "X" button at the far top-right; use this to remove the action.

To add a new action, use one of the Add New Action dropdown lists. These are only available in a few locations. Generally there is one for each indention level. After adding the action, use the green arrows to move it to the correct location.

Implementing the Loan Before Change Event

The before events (Before Change and Before Delete) are ideally suited for adding validation prior to making a change. You can also use them to update the fields of the record being added or changed. Before a `Loan` record is added or modified, there are several validations that should be performed. I will explain these, one section at a time, and then show you the resulting macro actions that you can add into your database.

Making Sure the Item is Available

When a `Loan` record is created, the user specifies the inventory item that is being checked out and the customer that is taking it. You'll add logic to the Before Change event to ensure that this item is actually available to be checked out. If it is on hold, the logic will also ensure that this is the customer that reserved it.

This code first checks to see if this is an insert; we don't need to perform this check on an update. It then looks up the selected `InventoryItem` record. If the `Status` is not Available or On Hold, it raises an error and stops the macro. These two actions are included in a `Group` action, so the group can be collapsed for readability.

If the `Status` is On Hold, it looks up the `Request` record that is associated with this inventory item. If the requesting customer is not the same as the one checking out the item, the macro raises an error. When an error is raised on a before event, the error message is displayed in a pop-up window and the update is aborted. To add this Data Macro, perform the following steps:

www.it-ebooks.info

1. Open the Loan table in the Datasheet View.
2. From the Table tab, click the Before Change button. This will display a blank macro.
3. Figure 3-12 shows how this logic should be implemented. Enter these actions into the Macro Editor.

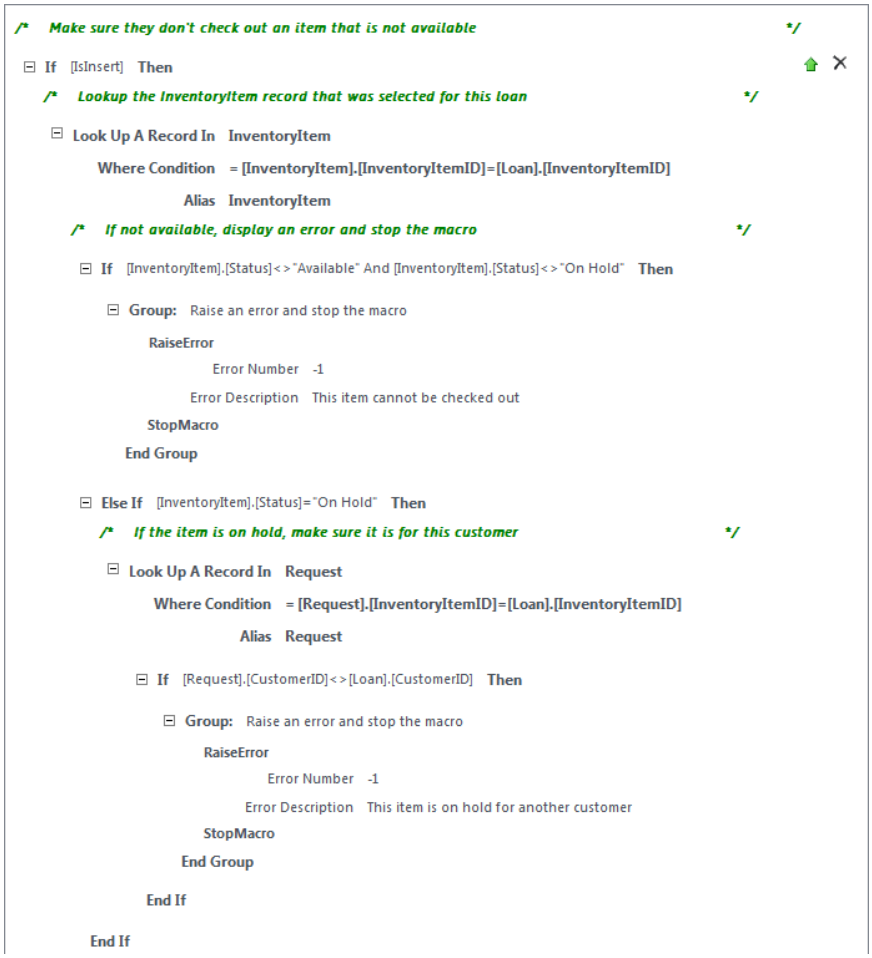


Figure 3-12. *Implementing the availability logic*

www.it-ebooks.info

Calculating the Due Date

Recall from the table design in Chapter 2 that the loan period is defined on the `Media` table. To determine the due date of a new `Loan` record, we'll need to look up the `LoanPeriod` from the `Media` table and add it to the current date. We will only perform this logic when this is an insert and when the `DueDate` field is not already specified.

To accomplish this, we'll need to perform a series of nested lookups. We'll first find the `InventoryItem` record and then the associated `Item` record, and then we can get the `Media` record. The `DueDate` field is then updated using a `SetField` action. The `FormatDateTime()` function is used to strip off the time portion. Also, note that an `EditRecord` action is not needed in a before event, because only the current record can be updated.

Add the actions shown in Figure 3-13 to the Before Change macro. Add these to the very end of the macro using the last Add New Action list box.

```
/*   When inserting a new loan, set the due date if not already specified   */
If  [IsInsert] And IsNull([DueDate])  Then
   /*   Lookup the InventoryItem record that was selected for this loan   */
   Look Up A Record In  InventoryItem
         Where Condition  = [InventoryItem].[InventoryItemID]=[Loan].[InventoryItemID]
                  Alias  InventoryItem
      /*   Lookup the Item for this InventoryItem   */
      Look Up A Record In  Item
            Where Condition  = [Item].[ItemID]=[InventoryItem].[ItemID]
                     Alias  Item
         /*   Lookup the Media record that defines the rules for this item   */
         Look Up A Record In  Media
               Where Condition  = [Media].[MediaID]=[Item].[MediaID]
                        Alias  Media
            /*   Calculate the due date by adding the loan period to the current date
                 Make sure to strip off the time portion   */
            SetField
                           Name  Loan.DueDate
                           Value  = FormatDateTime(Now()+[Media].[LoanPeriod],2)

End If
```

Figure 3-13. *Implementing the* `DueDate` *logic*

Calculating the Late Fee

When an item is being checked back in, the `CheckedIn` field on the `Loan` record is set to the current date/time. You can tell if a particular field is being changed by using the `Updated()` function. If the `CheckedIn` field is being modified and it is not null, then we can infer that this item is being checked in.

When an item is checked in, we need to see if it is overdue. You can check that by comparing the current date/time with the `DueDate` on the `Loan` record.

www.it-ebooks.info

■ **Note** You'll need to add a day to the due date to account for the time portion. For example, if the item is due on Jan 12 and it is currently noon on Jan 12, the due date, because it has no time portion, will be before the current date/time. By adding a day to the due date, we're allowing them to return the item up until midnight of the 12th.

This logic computes the number of days the item is late and then looks up the daily overdue fee from the `Media` table. Just as with the `DueDate` logic, this will require several nested `LookupRecord` actions.

Figure 3-14 shows the actions that are needed to perform this function. Add these to your current macro design.

```
/*  Calculate the late fee if the item is overdue                    */
If  Updated("CheckedIn") And Not IsNull([CheckedIn])  Then
    If  [DueDate]+1<Now()  Then
        /*  Compute the number of days overdue                       */
        SetLocalVar
                    Name  daysOverdue
              Expression  = Date()-[DueDate]
        /*  Get the daily late fee, which is defined in the Media table  */
        Look Up A Record In  InventoryItem
            Where Condition  = [InventoryItem].[InventoryItemID]=[Loan].[InventoryItemID]
                      Alias  InventoryItem
            Look Up A Record In  Item
                Where Condition  = [Item].[ItemID]=[InventoryItem].[ItemID]
                          Alias  Item
                Look Up A Record In  Media
                    Where Condition  = [Media].[MediaID]=[Item].[MediaID]
                              Alias  Media
                    /*  Calculate the overdue fee                    */
                    SetField
                                Name  Loan.OverdueFee
                               Value  = [daysOverdue]*[Media].[OverdueFee]

    End If
End If
```

Figure 3-14. *Implement the* `OverdueFee` *logic*

www.it-ebooks.info

Validating Renewals

If the `DueDate` is being changed, you can infer that the loan has been renewed. In this case, you'll need to see if renewals are allowed and if this will exceed the number of allowed renewals. If this renewal is not allowed, the macro should raise an error, which will abort the update. If it can be renewed, then the macro should increment the renewal count on the `Loan` record.

To do this, you'll need to use nested `LookupRecord` actions to get the `Media` record that determines the renewal policy. If this renewal is not allowed, it will call the `RaiseError` action and stop the macro. Otherwise, the `Renewals` count is updated.

Figure 3-15 shows the necessary actions; add these to your macro design.

```
/*   If updating the DueDate, increment the renewals                    */
⊟ If   Not [IsInsert] And Updated("DueDate")   Then
   /*   Lookup the number of renewals allowed                            */
   ⊟ Look Up A Record In   InventoryItem
          Where Condition   = [InventoryItem].[InventoryItemID]=[Loan].[InventoryItemID]
                    Alias   InventoryItem
      ⊟ Look Up A Record In   Item
            Where Condition   = [Item].[ItemID]=[InventoryItem].[ItemID]
                      Alias   Item
         ⊟ Look Up A Record In   Media
               Where Condition   = [Media].[MediaID]=[Item].[MediaID]
                         Alias   Media
            /*   See if we are allowed to renew this loan               */
            ⊟ If   [Loan].[Renewals]>=[Media].[RenewalsAllowed]   Then
               ⊟ Group:   Raise an error and stop the mac...
                     RaiseError
                           Error Number   -1
                     Error Description   This item cannot be renewed
                     StopMacro
                  End Group
               /*   Increment the number of renewals                    */
            ⊟ Else
                  SetField
                              Name   Loan.Renewals
                              Value   = [Loan].[Renewals]+1
               End If

End If
```

Figure 3-15. *Implementing the renewal policy*

www.it-ebooks.info

Adding the Current Loan Reference

The Loan table has a reference to the InventoryItem table to indicate the specific copy that is being loaned out. It would be helpful to also have a reference in the other direction so each InventoryItem record will store the current Loan record, if it is currently checked out. You'll add the reference now and then use data macros to set this value automatically.

■ **Note** This is an example of denormalization. The current loan for an inventory item can be determined by looking up all loans for that item and returning the last one. Therefore, this is redundant information. We will add it anyway, to optimize performance, but use data macros to minimize the negative consequences.

Adding the Lookup Field

Open the InventoryItem table using the Design View. Add a new field named **CurrentLoanID** and select the Lookup Wizard Data Type. This will start the Lookup Wizard that will guide you through setting up this field. You can refer to Chapter 2 for more details about this process.

1. In the first dialog box, select the first option, which is to look up the values from another table.
2. In the second dialog box, select the Loan table.
3. In the third dialog box, select only the LoanID field.
4. In the fourth dialog box, sort by the LoanID field.
5. Accept the default values for the fifth dialog box.
6. In the sixth dialog box, select the Enable Data Integrity check box and choose the Restrict Delete option.

After the CurrentLoanID field has been created, from the Design View of the InventoryItem table, select the CurrentLoanID field and the Lookup tab of the Field Properties. Make sure the Limit To List property is set to Yes.

Modifying the Loan After Insert Event

Now you'll need to make a minor change to the After Insert event on the Loan table. This is the macro that you created at the beginning of this chapter. You've added a new field to the InventoryItem table and now you'll update the macro to also set this field.

1. Open the Loan table in the Datasheet View. From the Table tab, click the After Insert button. This will display the existing macro logic.
2. Add another SetField action within the existing EditRecord data block. To do that, click on the EditRecord action, which will add an Add New Action dropdown list at the end of the EditRecord data block.

www.it-ebooks.info

3. Select the `SetField` action. For the Name property, enter **InventoryItem.CurrentLoanID**, and for the Value property enter [**Loan**].[**LoanID**]. The updated macro should look like Figure 3-16.

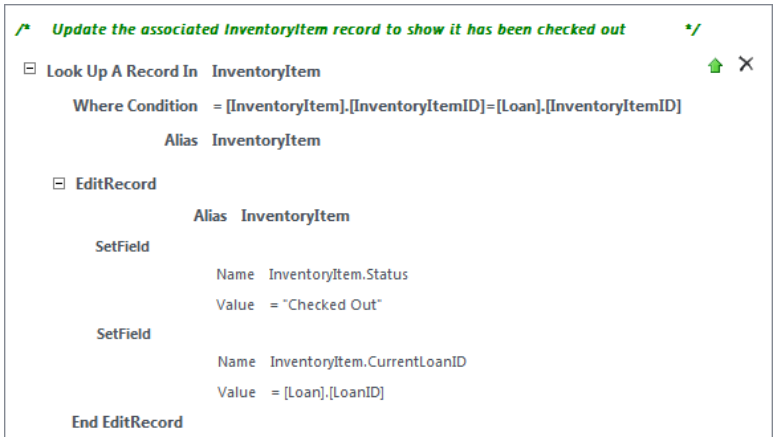


***Figure 3-16**. The updated After Insert macro*

While you're editing this macro, there's another section that you'll need to add. When you check out an item that was on hold, the associated `Request` record should be updated and marked complete. Figure 3-17 shows the actions that you'll need to add to the After Insert event. Add these at the end of this macro.



***Figure 3-17**. Updating the associated `Request` record*

www.it-ebooks.info

Modifying the Loan After Update Event

When an item is checked back in, you'll need to update the associated `InventoryItem` record to show that it is now available and to clear the `CurrentLoanID` field. Save the After Insert event and close the Macro Editor. Then click the After Update button, which will display a blank macro.

When designing an After Update macro, you can use the `Old` property, which is an alias to the copy of the record before the changes were. To determine if an item is being checked in, you'll see if the old value of the `CheckedIn` property is null and the current value is not null. If it is being checked in, then edit the associated `InventoryItem` record. Figure 3-18 shows the actions that you'll need to add to do this.

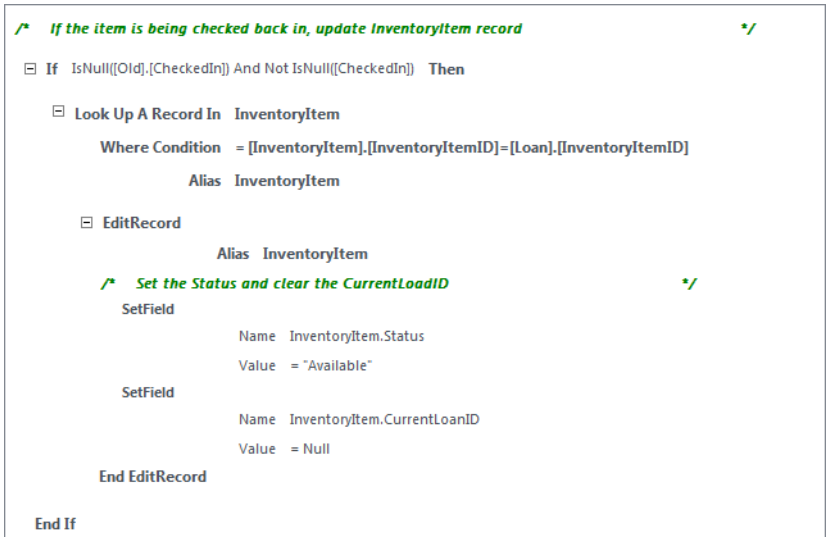


Figure 3-18. *Implementing the Loan After Update event*

Handling Requested Items

When a customer makes a request to reserve an item, a `Request` record is created. This references an `Item`, not an `InventoryItem`. Whichever copy becomes available first, that `InventoryItem` will be the one that is reserved. Every time an `InventoryItem` becomes available, you'll need to check to see if there is an outstanding request for that item.

When a pending request is found, the `Request` record needs to be updated to change its `Status` to Ready and to reference the `InventoryItem` that is being reserved. The `InventoryItem` record also needs to be updated to change its `Status` to On Hold. Because of the limitations that I outlined at the beginning of this chapter, you'll need to implement this in two parts. The triggering event will be on the `InventoryItem` record (when its status changes to Available). The `InventoryItem` record must be updated in the Before Change event. However, the `Request` record must be updated in the after events.

www.it-ebooks.info

Implementing the Before Change Event

Let's start with the Before Change event.

1. Open the `InventoryItem` table using the Datasheet View.
2. From the Table tab, click the Before Change button, which will display a blank macro.
3. Use an `If` action to see if the `Status` is being changed and the new value is Available. If this is true, then see if there is a pending `Request` record for this item. If there is, use a `SetField` action to change the `Status` to On Hold. The completed macro is shown in Figure 3-19.

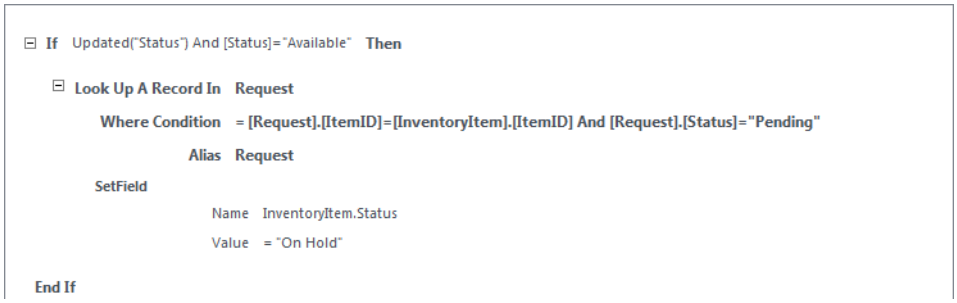


Figure 3-19*. The Before Change event*

Now you'll need to implement the after events to update the `Request` record. There are two ways that an inventory item can become available:

- A loaned inventory item is checked in.
- A new inventory item is added; that is, a new copy is received into inventory.

The `InventoryItem` After Update event will catch the first scenario and the After Insert event will catch the second. You will perform the identical actions in both cases.

Creating a Named Data Macro

The best way to implement this is to create a named macro that is called from both events. Named macros still need to be associated to a table. Open the `InventoryItem` table using the Datasheet View. From the Table tab, click the Named Macro button and then click the `Create Named Macro` link, as shown in Figure 3-20.

www.it-ebooks.info

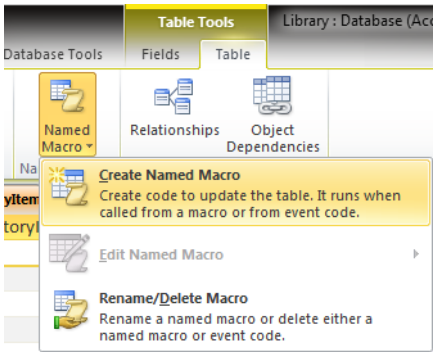


Figure 3-20. Creating a named macro

This macro needs to look for a `Request` record for the current item (the one becoming available) and is in Pending status. Because this query could return multiple records, you'll use `ForEachRecord` action. After the first record is processed, it will call the `ExitForEachRecord` action to exit the for-next loop.

Figure 3-21 shows the actions needed to implement this. Add these actions to your named data macro.

```
/*  See if there are any pending requests for this item                    */
For Each Record In  Request
    Where Condition  = [Request].[ItemID]=[InventoryItem].[ItemID] And [Request].[Status]="Pending"
              Alias  Request
    EditRecord
              Alias  Request
        SetField
                 Name  Request.InventoryItemID
                 Value  = [InventoryItem].[InventoryItemID]
        SetField
                 Name  Request.Status
                 Value  = "Ready"
    End EditRecord
/*  We can only assign one request so exit the loop after the first one    */
    ExitForEachRecord
```

Figure 3-21. Implementing a named macro

Named macros support parameters so you can pass information to them. However, when a named macro is called from a data event such as After Update, the data contexts from the initiating event are carried into the named macro. So this macro can access the `InventoryItem` data context because it was in the initiating event.

Click the Save button to save this macro. When prompted, enter the name **AssignPendingRequest** as shown in Figure 3-22.

www.it-ebooks.info

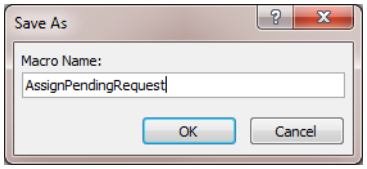


Figure 3-22. *Entering the macro name*

Calling a Named Macro

Now you'll need to implement the After Insert and After Update events.

1. Click the After Insert button, which will create a blank macro.
2. In the Add New Action dropdown list select the `If` action. For the condition enter **[InventoryItem].[Status] = "On Hold."** For the action select `RunDataMacro` from the dropdown list.
3. The Macro Name property is a dropdown list that shows all of the existing named macros. There should only be one; select `InventoryItem.AssignPendingRequest`. The macro should look like Figure 3-23.

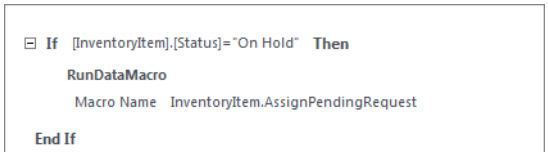


Figure 3-23. *The `InventoryItem` After Insert event*

Save this macro and close the Macro Editor. Then click the After Update button to define this event. The macro for this event should be identical to the After Insert event.

Computing Overdue Fees

You implemented logic in the Before Change event to compute the `OverdueFee` when an item is checked in (if it is overdue). However, for items that are still checked out, you will want to re-compute the current late fee on a daily basis. This might be part of a daily process that sends out reminder emails. To do this, you'll implement a named data macro, which will then be called from a user-executed macro.

Creating a Named Data Macro to Compute Overdue Fees

Let's get started.

1. Open the `Loan` table using the Datasheet View.
2. From the Table tab, create a named macro just like you did earlier. This will display a blank macro. Since this macro will be called from the UI there is no default data context.

www.it-ebooks.info

3. The first thing that this macro must do is to execute a data block action. In this case you'll use the `ForEachRecord` action to find all `Loan` records that are overdue. For each record, you'll compute the number of days that it is overdue. Then look up the daily fee in the `Media` table. Finally, use the `EditRecord` action to update the `Loan` record. The complete macro is shown in Figure 3-24. Enter these actions in your macro.



Figure 3-24. *Calculating the* `OverdueFee` *for all loans*

4. Save the macro and enter the name **CalculateAllLateFees** when prompted, as shown in Figure 3-25.

www.it-ebooks.info

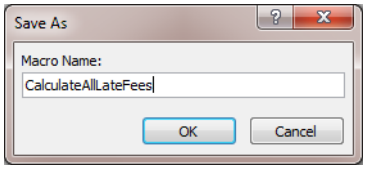


Figure 3-25. *Saving the* `CalculateAllLateFees` *macro*

Creating a User-Executed Macro

To call the `CalculateAllLateFees` data macro, you'll need to create a macro (not a data macro).

1. From the Create tab of the ribbon, click the Macro button. This will create a blank macro. Notice that it uses the same Macro Editor, except there are different actions available to you.
2. In the Add New Action dropdown list, select the `RunDataMacro` action. Then select the `CalculateAllLateFees` macro from the dropdown list. The complete macro is shown in Figure 3-26.

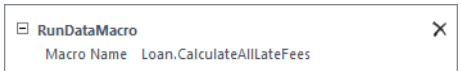


Figure 3-26. *Calling the* `CalculateAlllateFees` *data macro*

3. Click the Save icon at the top of the window and enter the name **CalculateLateFees** when prompted. You should now have a Macro listed in the Object Navigation pane.
4. Right-click the macro and select the Run link to execute this macro.

Debugging Data Macros

If a data macro gets an error, if will be logged to the Application Log table. You can access this table from the File page. If not selected, click the Info tab. You should have a link on this page to view the Application Log as shown in Figure 3-27.

www.it-ebooks.info

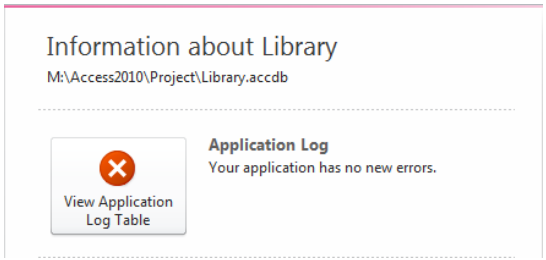


Figure 3-27. *The File Info page*

This link is only shown on this page if there is anything to report. If this page does not include this link, then there is nothing in the Application Log table. We'll take care of that now.

Using the LogEvent Action

You can also log your own messages to the log using the `LogEvent` action.

1. Open the `Loan` record using the Datasheet View.
2. From the Table tab, click the Named Macro button and then click the Edit Named Macro and CalculateAllLateFees links, as shown in Figure 3-28.

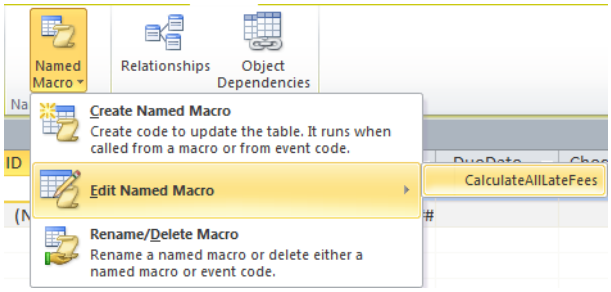


Figure 3-28. *Editing the `CalculateAllLateFees` macro*

3. This will display the existing macro. At the end of the macro, in the Add New Action dropdown list, select the `LogEvent` action. In the Description field enter **The CalculateAllLateFees macro has completed**. This action will look like Figure 3-29. Save and close this macro.

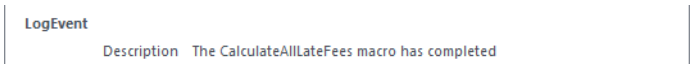


Figure 3-29. *The `LogEvent` action*

www.it-ebooks.info

4. Now re-run this data macro by running the `CalculateLateFees` macro from the Object Navigation pane.

Viewing the Application Log

You should now have a link to the Application Log on the File Info page. Click this link to view the contents of this log. You should see an entry in the log similar to the one shown in Figure 3-30.

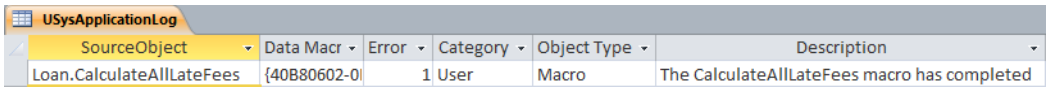


Figure 3-30. *A sample* `Application Log` *entry*

■ **Tip** If there are new errors in the `Application Log` table, the link on the File Info page will have a red background.

Testing the Application

Try out your application and make sure the macros are working as expected. To test the basic scenarios try the following:

- Insert a `Loan` record and verify that the `Status` of the selected `InventoryItem` has been changed to Checked Out and references the new `Loan` record. Verify the `DueDate` on the `Loan` record has been calculated. Set the `CheckedIn` date on the `Loan` record and verify the `Status` of the `InventoryItem` is now Available.
- Create another `Loan` record. Then edit the `DueDate` field making it several days in the past. Run the `CalculateLateFees` macro and verify the `OverdueFee` has been calculated.
- Create another `Loan` record and change its `DueDate` to be in the past. Then set the `CheckedIn` date and verify the `OverdueFee` was calculated.
- Create a `Request` record. Then insert a new `InventoryItem` record that references the same `Item` as the `Request` record. Verify the `Status` of the `InventoryItem` is now On Hold and the `Status` of the `Request` record is Ready.

Summary

So far, by just creating the data schema in Chapter 2 and implementing data macros in this chapter, you have a working application. The macros implement many of the business rules such as computing due dates and overdue fees. You also handle requested items; automatically reserving an inventory item when one becomes available.

www.it-ebooks.info

In this chapter you have learned about:

- The uses and limitation of before and after data events
- Using the Macro Editor
- How data contexts and alias work in a data macro
- Creating named macros
- Calling a data macro from the UI
- Using the `Application Log` to debug data macros

In the next chapter, you'll design both select and action queries.

www.it-ebooks.info

CHAPTER 4

Designing Queries

In this chapter, you'll design queries that will be included with your `Library` application. Throughout this project, you'll be creating a lot of queries. This chapter will introduce the basic concepts that you'll need to understand when creating and using queries in Access 2010. You'll create a few sample queries in this chapter, and then create the remainder in subsequent chapters as necessary.

There are two basic types of queries: *select queries* return data from one or more tables and *action queries* are used to insert, update, or delete records from a table. Access 2010 provides a Query Wizard that will guide you through the process of creating a query. Later I will also show you how to create and modify a query using the Design View.

Creating Select Queries

A select query is used to return a subset of data. For example, if you wanted to see all the `InventoryItem` records that are currently checked out. You could do this by creating a query that provides all the same columns as the `InventoryItem` table, and then adding a filter to only return the records with a `Status` of "Checked Out." Let's do that now.

Creating a Simple Query

From the Create tab in the ribbon, click the Query Wizard button, which will display the New Query dialog box, as shown in Figure 4-1.

www.it-ebooks.info

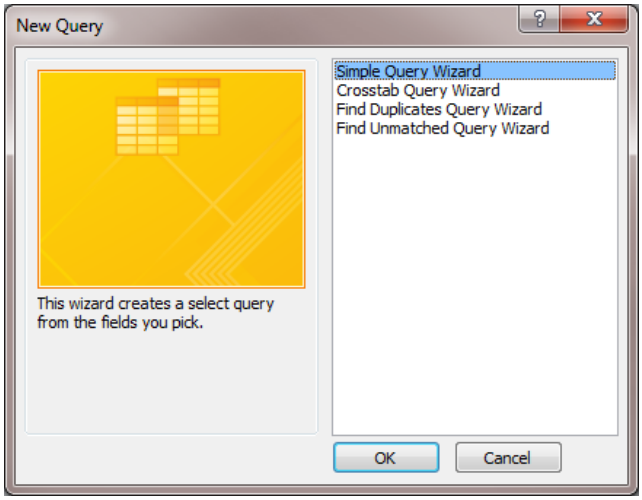


Figure 4-1. *The Query Wizard*

Select the Simple Query Wizard option and click the OK button to display the next dialog box. In the Tables/Queries dropdown list, select the `InventoryItem` table. Then click the ">>" button to add all the fields, as shown in Figure 4-2.

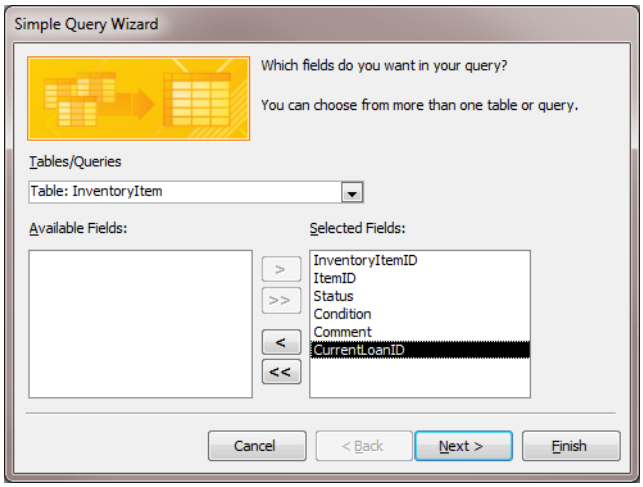


Figure 4-2. *Selecting the fields for your query*

Click the Next button to display the final dialog box. Enter the name **CheckedOutItems**, as shown in Figure 4-3. Select the second radio button, which will open the query using the Design View. Click the Finish button to create the query.

www.it-ebooks.info

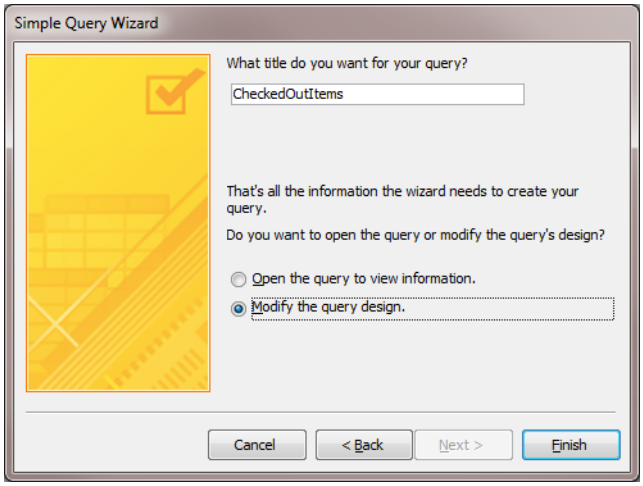


Figure 4-3. *Entering the name of the query*

The Query Wizard will then create the query and display it using the Design View. At this point, the query looks just like the `InventoryItem` table. It has the same set of columns and the query will return all the rows in the table. This is equivalent to a SQL statement like `SELECT * FROM InventoryItem`. You'll need to add a filter to the query to restrict the rows that are returned.

The query fields are listed in the bottom half of the Design View. In the Criteria row of the `Status` field, enter **"Checked Out"** as shown in Figure 4-4. This will cause the query to only include the records that have a value of "Checked Out" in the `Status` field. In SQL syntax this is adding the `WHERE Status = 'CheckedOut'` clause.

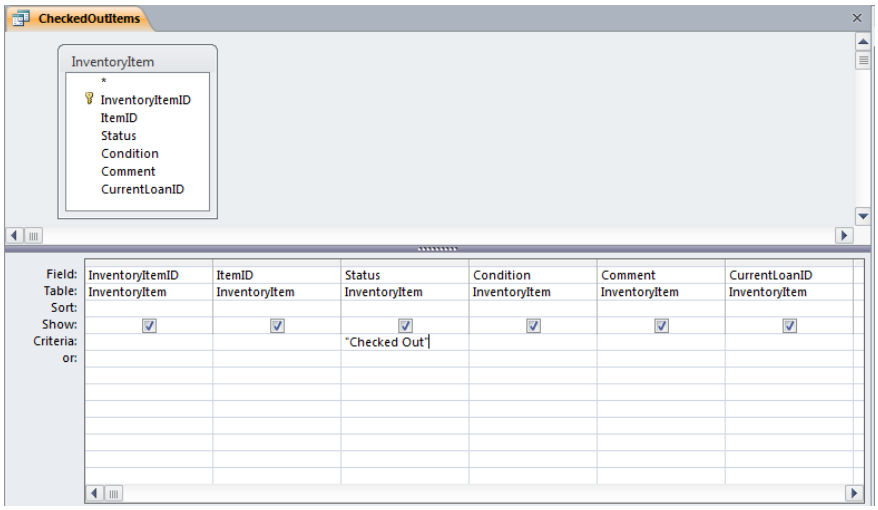


Figure 4-4. *Specifying the filter criteria*

www.it-ebooks.info

Save the query using the Save icon at the top of the application. To run the query, you can either click the Run button in the ribbon, or select the Datasheet View. The results of the query should look like Figure 4-5, depending on the data in your database.

CheckedOutItems

InventoryItemID	ItemID	Status	Condition	Comment	CurrentLoan
2	1	Checked Out	New		4
3	2	Checked Out	New		1
(New)		Available	New		

Figure 4-5. *Displaying the query results*

Adding Tables

Because the database design has been normalized (see Chapter 2), you'll often find that the contents of a single table are not very useful by themselves. The `CheckedOutItems` query is a good example. It lists the inventory items that are currently checked out, but there are several details that would be nice to have. You probably want to know when it was checked out, when it is due back, and who has it. You will also want a description of the item. All of this is available in related tables. You'll now modify the query to include other tables to get the desired additional fields.

Open the `CheckedOutItems` query using the Design View. To include additional tables to the query, click the Show Table button in the Design tab of the ribbon. This will display the Show Table dialog box shown in Figure 4-6.

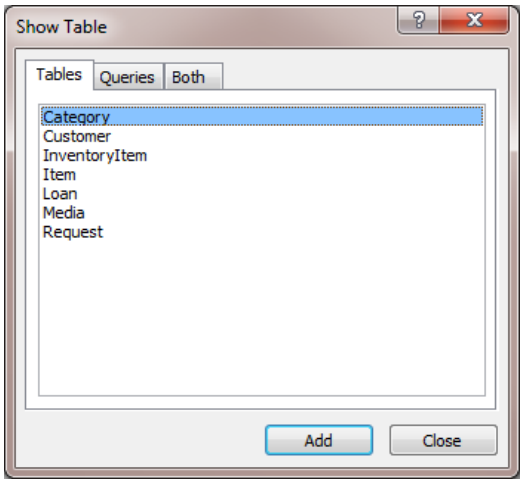


Figure 4-6. *The Show Table dialog box*

To add a table, select the desired table and click the Add button or simply double-click the table name. To add multiple tables, you can hold the Ctrl button down and select the tables you want to add and then click the Add button. Add the following tables to the query:

www.it-ebooks.info

- Customer
- Item
- Loan

Close the Show Table dialog box when you're done.

Using Joins

Notice that there are lines connecting the tables. Access automatically creates what are called *joins* in your query wherever there is a relationship between the tables. I will explain joins in more detail later.

■ **Tip** You can remove, add, or edit these joins in the Design View. This has no effect on the actual table relationships.

Rearrange the tables so the connecting lines are all visible, as shown in Figure 4-7.

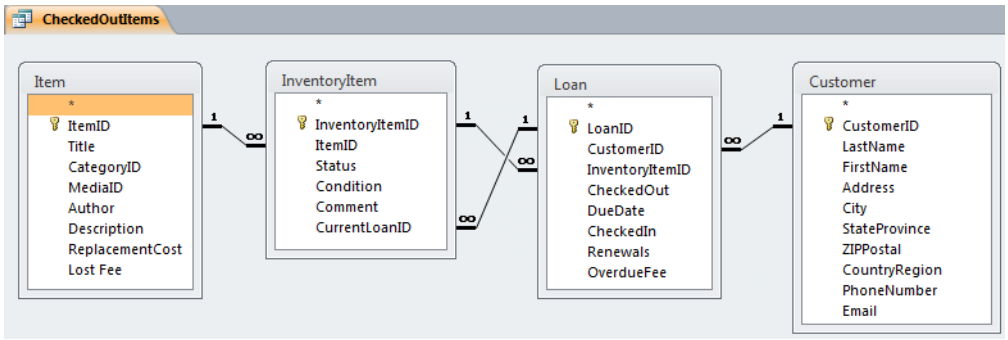


Figure 4-7. *The* `CheckedOutItems` *query with additional tables*

Notice that there are two lines between the `Loan` and `InventoryItem` tables. Both of these tables have a reference on the other table. The `Loan` table specifies the `InventoryItem` that has been checked out, while the `InventoryItem` specifies the current `Loan`. The first relationship is static; the `Loan` will always reference the same `InventoryItem`. The second is transient; each time the item is checked out, the `InventoryItem` will refer to a different `Loan`. It's very unlikely that you will ever want to use both joins in the same query. In this case, since `InventoryItem` is the main table that you're starting with, you'll use the `CurrentLoanID` field to get the associated `Loan`. Select the join that connects the `InventoryItemID` field on both tables and press the delete key.

www.it-ebooks.info

Understanding Multiplicity

Each line that connects two tables represents a join, and a *multiplicity* indicator is shown at each of its endpoints. The multiplicity specifies the number of instances that are allowed on each side of the relationship. In the join between the `Item` and the `InventoryItem` tables, for example, there is a "1" next to the `Item` table, and an infinity symbol (∞) next to the `InventoryItem` table. This indicates a one-to-many relationship (or many-to-one, depending on your perspective).

The "1" next to the `Item` table indicates that each `InventoryItem` record can refer to, at most, one `Item` record. The infinity symbol specifies that an `Item` can be related to an unlimited number of `InventoryItem` records (in other diagramming notations the letter "m" is used to indicate this). In Access, the multiplicity only indicates the maximum number. In other notations that you may have seen, the multiplicity indicates the possible values, which are usually 0, 1 or m (many). You might see "0,1" for example. This indicates there can be at most 1, but 0 is also allowed.

The multiplicity properties can help you analyze a query. When you join the `Item` table to the `InventoryItem` table, the multiplicity of 1 means that you will not increase the number of rows in the result. There can only be one `Item` for each `InventoryItem` record. However, if you started with the `Item` table, and then joined the `InventoryItem` table, the "many" multiplicity indicates that you could end up with more rows than you started with. You can see in this query that you will have one row for each `InventoryItem` record. By joining the additional tables, you will not increase the number of rows.

Using the Join Properties

Right-click the join between the `Item` and the `InventoryItem` table and click the *Join Properties* link. This will display the Join Properties dialog box shown in Figure 4-8.

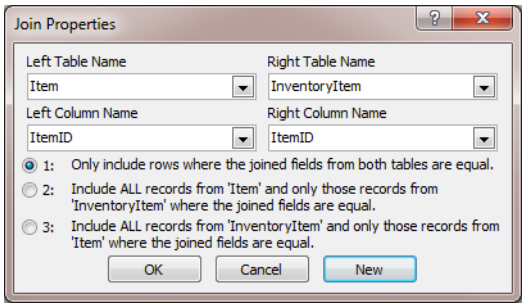


Figure 4-8. *The Join Properties dialog box*

When joining two tables, one table is referred to as the left table and the other is the right table. The left table is the one that is being added to the query. The table that you started with is called the right table. You can see from the Join Properties dialog box that `Item` is the left table and `InventoryItem` is the right table.

www.it-ebooks.info

■ **Note** Because queries in Access are normally displayed graphically, you might think that the left table is the one on the left side of the join in the Design View. In this case that happens to be true, but it is purely coincidental. If you look at the properties for the `Loan – InventoryItem` join, you'll see that `Loan` is the left table even though it is on the right side of the join.

When joining two tables, you'll need to specify the column(s) in both tables that are used to find matching records. In this join, the `ItemID` column on the `InventoryItem` table is used to find `Item` record(s) that have an `ItemID` with the same value. In Access, you can only use a single column in a join. Because we're using a single primary key field in all tables, that limitation does not usually cause a problem; it makes the design simpler and easier to follow. If you have a scenario where you need to match on multiple columns, you can add additional joins between the same tables.

In database queries, there are the following three types of joins:

- **Inner:** Returns only rows where there are matching records in both tables.
- **Outer:** Returns all records in one table and the matching records in the other
- **Full:** Returns records in both tables regardless of matches

The first option in the Join Properties dialog box indicates that an *inner* join should be used. For *outer* joins, you'll need to specify from which table all records should be returned from. A *left outer* join indicates that everything from the left table is returned, and only records from the right table where there is a match. This is the second option. The third option is the reverse of that, and is called a *right outer* join. Access doesn't support *full* joins.

Because the database is well constrained, it should not be possible to have an `InventoryItem` record that does not have a matching `Item` record. Any of the three options should give you the exact same results. However, I advise that you use outer joins in most cases. The purpose of this query is to return all items that are checked out. You're joining to additional tables to provide more details. Should the join fail for any reason, you still want to return the base record, `InventoryItem`, even if some of the details might be missing. You should choose the third option, which is a right outer join.

There are exceptions to this rule. Suppose, for example, the purpose of the query is to find customers that have items checked out. Should the join from `InventoryItem` to `Loan` or `Loan` to `Customer` fail, you could end up with an `InventoryItem` record with no `Customer` details. That would not be useful, so you choose option 1 or *inner* join to insure that you only get records that have a matching `Customer` record.

Edit the join properties for each join and select the third option. Notice that the connecting line between the tables now has an arrow head indicating the direction of the join. The line goes from the main table to the supporting table.

Adding Columns

The purpose for joining the additional tables is to add some columns to your query. The lower pane of the Design View shows the columns that are included in the query. There are the following three ways to add columns to this grid:

- Double-click the column in the upper pane. This will add the column to the right of the existing columns.

www.it-ebooks.info

- Drag a column from the upper pane to the lower pane. This will add the column to the location you drag it to.
- Manually edit a column in the lower pane and select the table and field information.

In each table in the upper pane there is an "*" field. By double-clicking or dragging this, you will add all the columns of that table to the query. Use the double-click method to add the following columns to the query:

- `Item.Title`
- `Item.Author`
- `Loan.CheckedOut`
- `Loan.DueDate`
- `Loan.OverdueFee`
- `Customer.LastName`
- `Customer.FirstName`
- `Customer.PhoneNumber`
- `Customer.Email`

Save the query and run it. You should now see the additional details.

■ **Note** When using an outer join and the matching record is not found, a row is still returned, but the columns provided by the related table will be null. Also, keep in mind that if a match is not found, any subsequent joins from the related table will also fail.

Making Additional Changes

Display the `CheckedOutItems` query using the Design View. There are some fields, such as `CurrentLoanID`, that may not be useful. You can remove these from the query. Select the `CurrentLoanID` field and press the Delete key. Remove the `ItemID` field as well.

The `InventoryItem.Status` field is used to filter the records that are returned. All the rows that are returned will have the value "Checked Out." It seems unnecessary to include this field in the query, since all records will have the same value. However, you can't delete it, because it is used as a filter. Instead, unselect the Show check box for this field.

If you want to change the order of the fields, select a field and then, while the field is highlighted, click on the very top of the grid column and drag it to the desired location. You can select multiple columns by holding down the Shift key and selecting each column. Using this method, make the `Condition` and `Comment` fields the last fields in the query.

www.it-ebooks.info

Adding Calculated Columns

Go to the first empty column, which should be just after the `Comment` field. For the Field value enter the formula `Date() - Loan.DueDate`. When you tab off this field, the value will be changed as

```
Expr1: Date()-[Loan].[DueDate]
```

Access puts square brackets around table and field names since they can contain spaces. Notice that `Expr1:` was placed in front of the formula. `Expr1` is the name of the column that will be used when displaying the results. Change `Expr1` to `DaysOverdue`. Move this column to just after the `Status` field.

When entering expressions you can use the same Expression Builder that you used earlier. To display the Expression Builder, right-click on the Field value and click the *Build* link.

Sorting the Results

You can control the order in which the records are displayed by selecting one or more Sort fields. The grid in the lower pane has a Sort row. The default value is blank, which indicates it is not used for sorting. All of the fields are currently blank. To sort by a particular field, change this value to either `Ascending` or `Descending`. For the `DaysOverdue` field that you just added, change its Sort value to `Descending`.

You can sort on multiple fields. To add other Sort fields, simply set the sort value for that column in the grid. Set the Sort value for the `Title` field to `Ascending`. If there are multiple sort fields, they will be evaluated in the order that the fields are displayed. For this query you'll want the oldest items first and then sorted by the `Title`. Move the `DaysOverdue` column to before the `Title` field. The query grid should look like Figure 4-9.

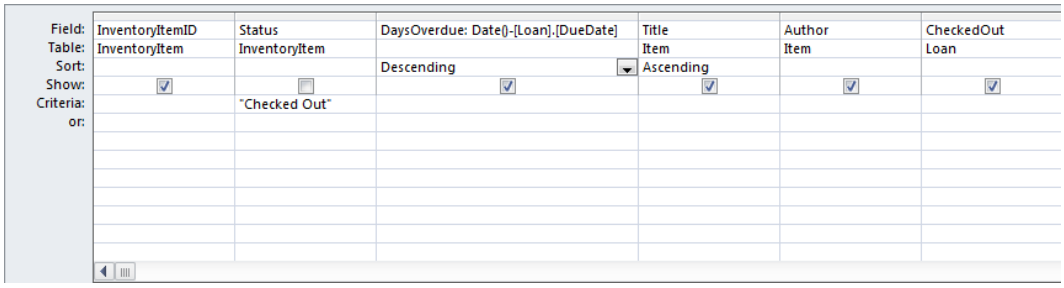


Figure 4-9. *The query grid with Sort fields*

Save and run the query. The results should look like Figure 4-10.

CheckedOutItems

InventoryItemID	DaysOverdue	Title	Author	CheckedOut
2	16	A Christmas Carol	Charles Dickens	1/2/2011 5:27:13 PM
3	12	A Tale of Two Cities	Charles Dickens	1/1/2011 11:14:37 AM
(New)				

Figure 4-10. *The final* `CheckedOutItems` *query*

www.it-ebooks.info

Using the SQL View

For those of you who like working with SQL, you can view and edit the SQL that is generated by the Design View. There are buttons displayed at the bottom right-hand corner of the application that allow you to change the view of the current object. Click the SQL View button, as shown in Figure 4-11.



Figure 4-11. *Selecting the SQL View*

The SQL that is displayed lacks any helpful formatting. By adding some white space characters, this can be formatted as shown in Listing 4-1.

Listing 4-1. *The* `CheckedOutItems` *SQL*

```
SELECT
   InventoryItem.InventoryItemID,
   Date()-[Loan].[DueDate] AS DaysOverdue,
   Item.Title,
   Item.Author,
   Loan.CheckedOut,
   Loan.DueDate,
   Loan.OverdueFee,
   Customer.LastName,
   Customer.FirstName,
   Customer.PhoneNumber,
   Customer.Email,
   InventoryItem.Condition,
   InventoryItem.Comment
FROM (Customer RIGHT JOIN Loan ON Customer.CustomerID = Loan.CustomerID)
RIGHT JOIN (Item RIGHT JOIN InventoryItem ON Item.ItemID = InventoryItem.ItemID)
    ON Loan.LoanID = InventoryItem.CurrentLoanID
WHERE (((InventoryItem.Status)="Checked Out"))
ORDER BY
   Date()-[Loan].[DueDate] DESC,
   Item.Title;
```

You can edit this SQL, but I don't recommend it. This might look similar to standard ANSI SQL, but it's not. Access has some peculiar way that it formats the `JOIN` logic. If you try to re-write this using Transact-SQL syntax, Access will not be able to parse it back into the designer.

■ **Caution** If you choose to modify the SQL, save the previous version of the SQL that was generated by Access. You will not be able to edit the query in the Design View if it has bad syntax. Rather than having to delete the query and start all over, you can paste in the previous version and get back to where you were.

---

www.it-ebooks.info

Using Queries as Views

You may have already noticed that, in most places where you need to select a table, you can use a query instead. This is a really useful feature that you will take advantage of throughout this project. Select queries are equivalent to a view in SQL Server. Just like tables, you can also insert and update rows. There are a couple of things you should be aware of when using a select query as a table.

In a denormalized query that joins multiple tables, the same information is often duplicated. The `CheckedOutItems` query, for example, includes details such as `Title` and `Author`. If there are multiple copies of the same item currently checked out, you will see the same `Item` details on each record. If you edit this information, the actual `Item` record will be modified, and every row in the query that uses that record will be updated automatically. I believe that is the desired behavior, but may be unexpected to the end user.

You should avoid inserting records into a query that joins multiple tables. One important reason for this is that often not all of the columns are included in the query. In that case you may not be able to supply all the required values. If you have a simple query that includes all the fields from two related tables you can use it to insert records. But there is no significant advantage to using the query instead of the actual table.

You can also use a query to define the lookup column when designing a table. In Chapter 2, I showed you how to use a use a lookup column on the `Loan` table to constrain the values of the `InventoryItemID` field. You selected the allowed values from the `InventoryItem` table. You could create query that only returns `InventoryItem` records where the `Status` is "Available." You could then use this query to define the lookup column, instead of the `InventoryItem` table, which would prevent you from selecting an `InventoryItem` record that was already checked out.

This works as expected and may seem like a great idea; however, there are a few drawbacks. If you define a lookup column using a query instead of a table, the Lookup Wizard does not create the relationship for you. Also, referential integrity is not supported with this approach. While you won't be able to select an invalid value, the database will not enforce this should someone try to delete the `InventoryItem` record. However, the biggest limitation is that this is not supported if you want to publish your database to SharePoint.

You should design the tables without using queries for the lookup columns. However, when building the forms that the end user will be using, you can use the query to populate a dropdown list. This will accomplish the same purpose without any of the limitations I mentioned.

Creating Action Queries

Action queries are used to insert, update, or delete records in a table. They are often used to perform maintenance functions. You'll now create a query that will insert a record into the `InventoryItem` table. This will be used whenever you receive a new copy into your inventory.

Creating the AddInventoryItem Query

A query that inserts records is called an *append* query in Access. To create an append query, from the Create tab of the ribbon, click the Query Design button. This will load the Design View and display the Show Table dialog box. Click the Close button without adding any tables. This will display a blank query.

www.it-ebooks.info

Changing the Query Type

By default, all new queries are added as a select query. In the Design tab, click the Append button as shown in Figure 4-12.

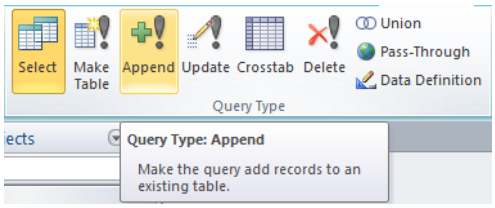


Figure 4-12. *Changing to an append query*

You will then be prompted to select the table that the query will insert into. Select the `InventoryItem` table, as shown in Figure 4-13.

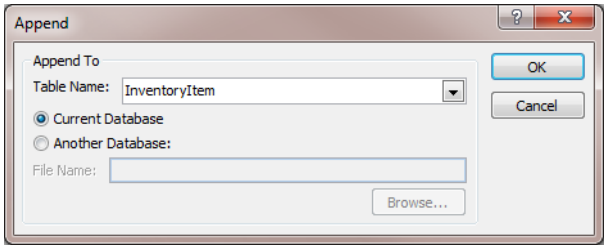


Figure 4-13. *Selecting the `InventoryItem` table to insert into*

The upper portion of the Design View is blank, and the lower pane has the familiar column grid that you used in the previous query. Notice that there is a row called Append To. Instead of returning columns in a display grid, each column in an append query populates a field in the target table. The Append To row is used to specify which field each column will populate.

I like to start by figuring out which fields need to be populated. Expand the dropdown list in the Append To field for the first column and select the first field that the query needs to populate. The `InventoryItemID` will be auto-populated, because it is defined as an `AutoNumber` field, so `ItemID` is the first field that the query needs to populate. In the next two columns, select the `Status` and `Condition` fields. The `Comment` field is not required and the `CurrentLoanID` should be null since the item is not loaned out. The lower pane should look like Figure 4-14.

www.it-ebooks.info

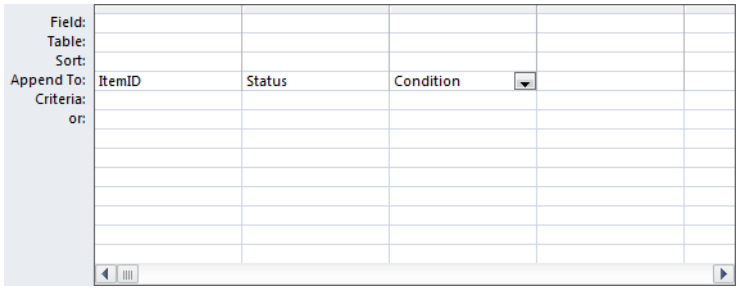


Figure 4-14. Selecting the output fields

The next step is to provide the values for each of these fields.

Adding a Parameter

The query will need to know which item this `InventoryItem` record should be created for. To do that you'll add a parameter to the query so the `ItemID` value can be passed in. To add a parameter, right-click somewhere in the upper (blank) pane and click the *Parameters* link.

This will display the Query Parameters dialog box. Enter **itemID** for the Parameter and **Long Integer** for the Data Type as shown in Figure 4-15. Click the OK button to save the changes.

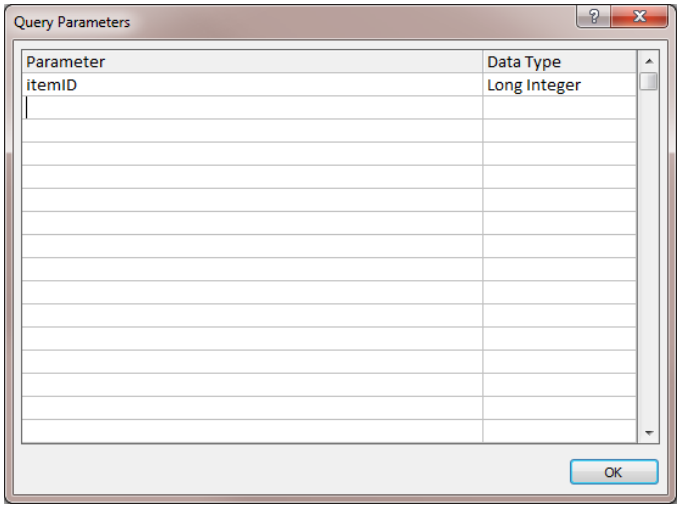


Figure 4-15. Defining the parameters

Now you're ready to specify the values for each of the three fields that will be populated. Enter the following:

- `ItemID`: **itemID**

www.it-ebooks.info

- `Status`: "Available"
- `Condition`: **"New"**

The query should look like Figure 4-16.

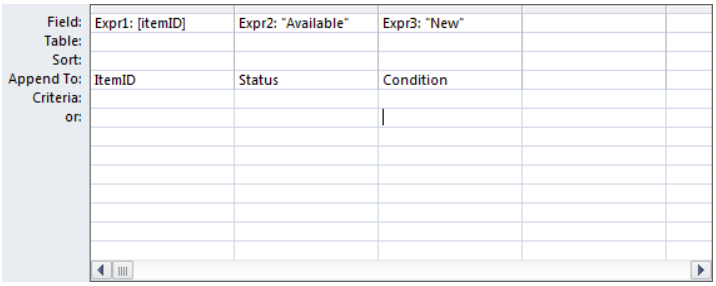


Figure 4-16. *The final query design*

Running the Query

Save the query and enter **AddInventoryItem** for the query name when prompted. Click the Run button in the Design tab. You should see a pop-up dialog box, shown in Figure 4-17, requesting the value of the `itemID` parameter. Enter **1** and click OK.

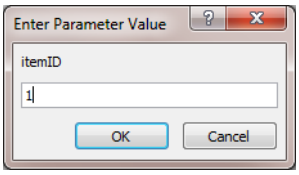


Figure 4-17. *Entering the `itemID` parameter*

■ **Note** Later in this book, you'll call this query from a form and write code to pass the `itemID` programmatically. For now, since no value was supplied, the query must prompt for it.

You should then see the warning shown in Figure 4-18. The record has been added, but Access is asking for confirmation before the insert is committed. Click the Yes button.

www.it-ebooks.info

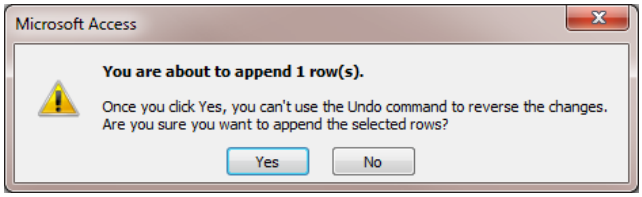


Figure 4-18. *Append data warning*

Open the `InventoryItem` table and verify that a new record has been added for `ItemID` 1.

Enhancing the Request Feature

You've just received an enhancement request: "Any customer request that has not been picked within seven days should be taken off hold and put back on the shelf." The `Request` table has a `RequestDate` field, which stores the date the item was requested by the customer. You'll need another field to store the date that the item was put on hold. You will need to record this date so you'll know when it has been more than seven days.

Modifying the Request Table

So, let's modify the `Request` table:

1. Open the `Request` table using the Design View.
2. Add a field named `ReadyDate` and select `Date/Time` for the Data Type. You can leave all the default properties except Show Date Picker. Set this to Never. This is not a value that the end user should be selecting.
3. Open the `InventoryItem` table using the Datasheet View. Edit the named macro, `AssignPendingRequest`, which you entered in Chapter 3.
4. Select the `EditRecord` action, which will add an Add New Action dropdown list inside this data block. Select the `SetField` action in this dropdown list. For the Name property enter `Request.ReadyDate` and for the Value property enter `Now()`.
5. The completed macro should look like Figure 4-19. Save and close the macro.

www.it-ebooks.info



Figure 4-19. *The modified* `AssignPendingRequest` *macro*

Now you'll need to update this table to set the `ReadyDate` field for the existing records. To do that you'll use an *update* query.

1. From the Create tab, click the Query Design button, select the `Request` table in the Show Table dialog box, and then close the dialog box.
2. Click the Update button in the Design tab to change this query from a select query to an update query.
3. Drag the `RequestDate` field from the upper pane to the grid in the lower pane. In the Update To row, enter `Now()` and in the Criteria row, enter **Is Null**. The query should look like Figure 4-20.

www.it-ebooks.info

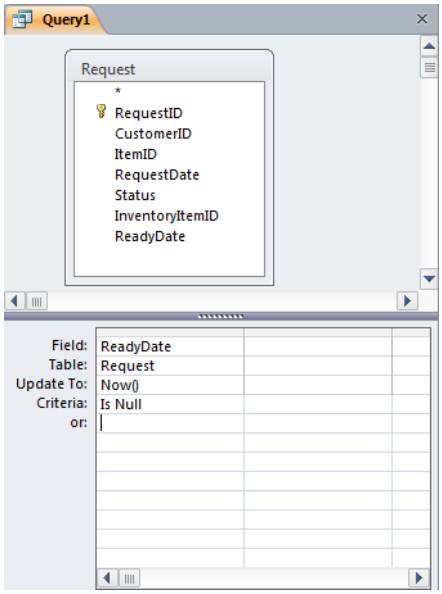


Figure 4-20. *Setting the initial* `RequestDate` *value*

4. Run this query by clicking the Run button in the Design tab. You will be prompted about rows being updated. Click the Yes button to confirm the update.
5. You can close the query without saving it. You won't need to run this again.
6. Open the `Request` table using the Datasheet View and verify that all records have the `ReadyDate` field populated.
7. If there is already a record in the `Request` table with a "Ready" status, change its `ReadyDate` field to be at least a week old. If not, add a new record, set its `Status` to "Ready" and select an `InventoryItem` record.
8. Finally, open the `InventoryItem` table and change the `Status` on the selected record to "On Hold."

Creating the CancelOldRequests Query

Now you're ready to implement an update query to cancel any requests that have been ready for more than seven days.

1. From the Create tab, click the Query Design table. In the Show Table dialog box, select the `Request` table and then close the dialog box.
2. Drag the `ReadyDate` and `Status` fields from the upper pane to the grid in the lower pane.

www.it-ebooks.info

3. In the ReadyDate column, enter **< Now() - 7** in the Criteria row.
4. Enter **"Ready"** in the Criteria row for the Status column.
5. Click the Run button in the Design tab and verify that the correct records are displayed.

■ **Tip** So far you have only created a select query; it does not update anything, yet. I find it very helpful to implement an update query as a select query first. You can then run the query to verify that the criteria is selecting the correct set of rows that you want to update. After you have done that, you can easily change the query into an update query.

---

6. The query should return at least one record that has a Status of "Ready" where the ReadyDate is more than seven days old. If it doesn't, open the Request table and edit the data so you'll have a record to test with.
7. Go back to the Design View by clicking the View button in the ribbon.
8. Click the Update button in the Design tab to make this an update query.
9. For the Status column, enter **"Cancelled"** in the Update To row. The query will update all the rows selected by the criteria, which you just tested, and change the Status to "Cancelled."
10. Save the query and enter the name CancelOldRequests when prompted.

Adding a Data Macro

The query will update the appropriate Request records, changing their Status to "Cancelled." When the request is cancelled, the associated InventoryItem record needs to be changed as well, so it is available for other customers. Therefore, before you run this query against the Request table, you'll need to create a data macro to also update the associated InventoryItem record.

1. Open the Request table using the Datasheet View.
2. In the Table tab, click the After Update button to design the After Update macro.
3. In the Macro Editor, enter the actions shown in Figure 4-21.

www.it-ebooks.info

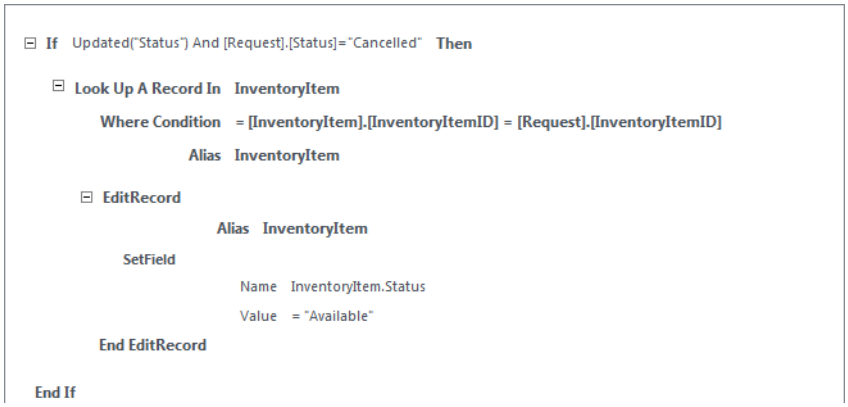


Figure 4-21. The `Request.AfterUpdate` macro

This macro checks to see if the `Status` field has been updated to "Cancelled." If it has, the macro then looks up the associated `InventoryItem` record and changes its `Status` to "Available." You can refer to Chapter 3 for more information about data macros.

Running the Query

You will probably want to run this query periodically so to make it easy, you'll create a macro to run it.

1. Click the Macro button in the Create tab.
2. In the "Add New Action" dropdown list, select the `OpenQuery` action. For the Query Name property, select `CancelOldRequests` from the dropdown list. Select Datasheet for the View property and Read Only for the Data Mode property. The action should look like Figure 4-22.



Figure 4-22. The `CancelOldRequests` macro

3. Save the macro and enter the name `CancelOldRequests` when prompted. You should see this macro added to the navigation pane on the left-hand side of the application.
4. Right-click this macro and then click the *Run* link. You will see a pop-up window, as shown in Figure 4-23. Click the Yes button to allow the query to run.

www.it-ebooks.info

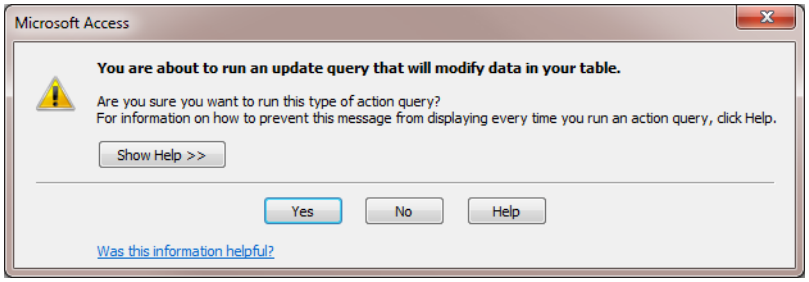


Figure 4-23. *Warning about running an update query*

5. You will also see another pop-up dialog box showing how many records will be updated. Click the Yes button to confirm the update.
6. Open the `Request` table and verify that the appropriate records have been cancelled. Also, check the `InventoryItem` records to see if they are now available.

Creating a Crosstab Query

For the final query sample you'll use the Query Wizard to create a more complex *crosstab* query. The term crosstab is a shortened form of *cross tabulation*. A crosstab query uses aggregate functions to display summary information. For example, you'll create a query that will show you the total number of loans that have been made for each category and media type. The contents of the `Loan` table are summarized (tabulated) based on one or more attributes of that table. This will indicate what types of items have been the most popular.

Building the AllLoans Query

A crosstab query uses a single table or query for its data source. Because this information is spread across several tables, you'll first need to create a simple select query that joins all the necessary tables and provides a denormalized view. This will be similar to the `CheckedOutItems` query that you created earlier. The main table in this case is the `Loan` table because you want to gather statistics on what has been loaned out. The other tables are joined to get details about the item being loaned. Here's how to build the query that joins all the necessary tables:

1. From the Create tab of the ribbon, click the Query Design button. In the Show Table dialog box, select the following tables:
   - `Category`
   - `InventoryItem`
   - `Item`
   - `Loan`
   - `Media`

www.it-ebooks.info

2. Click the Add button to add these tables to the query then click the Close button.

3. You'll need to rearrange the tables so the connecting lines are easier to follow. Also, there are two joins between the `Loan` and `InventoryItem` tables. With this query, remove the one between `Loan.LoanID` and `InventoryItem.CurrentLoanID`. The query design should look like Figure 4-24.

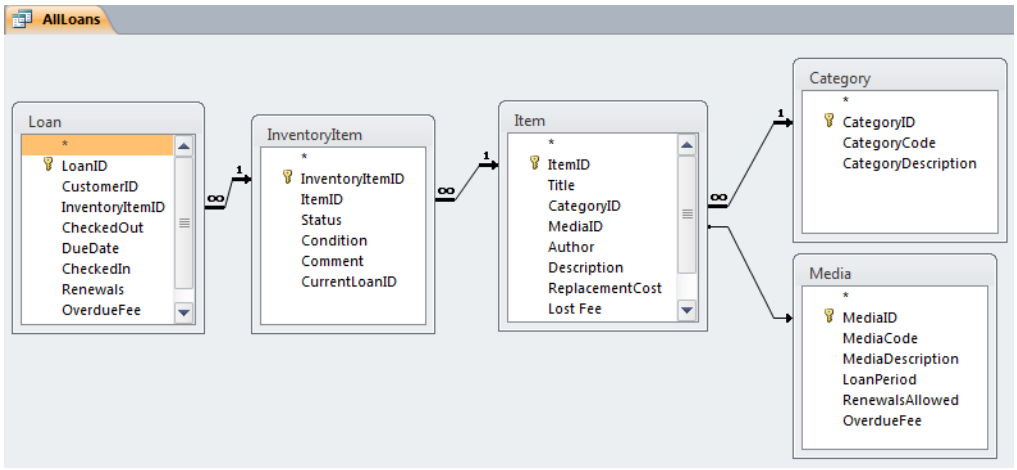


Figure 4-24. *The `AllLoans` query design*

4. Just like with the `CheckedOutItems` query, right-click each of these joins, select the *Join Properties* link and select the third option, which is a right outer join.

5. Save the query and enter the name `AllLoans` when prompted.

6. Now you'll need to enter the fields that should be returned. Double-click each of the following fields to add them to the grid in the lower pane:

   - `Loan.LoanID`
   - `Loan.CheckedOut`
   - `Loan.DueDate`
   - `Loan.OverdueFee`
   - `Item.Title`
   - `Item.Author`
   - `Item.ReplacementCost`
   - `Category.CategoryCode`
   - `Category.CategoryDescription`

www.it-ebooks.info

- `Media.MediaCode`
- `Media.MediaDescription`

7. Save the query and then click the Run button in the Design tab to test the query. You should have one record for each record in the `Loan` table.

■ **Note** A crosstab query is used to summarize information. To make the sample query more interesting, you'll probably need to create more `Item`, `InventoryItem`, and `Loan` records. Make sure at least several of the categories and media types are represented in the `Loan` table.

Designing the LoanSummary Query

Now you're ready to create a crosstab query. Follow these steps:

1. From the Create tab, click the Query Wizard button. In the New Query dialog box, select the Crosstab Query Wizard, as shown in Figure 4-25.

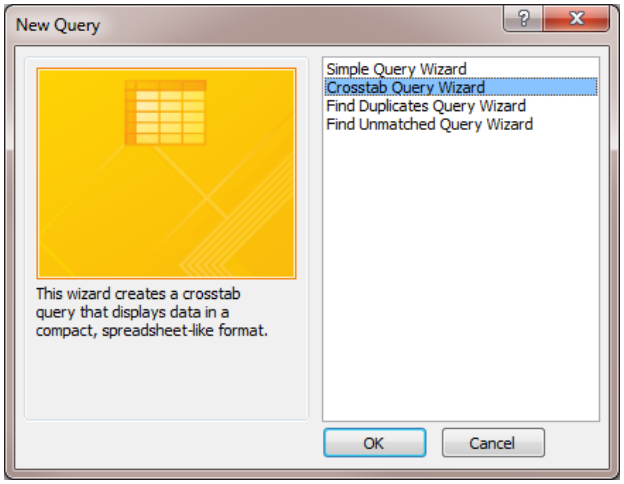


Figure 4-25. *Selecting the Crosstab Query Wizard*

2. The first step is to select the table or query that contains the data that is to be summarized. Choose the Queries option and then select the `AllLoans` query that you just created as shown in Figure 4-26.

www.it-ebooks.info

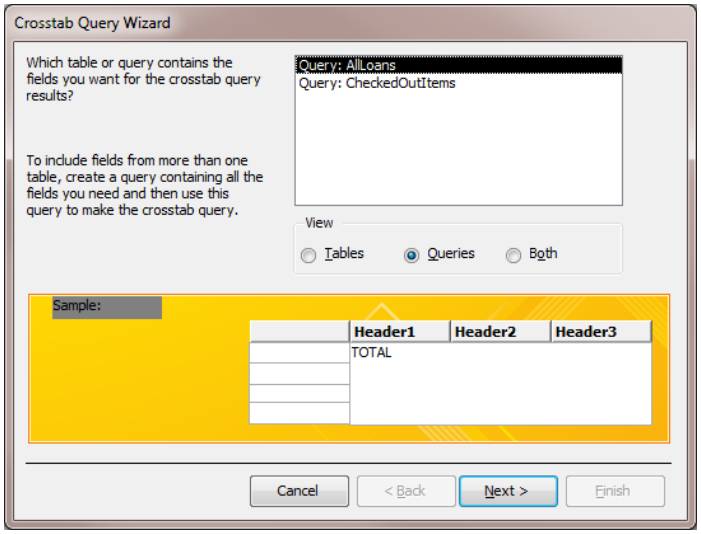


Figure 4-26. *Selecting the* `AllLoans` *query*

3. A crosstab query allows you to summarize data by grouping them by two different values. This query, for example, will have a row for each category and a column for each media type. In the second dialog box, shown in Figure 4-27, you'll select the field used for the row labels. Select the `CategoryDescription` field and click the ">" button to add it to the Selected Fields list.

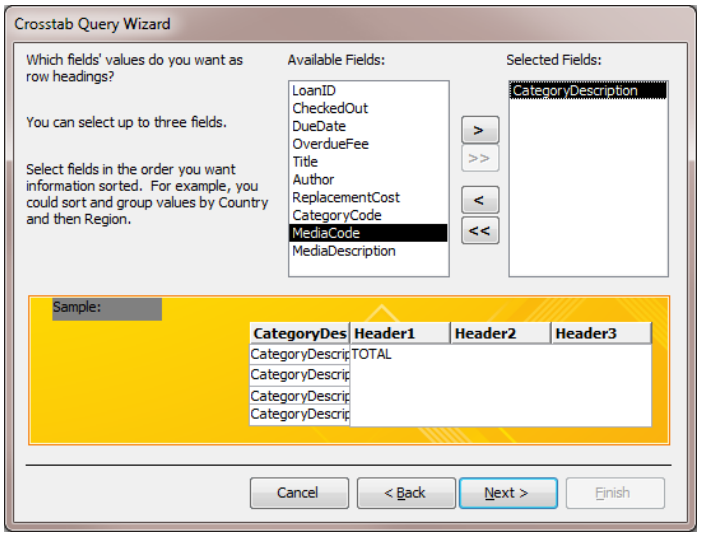


Figure 4-27. *Selecting* `CategoryDescription` *for the row labels*

www.it-ebooks.info

4. In the next dialog box, you'll select the field that should be used for the column headings. Select the `MediaDescription` field as shown in Figure 4-28.

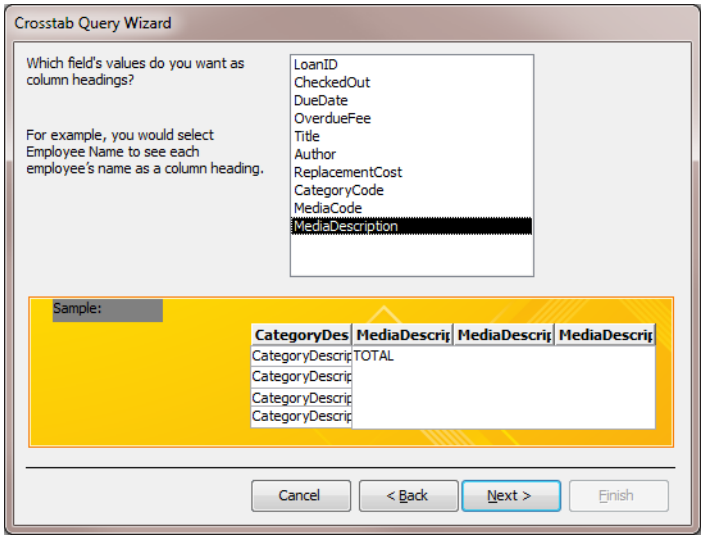


Figure 4-28. *Selecting* `MediaDescription` *for the column headings*

5. The last step in designing the crosstab query is to select the field that will be used to generate the data in each cell. This will use an aggregate function such as `Count`, `Sum`, `Avg`, or `Max`. This query will show the number of loans made for each category/media type. To do this, use the `Count` function. Since the function is simply counting the number of records, the field you use doesn't really matter. However, `LoanID` is a logical choice, since it will be unique for each row. Select the `LoanID` field and the `Count` function as shown in Figure 4-29.

www.it-ebooks.info

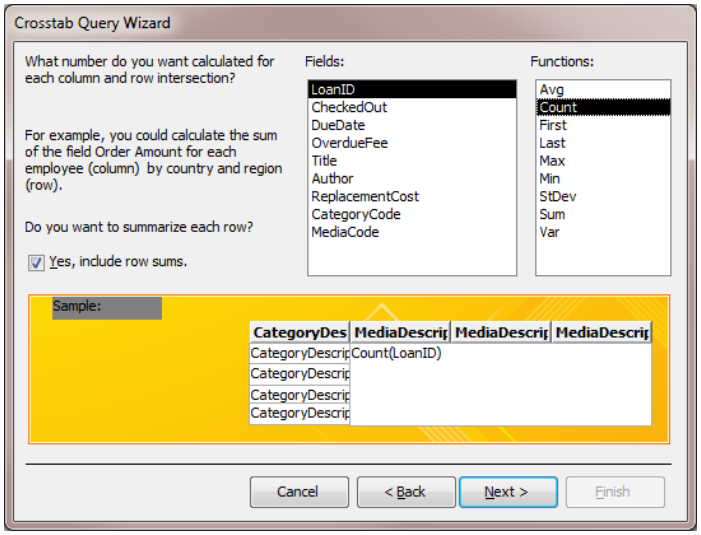


Figure 4-29. *Selecting the aggregate function to use*

6. In the final dialog box, shown in Figure 4-30, enter **LoanSummary** for the query name.

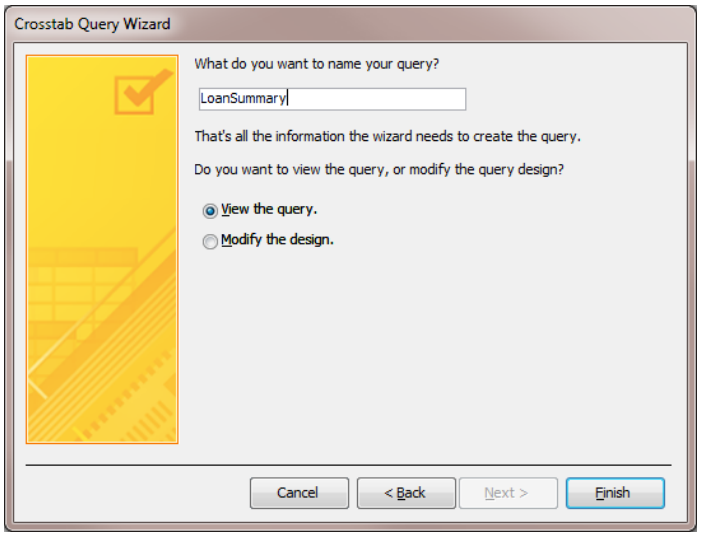
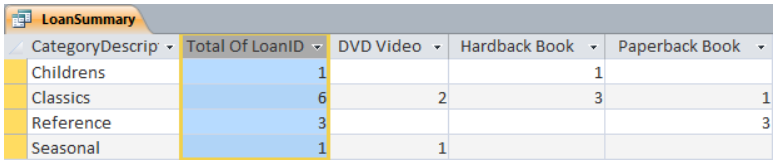


Figure 4-30. *Entering the query name*

www.it-ebooks.info

7. Click the Finish button to create the query, close the wizard and display the query using the Datasheet View. The results should be similar to Figure 4-31, depending on the data you entered.

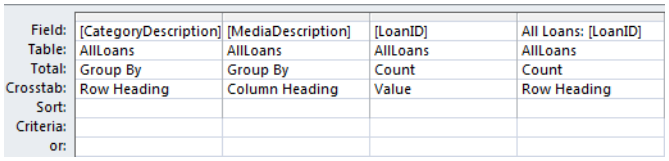


LoanSummary

CategoryDescrip	Total Of LoanID	DVD Video	Hardback Book	Paperback Book
Childrens	1		1	
Classics	6	2	3	1
Reference	3			3
Seasonal	1	1		

Figure 4-31. *The completed LoanSummary query*

8. The column heading of the second column is not quite right. Open the query using the Design View. Change the Field description of the last column to **All Loans: [LoanID]** as shown in Figure 4-32.



Field:	[CategoryDescription]	[MediaDescription]	[LoanID]	All Loans: [LoanID]
Table:	AllLoans	AllLoans	AllLoans	AllLoans
Total:	Group By	Group By	Count	Count
Crosstab:	Row Heading	Column Heading	Value	Row Heading
Sort:				
Criteria:				
or:				

Figure 4-32. *Changing the column heading*

9. Save the query and then click the Run button in the Design tab.

Summary

In this chapter you created several select and action queries using both the Query Wizard and the query Design View. Some of the key benefits of using queries that you demonstrated in this chapter include the following:

- Providing a subset of the data from a table
- Creating a denormalized view by joining several related tables
- Inserting records into a table
- Updating a group of records in batch mode
- Calling an action query from a macro
- Summarizing aggregated data using a crosstab query

In the next and final chapter in this part of the book, you'll see the power of creating pivot tables to assist in reporting analyzing and data.

www.it-ebooks.info

CHAPTER 5

Creating PivotTables

PivotTables are a great way to analyze a large amount of data. They operate on a table that contains raw data and produces a table of summary data. A PivotTable allows you to summarize information in a variety of ways. This ability to view the same data from multiple perspectives gives PivotTables their name. PivotTables were designed to rotate, or pivot, the data around a particular aspect. Consider a cylinder, for example. From the end it looks like a circle, while from the side it looks like a rectangle. A PivotTable provides graphical tools to quickly change perspectives so you can determine which views are most useful.

In this chapter I'll show you how to create a PivotTable and demonstrate many of the features in Access 2010 for using and customizing them.

Slicing, Dicing, and Drilling

In most applications, you will often have one or more large tables that you'll need to provide some way to extract information from. For example, in our `Library` project, the `Loan` table can be quite large. It has a record for each time an item is checked out. Even in a modest-sized library, you could have thousands of records created each day. You can analyze a large amount of data more effectively by breaking it down into various views that each describes a different aspect of the data.

A PivotTable starts with a value that you need to report. This could be the total sales, number of customers, or average response time. This value is expressed as a statistical computation such as `Sum`, `Count` or `Average`. You could provide a single value for the entire data set; however, most of the time you'll want to see that value expressed as a function of some other attribute. For example, "total sales per day" or "average response time for each type of request."

This technique is called *slicing*; the entire data set is cut into slices with each slice representing an attribute value such as date or request type. The value (total sales or response time) is then computed for each slice. For example, you could slice the `Loan` table using the media type. This would allow you to see how books or videos are moving. Likewise you could slice the table by the category to see whether Children's or Classics were more popular.

You'll often need to slice the data in multiple ways simultaneously. Suppose you wanted to determine the most popular media type for each category. This is what it means to *dice* the data. (When you cut a tomato in one direction you get slices, but when you cut it in multiple directions you get diced tomatoes.) The value is then computed for each combination of attribute values, such as Seasonal DVDs or Children's Books.

If you were to graph these values, you might display the media type on the X-axis and the category on the Y-axis. Because of this, each attribute that is used to dissect the data is sometimes referred to as a *dimension*. A dimension is an aspect that you'll use to slice the data. Media types and categories are good candidates for dimensions. With a PivotTable, you can dice your data in a theoretically unlimited

www.it-ebooks.info

number of directions; however, it is difficult to present more than two or three dimensions at a time. As you'll see in this chapter, Access 2010 has some creative ways to visually present multiple *dimensions*.

Often you need to *drill* into your data, which is to increasingly narrow your focus. Let's say, for example, that want to see when your busiest times are. You can start by finding the busiest days and then drill down by hours to determine more specific time periods. Drilling is analogous to taking a single slice and cutting it into smaller pieces. You can take one of the smaller pieces and further dissect it, if necessary.

■ **Note** PivotTables are provided in other tools, such as SQL Server Reporting Services (SSRS) and Excel. The concepts are similar in each tool, but there are differences in their implementation and the specific features that are provided.

Creating a PivotTable View

A PivotTable is not actually a *table*, but a specialized *view* of a table. You've already used the Datasheet View, Design View, and SQL View. In Access, you'll use the PivotTable View to display an existing table. As I explained in Chapter 4, you can also use a select query just like a table, so you can use the PivotTable View on a query as well.

There are two important implications to this approach.

- First, all the columns that you want included in the PivotTable must be in a single table or query. If you've done a proper normalized table design (see Chapter 2), your tables will not likely work too well for this purpose. Instead, you will probably need to create a query that joins all the necessary tables, as explained in Chapter 4.
- Second, Access does not actually store the summarized data that is displayed in a PivotTable. When you save a PivotTable View, Access saves the view's configuration, not the data. You may notice a pause when you open a PivotTable View, because Access has to load the entire table or query and re-calculate the summary information. Also, you cannot edit the summarized data. When data in the underlying table is changed, the PivotTable View is *not* automatically updated, but it can be easily re-summarized. I'll show you how to do that later in the chapter.

In this chapter, you'll use the `AllLoans` query that you created in Chapter 4. It provides all the loan details that you'll need for your PivotTable. So let's open the PivotTable View.

1. Open the `AllLoans` query, which should open in the default Datasheet View.
2. Click the dropdown button under the View button in the ribbon. This will display all the available views.
3. Select the *PivotTable View* link, as shown in Figure 5-1.

www.it-ebooks.info

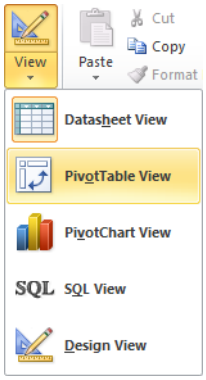


Figure 5-1. *Selecting the PivotTable View*

You can also click the PivotTable View button in the lower-right corner of the application, as shown in Figure 5-2.

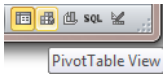


Figure 5-2. *Selecting the available views*

Understanding the PivotTable Layout

The initial blank PivotTable is shown in Figure 5-3. To design it, you'll add fields from the underlying table – in this case, the `AllLoans` query – to one of four areas.

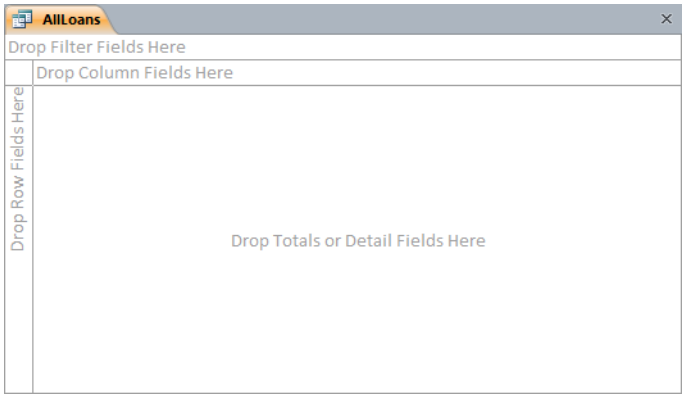


Figure 5-3. *The initial blank PivotTable*

www.it-ebooks.info

The large section in the center is where you'll place the value(s) that will be summarized. This is referred to as the Data area. Each record from the `AllLoans` query will fall into one of the cells in this area. Typically, each cell represents an aggregate function of the records assigned to that cell. For example, the cell could be the *sum* of `OverdueFee`, the *count* of `LoanID`, or the *maximum* `DaysOverdue`. In this area, you'll specify the field and the aggregate function such as `sum`, `count`, and `max`, to be applied to that field. The fields that are placed in this section are called Totals fields, because they are often sums or counts. The more generic name Detail field is also used, because not all aggregate functions produce totals.

The areas above and to the left of the main section contain the Column and Row fields, which are used to specify which cell each record is assigned to. For example, the Column field could be `CategoryDescription` and the Row field could be `CheckedOut`. Access finds the appropriate cell for each record by looking up its values for these fields. Each cell represents a unique combination of category and the date the item was checked out.

The section at the very top contains Filter fields. As I mentioned, you can only effectively display two dimensions at a time. The Filter field provides a third dimension, but does so by limiting the view based on the value of that field. For example, if you used `MediaDescription`, you could analyze one media type at a time. You could also select any subset of media types, including all of them.

You should also see a PivotTable Field List window, which is shown in Figure 5-4. This shows you all the columns in the `AllLoans` query. If this window is not currently visible, click the Field List button in the Design tab of the ribbon.

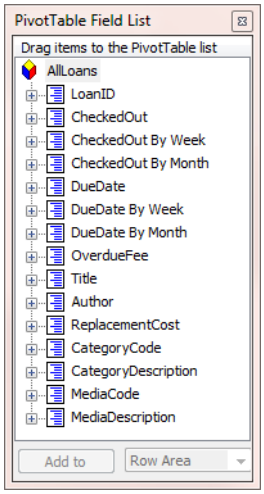


Figure 5-4. *The PivotTable Field List window*

To design the PivotTable, simply drag the desired columns from this window to the appropriate areas on the PivotTable View.

Using the PivotTable Fields

This PivotTable will be used to analyze what items are being checked out, so the first thing you'll specify is the data values that will provide that information. This will define what the view is summarizing. In this case, you'll present the number of items that have been loaned out. A `Loan` record is created each

www.it-ebooks.info

time an item is borrowed, so you'll simply count the number of records. To do that, use the `LoanID` field and the `Count` function, as follows.

4. Drag the `LoanID` field to the data area. The view will look similar to Figure 5-5.

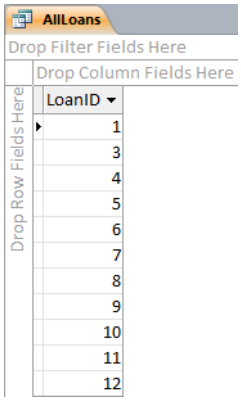


Figure 5-5. *Adding the* `LoanID` *field*

5. By default, the PivotTable View shows the details by listing all the records that are represented in each cell. Because no aggregate function has been specified yet, the data area will show no totals. Select the `LoanID` field in the data area and then click the AutoCalc button in the ribbon and select the *Count* link as shown in Figure 5-6.

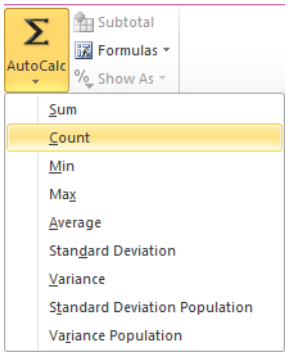


Figure 5-6. *Selecting the* `Count` *aggregate function*

6. This will add a row to the data area that represents the `Count` of loans. Click the Hide Details button in the ribbon to suppress the details. The PivotTable View will now show only the total number of loans, as shown in Figure 5-7.

www.it-ebooks.info

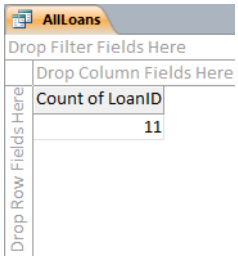


Figure 5-7. *The total loan count*

7. In my database, I have loaned out a total of 11 books. The column heading "Count of LoanID," while accurate, is probably not the best presentation. To change this, right-click this cell in the data area and select the *Properties* link. Select the Captions tab and enter **Loans** in the Caption field, as shown in Figure 5-8.

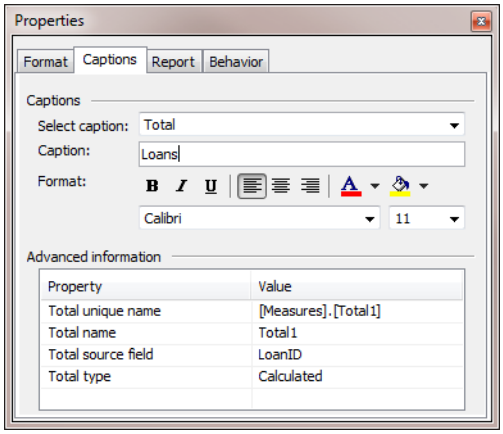


Figure 5-8. *Changing the field caption*

Notice that you can also adjust other properties such as the font through the Properties dialog box.

Defining the Column Field

Now you're ready to define the dimensions that will be used to evaluate the number of loans. Start by dragging the `CategoryDescription` field to the Column section. The PivotTable View will now include a column for each unique value of `CategoryDescription`. Notice that there is now a column for Childrens, Classics, Reference, and Seasonal, as shown in Figure 5-9. There is also a Grand Total column that presents the overall total from all categories.

www.it-ebooks.info

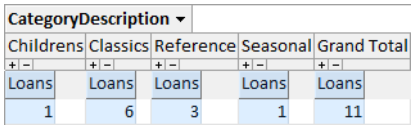


Figure 5-9. *Adding the* `CategoryDescription` *to the Column area*

■ **Note** The PivotTable View does not necessarily list every category defined in the `Category` table; it only displays the categories that are represented in the `AllLoans` query. For example, I also have a Fiction category that is not shown, because no item from this category has been loaned yet.

There is a small "+" and "-" button under each column header. You can use these to show the details for each cell. In my database, there are three loans for items in the Reference category. By clicking the "+" button, you can see that `LoanID` 9, 11, and 12 were for Reference items, as shown in Figure 5-10.

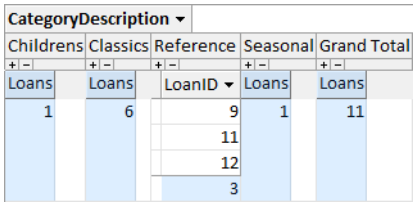


Figure 5-10. *Displaying the details of a cell*

Using Date Fields

The `CheckedOut` field records the date/time when the item was checked out. Drag this to the Row section. Because you're the only user, no two items can currently be checked out at exactly the same time. Because each `Loan` record will have a different value in the `CheckedOut` column, you'll end up with a row for each item, as shown in Figure 5-11.

www.it-ebooks.info

	CategoryDescription									
	Childrens		Classics		Reference		Seasonal		Grand Total	
CheckedOut	Loans		Loans		Loans		Loans		Loans	
1/1/2011 11:14			1						1	
1/2/2011 17:24			1						1	
1/2/2011 17:27			1						1	
1/2/2011 17:28			1						1	
1/8/2011 21:51							1		1	
1/8/2011 22:01			1						1	
1/8/2011 22:02			1						1	
1/8/2011 22:02					1				1	
1/8/2011 22:02	1								1	
1/8/2011 22:02					1				1	
1/8/2011 22:02					1				1	
Grand Total	1		6		3		1		11	

Figure 5-11. *Adding the* `CheckedOut` *field*

Date fields are actually a composite of several values, such as year, month, day, and so on. This makes them very useful in a PivotTable View. Notice in the PivotTable Field List window, that the `CheckedOut By Month` and `CheckedOut By Week` fields are included in addition to the `CheckedOut` field. Similar fields were also added for the `DueDate` field.

Remove the `CheckedOut` field from the PivotTable View by right-clicking on the field and selecting the *Remove* link. You can also select the field and click the Remove Field button in the Design tab of the ribbon. Drag the `CheckedOut By Month` field to the Row section. Expand the Years, Quarters, Months, and Days columns to drill into the data, as shown in Figure 5-12.

						CategoryDescription				
						Childrens	Classics	Reference	Seasonal	Grand Total
Years	Quarters	Months	Days	Hours	Minutes	Loans	Loans	Loans	Loans	Loans
⊟ 2011	⊟ Qtr1	⊟ Jan	⊞ 01-Jan				1			1
			⊟ 02-Jan	⊟ 17	⊞ 24		1			1
					⊞ 27		1			1
					⊞ 28		1			1
					Total		3			3
				Total			3			3
			⊞ 08-Jan			1	2	3	1	7
			Total			1	6	3	1	11
		Total				1	6	3	1	11
	Total					1	6	3	1	11
Grand Total						1	6	3	1	11

Figure 5-12. *Drilling into the date properties*

This is an ideal example of drilling into your data. Starting with the total for the year, you can then expand a specific quarter and month to see the daily totals. The "+" and "-" button on each summary row allow you to expand or collapsed that particular value.

Notice the dropdown indicator next to the `Years` column label. If you click this you'll see the dialog box shown in Figure 5-13. This allows you to select the years that you want included in this view.

www.it-ebooks.info

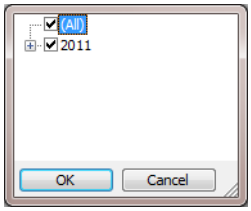


Figure 5-13. *Selecting the years to include in the view*

This is only allowed at the top level of the date hierarchy. The same dropdown button is displayed next to the `CategoryDescription` column. In the same way you can choose to limit the categories that will be included in the view.

■ **Tip** The `CheckedOut By Week` field works just like `CheckedOut By Month`, except the hierarchy is presented as Year, Week, and Day of Week. Months do not divide into weeks very well. This field divides each year into 52 weeks, but does not group by quarter or month. (Every 5 or 6 years has 53 weeks.) Likewise, the `CheckedOut By Month` field divides a year into quarters and months, but does not group into weeks. Both will present totals by year and day. You'll need to use one or the other depending on whether the quarter/month or week grouping is more useful for your analysis.

Adding a Filter Field

Drag the `MediaDescription` field to the Filter area. The Filter area is used to limit the records that are represented in the view. By default, the All value is selected, so this doesn't affect the results in the PivotTable View. Click the dropdown button, which will list the values that can be selected as shown in Figure 5-14.

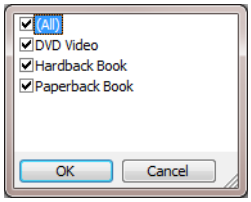


Figure 5-14. *Selecting the filter values*

Unselect the All option and then select DVD video. The PivotTable View should look like Figure 5-15.

www.it-ebooks.info

MediaDescription									
DVD Video									
						CategoryDescription			
						Classics		Seasonal	Grand Total
Years	Quarters	Months	Days	Hours	Minutes	Loans		Loans	Loans
⊟ 2011	⊟ Qtr1	⊟ Jan	⊞ 08-Jan			2		1	3
			Total			2		1	3
		Total				2		1	3
	Total					2		1	3
Grand Total						2		1	3

Figure 5-15. *Restricting the view to only include DVD Video items*

If you select more than one value for the filter, under the `MediaDescription` column heading it would display "(Multiple Items)."

Refreshing the PivotTable

As I mentioned earlier, a PivotTable is a summarized snapshot of your underlying table. It is not automatically updated as data is added or modified. This is done primarily for performance reasons. The summary data is recalculated when the PivotTable view is opened. Usually, when doing data analysis, you're not concerned with real-time, up-to-the-minute data, so this is a reasonable compromise.

If you need to re-calculate the data PivotTable based on the current data, just click the Refresh Pivot button on the Design tab of the ribbon, as shown in Figure 5-16.

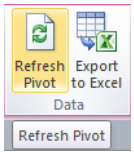


Figure 5-16. *Using the Refresh Pivot feature*

Adding Other Fields

So far, I've covered the basic usage of a PivotTable View. You specify a field that contains the data values (count of `LoanID` in this case) and then add fields to the Column, Row, and Filter sections that allow you summarize the data using these attributes. Now I'll show you how to extend this functionality.

Including Multiple Values

The view currently has a single data point that represents the number of loans. You can add additional data points. Suppose you also wanted to know how much was charged in overdue fees. To do this, simply add the `OverdueFee` field to the Data area.

8. In the PivotTable Field List window, select the `OverdueFee` field and then select the Data Area, as shown in Figure 5-17.

www.it-ebooks.info

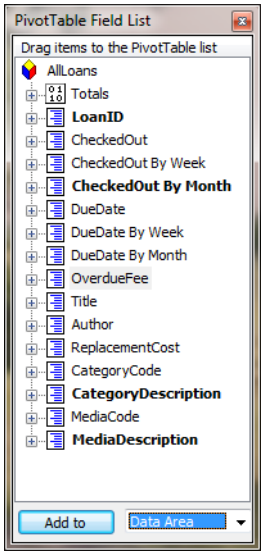


Figure 5-17. *Adding* `OverdueFee` *to the Data area*

9. Click the Add to button to add this to the PivotTable View. The `Sum` function is used by default, which is correct for this field, because you want the total amount of overdue fees that were charged. However, the caption is somewhat wordy.

10. Right-click this new column in the Data area and change the caption to **Fees** as shown in Figure 5-18.

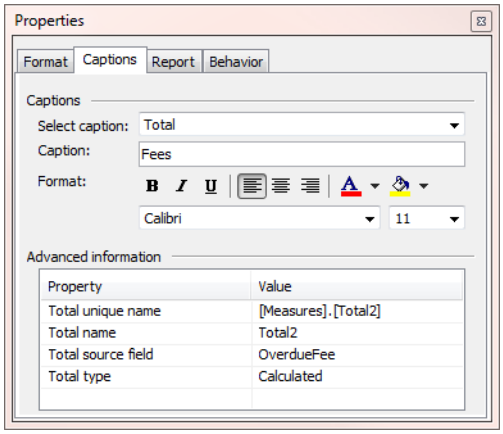


Figure 5-18. *Changing the field caption*

www.it-ebooks.info

The modified PivotTable View should look like Figure 5-19.

						CategoryDescription									
						Childrens		Classics		Reference		Seasonal		Grand Total	
Years	Quarters	Months	Days	Hours	Minutes	Loans	Fees	Loans	Fees	Loans	Fees	Loans	Fees	Loans	Fees
⊟ 2011	⊟ Qtr1	⊟ Jan	⊞ 01-Jan					1	$0.00					1	$0.00
			⊞ 02-Jan					3	$5.00					3	$5.00
			⊞ 08-Jan			1	$0.00	2	$0.00	3	$0.00	1	$0.00	7	$0.00
			Total			1	$0.00	6	$5.00	3	$0.00	1	$0.00	11	$5.00
		Total				1	$0.00	6	$5.00	3	$0.00	1	$0.00	11	$5.00
	Total					1	$0.00	6	$5.00	3	$0.00	1	$0.00	11	$5.00
Grand Total						1	$0.00	6	$5.00	3	$0.00	1	$0.00	11	$5.00

Figure 5-19*. The PivotTable View with a second data field*

Using Calculated Columns

Now you'll add one more data value, which is the number of items that were (or are) overdue. The current `AllLoans` query does not have a field with that information, however. So you'll need to first modify the query and add a calculated column.

Open the `AllLoans` query and switch to the Design View by using the View button in the Design tab of the ribbon. Scroll to the first empty column in the lower pane of the Design View. Right-click in the Field row and select the *Build* link. Enter the following formula:

```
Overdue: -1*((Not IsNull([CheckedIn]) And [CheckedIn]>[DueDate]+1) Or (IsNull([CheckedIn])↩
 And [DueDate]<Now()-1))
```

An item is considered overdue if it was checked in after the due date or if it is not yet checked in but the due date is in the past. The formula uses the `IsNull` function to determine if the item has been checked in. Because the item is not considered overdue until midnight on the due date, the formula must add a day to the due date. (The `DueDate` field does not contain a time portion; see Chapter 3 for details.)

In Visual Basic, a `Boolean` field has a numeric value of 0 when false, and -1 when true. To account for this, the formula multiples the result by -1. So every record will have a value of 1 if overdue and 0 if not. The PivotTable View can simply sum this new column to get a count of overdue items.

Switch to the Datasheet View and verify the values of this column are correct for each record. Then go back to the PivotTable View and you should see the new `Overdue` field in the PivotTable Field List window. Add this field to the Data area just like you did with the `OverdueFee` field. Change the caption of this column to **Overdue**. The PivotTable View should look like Figure 5-20.

www.it-ebooks.info

CategoryDescription														
Childrens			Classics			Reference			Seasonal			Grand Total		
Loans	Fees	Overdue	Loans	Fees	Overdue	Loans	Fees	Overdue	Loans	Fees	Overdue	Loans	Fees	Overdue
			1	$0.00	1							1	$0.00	1
			3	$5.00	2							3	$5.00	2
1	$0.00	0	2	$0.00	2	3	$0.00	3	1	$0.00	1	7	$0.00	6
1	$0.00	0	6	$5.00	5	3	$0.00	3	1	$0.00	1	11	$5.00	9
1	$0.00	0	6	$5.00	5	3	$0.00	3	1	$0.00	1	11	$5.00	9
1	$0.00	0	6	$5.00	5	3	$0.00	3	1	$0.00	1	11	$5.00	9
1	$0.00	0	6	$5.00	5	3	$0.00	3	1	$0.00	1	11	$5.00	9

Figure 5-20. *The PivotTable View with a third data field*

Adding Additional Column Fields

In the same way, you can also add additional columns to either the Column or Row area. For example, instead of using the `MediaDescription` field in the Filter area, you can move it to the Column area. Select the `MediaDescription` column in the Filter area and drag it over the PivotTable View. Notice that a blue marker will appear when the mouse pointer is over an area that will accept this field. Drag it to the right of the `CategoryDescription` column, as shown in Figure 5-21.

CategoryDescription				
Childrens			Classics	
Loans	Fees	Overdue	Loans	Fees

Figure 5-21. *Dragging the `MediaDescription` field to the Column area*

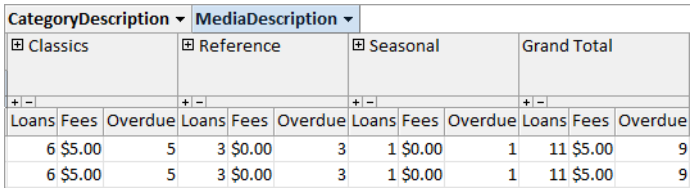
Now, instead of only displaying a single media type at a time, the `MediaDescription` field used is a sub-grouping under the `CategoryDescription` field, as demonstrated in Figure 5-22.

CategoryDescription	MediaDescription											
⊟ Classics												
DVD Video			Hardback Book			Paperback Book			Total			
Loans	Fees	Overdue	Loans	Fees	Overdue	Loans	Fees	Overdue	Loans	Fees	Overdue	
2	$0.00	2	3	$5.00	2	1	$0.00	1	6	$5.00	5	
2	$0.00	2	3	$5.00	2	1	$0.00	1	6	$5.00	5	

Figure 5-22. *`MediaDescription` as a sub-grouping under `CategoryDescription`*

For the Classics category, the PivotTable View now shows the total loans, fees, and overdue items for each media type. It also displays a total across all media types. You can also collapse each of the categories to only display the total, as shown in Figure 5-23.

www.it-ebooks.info



CategoryDescription	MediaDescription										
⊞ Classics			⊞ Reference			⊞ Seasonal			Grand Total		
Loans	Fees	Overdue	Loans	Fees	Overdue	Loans	Fees	Overdue	Loans	Fees	Overdue
6	$5.00	5	3	$0.00	3	1	$0.00	1	11	$5.00	9
6	$5.00	5	3	$0.00	3	1	$0.00	1	11	$5.00	9

Figure 5-23. *Collapsing each of the categories*

This hierarchy of attributes functions much like the `CheckedOut By Month` field that you added to the Row area. You can collapse or expand specific values to drill down into the data. Using this approach, you can add any number of fields to the Column or Row areas

---

■ **Tip** You added the `MediaDescription` field to the right of the `CategoryDescription` field. This caused the media type to be a secondary level under category. If you had dragged it to the left the `CategoryDescription`, then the media type would be the top-level in the hierarchy.

Creating a Field Hierarchy

When planning these multi-level groupings, you should try to use attributes that are logically related. The date field is the perfect example. Months and weeks are logical subdivisions of a year as are days, hours and minutes. So it is logical to drill down from year to month to day. However, categories and media types are orthogonal attributes.

When using multiple fields in a Column or Row area you have to specify the order in which the fields are used in the hierarchy. In this case the category was first so you can see how a summary for each category and then drill down to see what media type were loaned from each category. By placing media type as a sub-group under category, you cannot view a media type across all categories.

Someone may prefer to see how a specific media type was represented in each category. You cannot accomplish both at the same time. When you have a situation like this where you need to summarize in both directions, then you should put one attribute in the Row area and the other in the Column area.

A better design would be to move the `CheckedOut By Month` field to the Filter area and move the `MediaDescription` field to the Row area. Make those adjustments in your PivotTable View and it should look like Figure 5-24.

www.it-ebooks.info

CheckedOut By Month															
All															
	CategoryDescription														
	Childrens			Classics			Reference			Seasonal			Grand Total		
MediaDescription	Loans	Fees	Overdue	Loans	Fees	Overdue	Loans	Fees	Overdue	Loans	Fees	Overdue	Loans	Fees	Overdue
DVD Video				2	$0.00	2				1	$0.00	1	3	$0.00	3
Hardback Book	1	$0.00	0	3	$5.00	2							4	$5.00	2
Paperback Book				1	$0.00	1	3	$0.00	3				4	$0.00	4
Grand Total	1	$0.00	0	6	$5.00	5	3	$0.00	3	1	$0.00	1	11	$5.00	9

Figure 5-24*. Rearranging the fields in the PivotTable View*

Notice that you can simultaneously see both category and media type summaries, as well as how the other attribute was represented in the total. Click the dropdown button next to the `CheckedOut By Month` field in the Filter area. The dialog box shown in Figure 5-25 will appear which you can use to filter the view to include only specific months or days, for example.

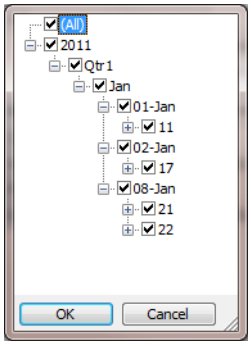


Figure 5-25*. Selecting the dates to include*

Remember, the Filter field(s) define the set of records that are included in the PivotTable View. By placing the CheckedOut By Month field here, you can control the time period that you want to analyze. Date fields are often a good candidate for a Filter field. However, if you want to see an attribute tracked over time, you will generally use the appropriate date field in Row area.

In the current database schema, there is a single-level categorization of items. For a large library you would probably want multiple levels, such as the Dewey Decimal system. The Dewey Decimal system defines three main levels of organization, which are referred to as *classes*, *divisions*, and *sections*. It is called a decimal system because there are 10 classes that each has 10 divisions, which, in turn, has 10 sections each. Thus there are 100 divisions and 1000 sections.

If you were to implement a structure such as this, then this would be an excellent candidate for using a multi-level field hierarchy in your PivotTable View. Look for other fields that logically fit into a hierarchy. The `Author` and `Title` fields are another good example.

www.it-ebooks.info

Using the PivotChart View

Once you have defined a PivotTable View, you can easily turn this into a graphical presentation. Select the PivotChart View button from the lower right corner or select the *PivotChart View* link from the View button in the ribbon. The default settings will probably look something like Figure 5-26.

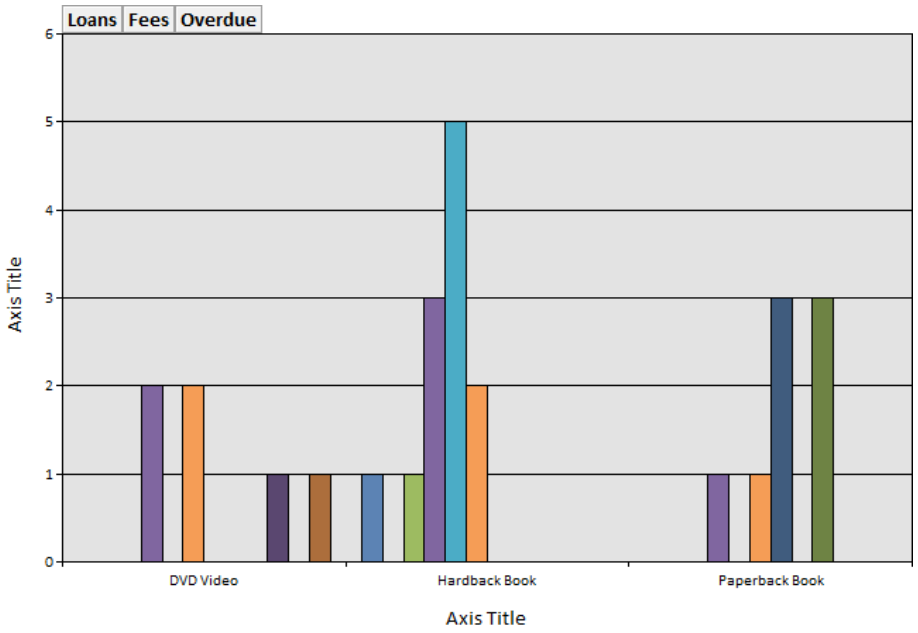


Figure 5-26. *The initial PivotChart View*

Configuring a PivotChart View

This is a fairly complex PivotTable with several values to chart. Each value is displayed as a bar in this bar graph. To see what each bar represents, you'll need to display the legend. Click the Legend button in the Design tab of the ribbon. The legend will look similar to Figure 5-27.

www.it-ebooks.info



Figure 5-27. *The PivotChart legend*

The problem with this chart is that it is trying to display too many values. In a bar chart, the Column field(s) are used to define the bar legend (a color for each column) and the Row field(s) are used for the X axis. Click the "Switch Row/Column" button, which will transpose the Row and Column fields. The resulting chart should now look like Figure 5-28.

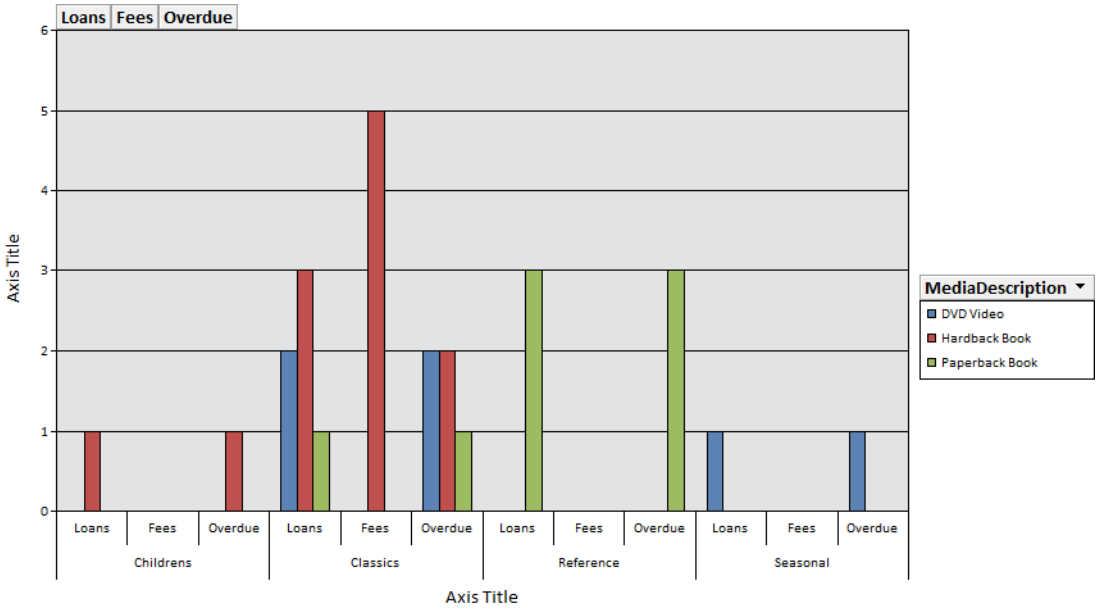


Figure 5-28. *Switching the Row and Column fields*

This is a little bit easier to follow. For the Classics category, for example, you can now see how each media type contributes to the summary totals `Loans`, `Fees`, and `Overdue`. Notice the dropdown button next to the `CategoryDescription` and `MediaDescription` fields. You can use these to filter the records that

www.it-ebooks.info

are included in the chart. Click the `CategoryDescription` dropdown and unselect all of the categories except Classics, as shown in Figure 5-29.

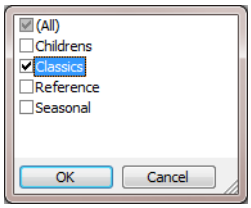


Figure 5-29. *Selecting only the Classics category*

Changing the Chart Type

With the data limited to a single category, you may want to use a different type of chart. Click the "Change Chart Type" button in the ribbon. Select the Line type shown in Figure 5-30.

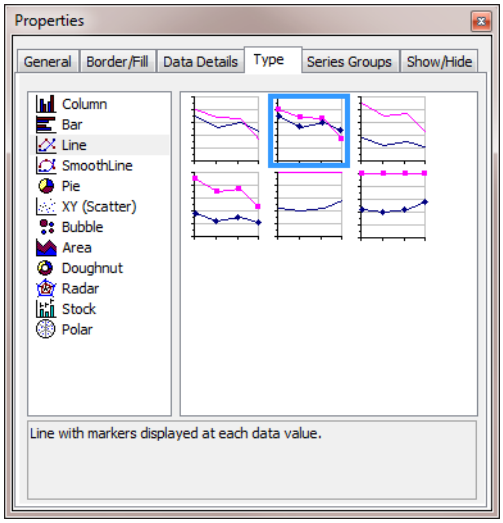


Figure 5-30. *Selecting a Line chart*

The resulting chart is shown in Figure 5-31.

www.it-ebooks.info

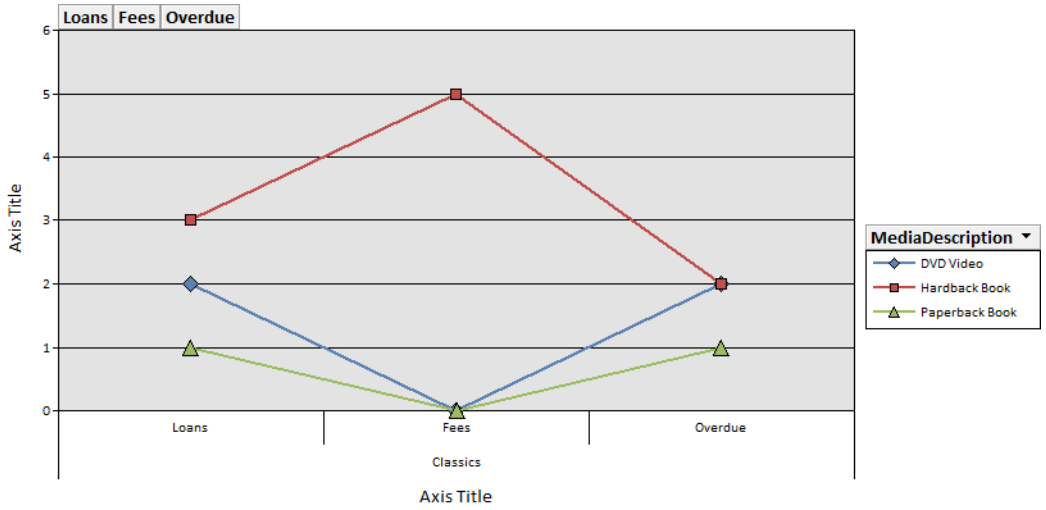


Figure 5-31*. The PivotChart View using a line chart*

You can see from this chart that overdue fees have been charged for Hardback books, but no fees have been charged for DVD videos or paperback books even though there have been overdue items of these media types.

■ **Note** You will likely have different values in your database. My purpose here is to give you an example of how to read the chart.

Now display the PivotTable View by selecting the PivotTable link from the View button. Notice that this view now looks very different from where you left it. This illustrates a very important point: The PivotTable View and the PivotChart View use the same Row, Column, and Filter area definitions. If you change this configuration, it is automatically changed in the other, as well. More subtle and perhaps more significant, you can only have one PivotTable/PivotChart View for each table or query.

■ **Caution** You can define only one PivotTable/PivotChart View for each table or query. The PivotTable and PivotChart views share the same configuration; when you change one, the other is also updated.

---

www.it-ebooks.info

Exporting a PivotTable View to Excel

Access 2010 allows you to export a PivotTable or PivotChart view to Excel. This will allow you to manipulate the data and view using the Excel application. As I mentioned earlier, the concepts are similar but the implementation is different.

From the PivotTable View, click the `CategoryDescription` dropdown and select all categories. Save the database and then click the "Export to Excel" button in the Design tab of the ribbon. You will see a pop-up window warning you about possible compatibility issues, shown in Figure 5-32. Click the OK button.

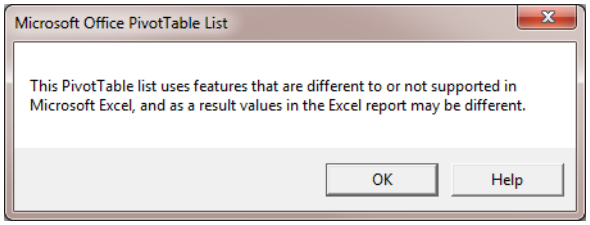


Figure 5-32. *Compatibility warning*

Access will then open the Excel application and create a PivotTable worksheet, as shown in Figure 5-33. Notice the similarities to Access 2010 and the differences.

	A	B	C	D	E	F	G	H	I	J
1	Years	(All)								
2	Quarters	(All)								
3	Months	(All)								
4	Days	(All)								
5	Hours	(All)								
6	Minutes	(All)								
7										
8		CategoryDescription	Data							
9		Childrens			Classics			Reference		
10	MediaDescription	Loans	Fees	Overdue1	Loans	Fees	Overdue1	Loans	Fees	Overdue1
11	DVD Video				2	$0.00	2			
12	Hardback Book	1	$0.00	1	3	$5.00	2			
13	Paperback Book				1	$0.00	1	3	$0.00	3
14	Grand Total	1	$0.00	1	6	$5.00	5	3	$0.00	3

Figure 5-33. *The PivotTable View in Excel*

The most obvious difference is that the multi-field hierarchy is not as evident. Notice that the `CheckedOut By Month` field is replaced by a set of fields, one for each of the data components such as Year, Quarter and Month.

The raw data is also imported in the second worksheet, as shown in Figure 5-34.

www.it-ebooks.info

	A	B	C	D	E	F	G	
1	LoanID	CheckedOut	DueDate	OverdueFee	Title	Author	ReplacementCost	Cat
2	1	1/1/2011 11:14	12/27/2010	0	A Tale of Two Cities	Charles Dickens	7.99	CLA
3	3	1/2/2011 17:24	1/23/2011	0	A Christmas Carol	Charles Dickens	11.95	CLA
4	4	1/2/2011 17:27	12/23/2010	2.5	A Christmas Carol	Charles Dickens	11.95	CLA
5	5	1/2/2011 17:28	12/23/2010	2.5	A Christmas Carol	Charles Dickens	11.95	CLA
6	6	1/8/2011 21:51	1/15/2011	0	It's a Wonderful Life	Frank Capra	8.75	SEA
7	7	1/8/2011 22:01	1/15/2011	0	Hamlet	William Shakespear	15.95	CLA
8	8	1/8/2011 22:02	1/15/2011	0	Hamlet	William Shakespear	15.95	CLA
9	9	1/8/2011 22:02	1/22/2011	0	Roget's Thesaurus	Peter Mark Roget	9.5	REF
10	10	1/8/2011 22:02	1/29/2011	0	Fun with Dick and Jane	Gray and Sharo	12	CHI
11	11	1/8/2011 22:02	1/22/2011	0	Roget's Thesaurus	Peter Mark Roget	9.5	REF
12	12	1/8/2011 22:02	1/22/2011	0	Roget's Thesaurus	Peter Mark Roget	9.5	REF

Figure 5-34. *The underlying Excel data*

Excel uses the raw data that the PivotTable is based on. In order for the PivotTable to work, all of the raw data must be imported, as well. You can use this approach create a new PivotTable and/or chart in Excel using the data from Access and the PivotTable features from Excel.

■ **Caution** When you export data to Excel, this is a one-time operation. Subsequent changes to the data in Access are *not* pushed to Excel using manual or automatic means. The only way to update Excel is to re-import the data from Access.

Summary

PivotTables and charts are an excellent way to present and analyze data from your Access database. You configure a PivotTable or chart by adding one or more fields from the underlying table to each of the following areas:

- Data area: This contains the values that ate presented and normally use an aggregate function to summarize the data.
- Row and Column: Fields in these areas define the attributes that are used for slicing and dicing the data.
- Filter: (Optional) Fields in the Filter are used to limit the data that is being presented.

If you specify multiple Row or Column fields, they are added in hierarchical fashion allowing you to drill down from one attribute to the next. The built-in By Month and By Week date fields are an excellent example of this.

The most significant limitation to keep in mind is that you can only have one PivotTable/PivotChart View for each table or query. The PivotTable and PivotChart views share the same configuration values; when you change one, both are updated. If you want to create additional PivotTables, refer to Chapter 4 for instructions on how to create a query that joins several tables.

You can also export the data and PivotTable to Excel and manipulate the presentation using the Excel tools.

www.it-ebooks.info

PART 3

Creating Forms and Reports

In Part 2, you created your database, designed the tables, and wrote data macros to implement many of the business rules. You created queries to provide de-normalized views into your data, and even designed a pivot table to analyze the data that is being collected. Although still a bare-bones solution, your database fulfils all the basics requirements. In Part 3, however, you'll put flesh on those bones and create a rich user experience.

Chapter 6 will show you how to use the built-in form templates to easily generate many of the simpler forms. Chapters 7, 8, and 9 demonstrate how to build more complex forms from scratch. This will demonstrate a lot of handy tricks for creating useful forms. Chapter 10 will show you how to create menus and navigation pages that will guide the user to the provided features. In Chapter 11, you'll focus on the visual, branding aspects of your application, including themes, graphics, and background images.

Finally, in Chapter 12, I'll show how to create reports, which are essentially forms that are designed for print output.

www.it-ebooks.info

CHAPTER 6

Standard Forms

One of the really great features of Access is that it can generate forms for you based on your table design. This is another good reason for starting with a well thought out data schema. In this chapter, I'll demonstrate several common form patterns that will satisfy many of your UI requirements. These are created using standard form templates or the Form Wizard, and do not require writing any code.

This chapter will also provide a foundation that applies to both these simple forms as well as advanced custom forms. In subsequent chapters, I'll show you how to design forms yourself and use macros and VBA code to implement more complex solutions. These custom forms are based on the same general principles that I will explain in this chapter.

Creating a Single Form

We'll start by creating a simple form to display records in the `Category` table.

Using the Form Wizard

From the Create tab of the ribbon, click the Form Wizard button. In the first dialog box of the Form Wizard, select the `Category` table, and then move the `CategoryCode` and `CategoryDescription` fields to the Selected Fields list, as shown in Figure 6-1.

www.it-ebooks.info

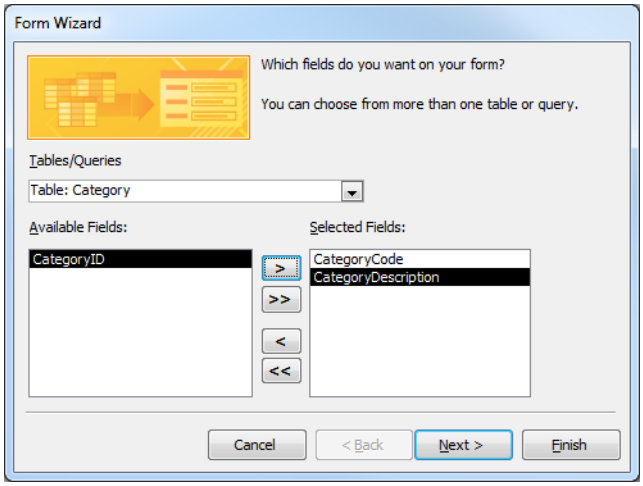


Figure 6-1. *Selecting the record source for the form*

Each form is based on a single table or query, and the first step in designing a new form is to specify which will be the source for this form. After selecting a table or query from the dropdown list, all the available fields will display. You can then select all of the fields to be included on the form or just a subset of them.

■ **Caution** If your form will be used to add records, you should generally use a table for the source, rather than a query. Queries usually supply only a subset of fields or rows and often use multiple tables. All of these characteristics are problematic when inserting records. That is not to say that you cannot use a query; however, if you use a query for a form that allows new records, make sure that every required field is included on the form or has a default value assigned. Otherwise, the form will not be able to save a new record.

Notice that the primary key was not included in this form; this is a common design practice. As I discussed in Chapter 2, the primary key is a surrogate key generated by the database engine to ensure uniqueness. In many cases it is not meaningful to the end user, so there is no need to display it in that scenario.

In the second dialog, you'll need to choose how you want the fields organized on the form. As shown in Figure 6-2, there are four layout options available. I will explain these choices later in this chapter. Because this form only has two fields on it, just leave the default option of Columnar and click the Next button.

www.it-ebooks.info

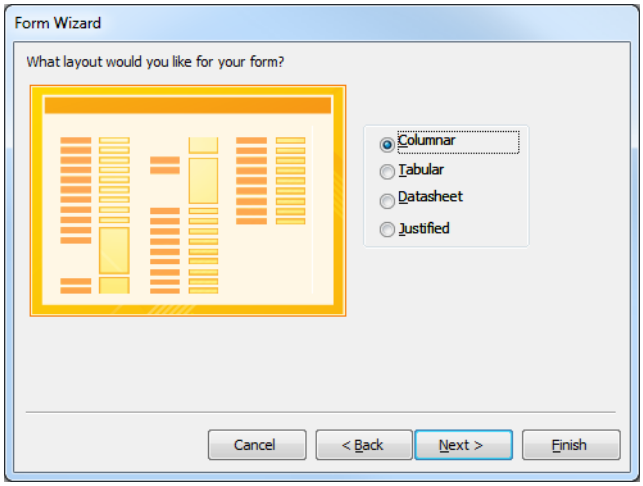


Figure 6-2. *Choosing the desired layout option*

In the final dialog box, enter **Category** for the title of this form. For the radio options, select the first option, "Open the form to view or enter information," as shown in Figure 6-3.

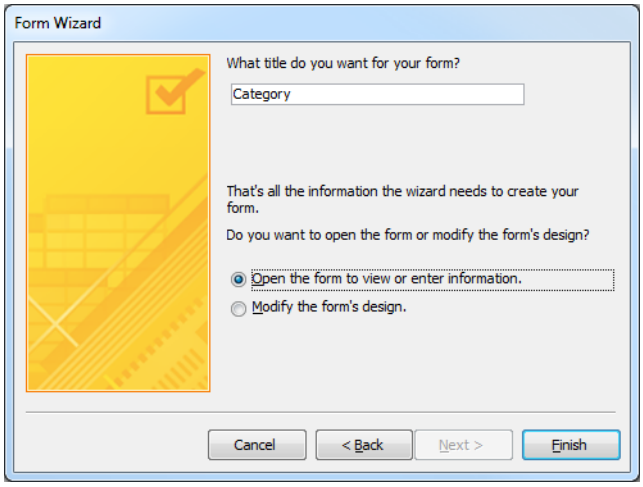


Figure 6-3. *Specifying the form name*

The new form should look like Figure 6-4.

www.it-ebooks.info

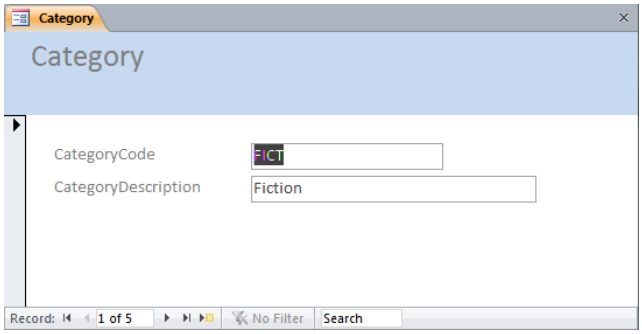


Figure 6-4. *The initial* `Category` *form*

In this simple form, a single record is shown. Notice the record navigator at the bottom of the form. You can use this to move to the first, previous, next, or last record in the table. The last control in this group will display a blank record for adding a new category.

Using the Available Views

Just like with tables and queries there are several views available when working with a form. To display all the view options, perform the following steps:

1. Click the Design View button in the ribbon. If the Property Sheet is not currently visible, click the Property Sheet in the Design tab of the ribbon. There is a dropdown list at the top of the Property Sheet that you can use to select the object that you want to view. You can select the Form, any of the sections such as Form Header or Detail, or one of the individual controls. Because there are a lot of properties, these are grouped into separate tabs.
2. Select the Form object and the Format tab. There are several properties that control what views are allowed for this form. The Form Wizard generates the form with the Datasheet View turned off. Change this value to Yes as shown in Figure 6-5.

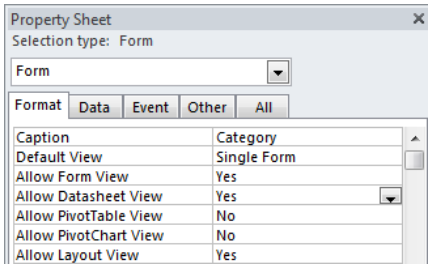


Figure 6-5. *Allowing the Datasheet View*

www.it-ebooks.info

Notice that the View options in the ribbon now has four options, which are shown in Figure 6-6.

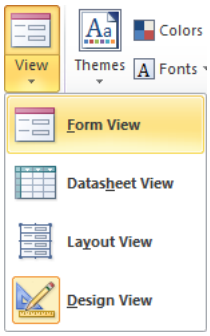


Figure 6-6. *The allowed views*

Every form usually has two modes that it can be viewed in; Form View and Datasheet View. You've already seen the Form View. In this view, a single record is displayed and a record navigator control is used to move through the available records. Each field is represented by an appropriate data-bound control, such as Text Box, Check Box, or ComboBox depending on the type of data contained in the field. In addition, Label controls are used to annotate what each data control is for. All of these controls are arranged on the form.

Select the Datasheet View from the ribbon, which should look like Figure 6-7.

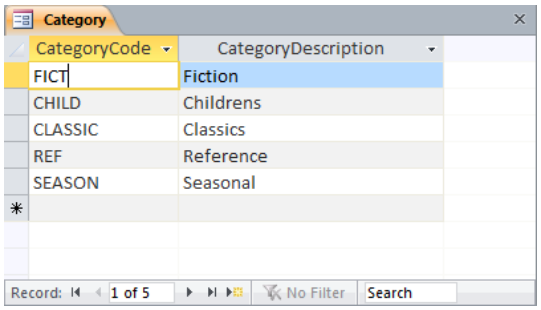


Figure 6-7. *Displaying the Datasheet View*

The Datasheet View of a form looks very much like the table when it is displayed in the Datasheet View. In this view, you can re-order the columns by selecting one and then dragging it to the desired position. You can change the width of the column by clicking the gridline and dragging it left or right. The column headings are defined by the Datasheet Caption property. Let's change them now, with the following steps:

1. Select the `CategoryCode` column and then, in the Property Sheet, select the Other tab.

www.it-ebooks.info

2. The default value of the Datasheet Caption property is blank. When this is the case, the heading text is defined by the Control Source property, which is the associated column of the underlying table. The Caption that was defined for that column will be used as the column heading. Enter **Code** for the Datasheet Caption property.
3. Likewise, for the `CategoryDescription` column, enter **Description** for the Datasheet Caption property.

■ **Note** The configuration of the Datasheet View is independent of the Form View. Changing the order of the columns, the column widths, or the column headings has no effect on the layout of the Form View. If you allow both views of your form, you should check the layout of the Datasheet View and make sure it looks like you want it to.

The Layout View and Design View are used to design and modify the form. You will use these views extensively in subsequent chapters as you build custom forms. The Design View allows you configure all aspects of the form. The Layout View is an interesting and useful view. It looks like the Form view and displays the fields just like the Form View would, but it also allows you to add and rearrange controls. It is essentially a WYSIWYG editor.

Sorting the Records

To sort the records, click the dropdown icon next to the Description column and click the Sort A to Z link, as shown in Figure 6-8.

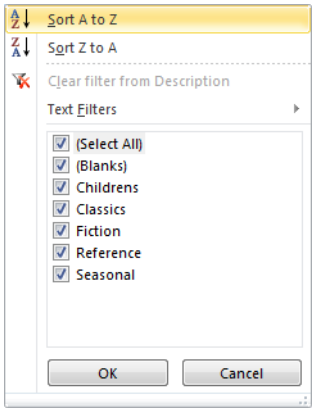


Figure 6-8*. Sorting the Datasheet View*

www.it-ebooks.info

This will display the records in alphabetical order based on their descriptions. This change is reflected on both views. You can verify this by switching to the Form View; the records should be in the same order as in the Datasheet View. Go to the Design View and select the Data tab of the Property Sheet. Notice the Order By property has been populated, as shown in Figure 6-9.

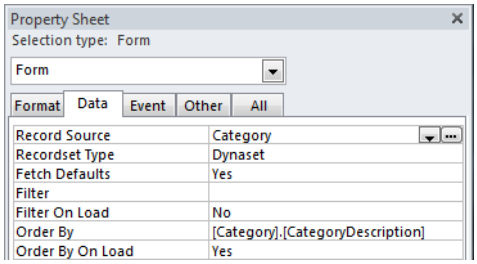


Figure 6-9. *The Order By property*

Using Split Forms

All forms are based on these two display modes, Form View and Datasheet View. However, there are also two variations of these that are a composite of both views. The first is called Split Form, which shows both views simultaneously. The other is a Continuous Form, which combines features from both the Datasheet and Form views.

The Split Form is a really useful feature, especially for smaller tables. It enables you to see all the records in a compact Datasheet View. At the same time, a single record can be viewed and modified in the more user-friendly Form View. A good example would be for longer text fields. The Datasheet View will probably only show the first few words but the Form View can display the entire text on multiple lines, with scrollbars if necessary.

Generating the Media Form

You'll now create a Split View for the Media table with the following steps:

1. Close the `Category` form and any other tabs that may be open.
2. Select the `Media` table in the Navigation pane.
3. From the Create tab in the ribbon, click the More Forms dropdown and then click the *Split Form* link, as shown in Figure 6-10.

www.it-ebooks.info

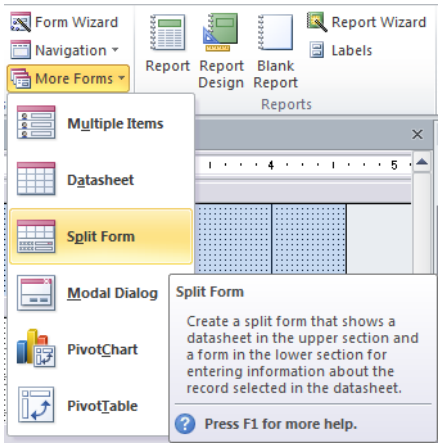


Figure 6-10. *Creating a Split Form*

This will generate a form based on the current table, which should look like Figure 6-11.

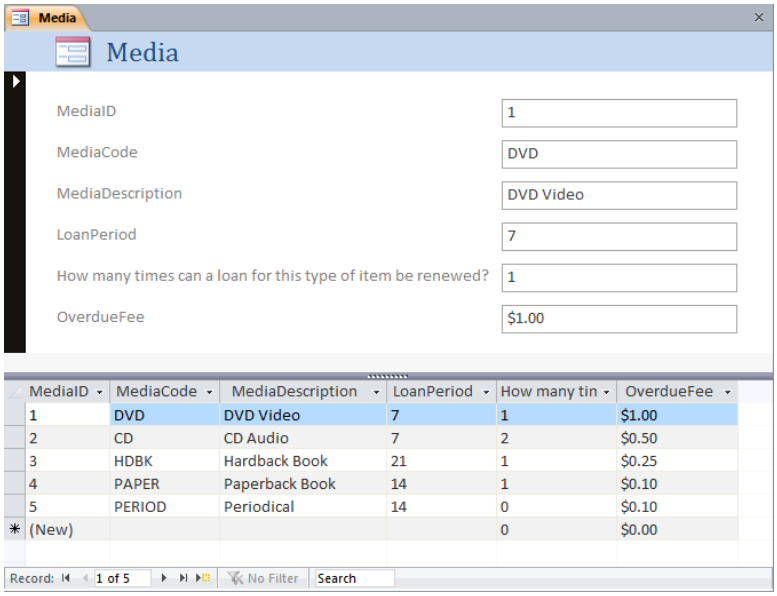


Figure 6-11. *The initial Split Form*

www.it-ebooks.info

■ **Tip** Creating a form using the Split Form button does not start the Form Wizard, which allows you to configure how the form will be generated. Instead it creates a form based on the currently selected table and includes all the available fields. You need to make sure you have the correct table selected in the Navigation pane before using this option. If you create a form from the wrong table, just delete it and try again.

Modifying the Form Fields

The form was generated using all the fields. You'll need to remove the `MediaID` field from both forms; to do so, right-click the `MediaID` column in the Datasheet View (the lower portion of the form) and click the *Delete* link. This will also remove this column from the Form View.

Just like with the previous form, you configure the layout of both the Form View and Datasheet View separately. We'll start with the Form View.

1. The label for the `RenewalsAllowed` field is a little long (How many times can a loan for this type of item be renewed?). To make this form look better, you'll use the Layout View to resize the controls. From the Design tab of the ribbon, click the Layout View button.
2. Select the label control for the `RenewalsAllowed` field, which will highlight the control with an orange border.
3. Resize this control so it is about half as wide and twice as high. The form should look like Figure 6-12.

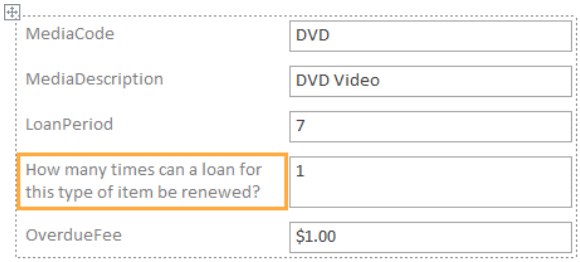


Figure 6-12. *Resizing the label control*

4. Now adjust the column heading in the Datasheet. To do that, use the Property Sheet and select the `RenewalsAllowed` control.
5. In the Other tab, enter **Renewals** for the Datasheet Caption property.
6. Save the form and select the default form name `Media` when prompted.

Go back to the Form View to view the final version, which is shown in Figure 6-13.

www.it-ebooks.info

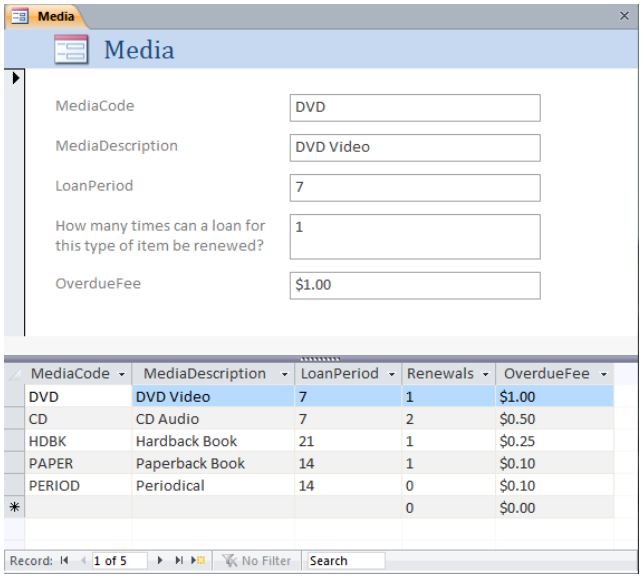


Figure 6-13. *The final* `Media` *form layout*

■ **Tip** If you remove a field from one of the views, it is automatically removed from both. This happened, for example, when you removed the `MediaID` field. Normally, both views will include the same fields. However, there are a couple of ways around this if you want to have different fields in each view. To remove a field from the Form View only, use the Property Sheet to set the Visible property to No. The field will still be included in the Datasheet View, but both the field and its associated label will be hidden from the Form View. To remove a field from only the Datasheet View, simply resize the column to a 0 width.

Figure 6-14 shows the form with `OverdueFee` field removed from the Form View and the `RenewalsAllowed` field from the Datasheet View.

www.it-ebooks.info

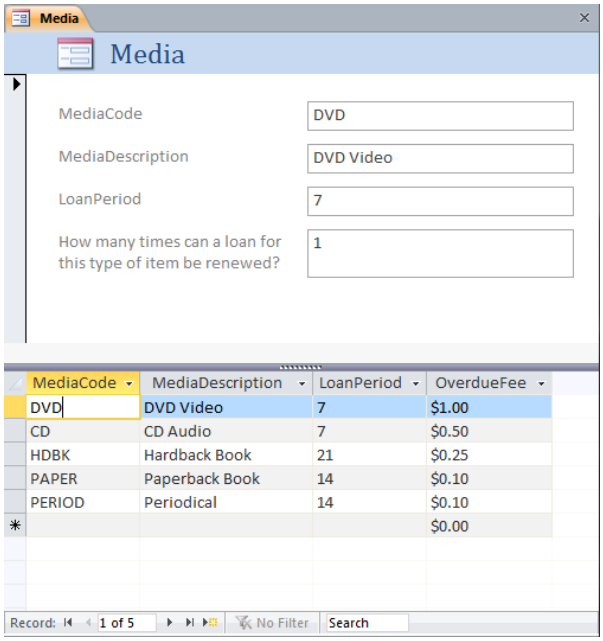


Figure 6-14. Updated form with fields removed

■ **Tip** In the Format tab of the Property Sheet, you can set the Split Form Orientation property. The default value is Datasheet On Top, but you can change this to put the Datasheet View on the top if you prefer. You can also put it on the right or left.

Using Continuous Forms

The other variation is called a Continuous Form. It's like a Datasheet View in that all records are displayed sequentially. However, each record is displayed as a collection of controls just like the Form View. It is basically the Form View, except instead of only displaying a single record, the form controls are repeated over and over again. Obviously, this won't work very well for long forms, but it can be useful for relatively short forms.

Generating the InventoryItem Form

You create a Continuous Form just like the Form View that you created at the beginning of the chapter. Then you have to the change the Default View property to Continuous Form. You'll now create a Continuous Form for the `InventoryItem` table following these steps:

www.it-ebooks.info

1. From the Create tab of the ribbon, click the Form Wizard button. In the first dialog box, select the `InventoryItem` table and include all the available fields as shown in Figure 6-15.

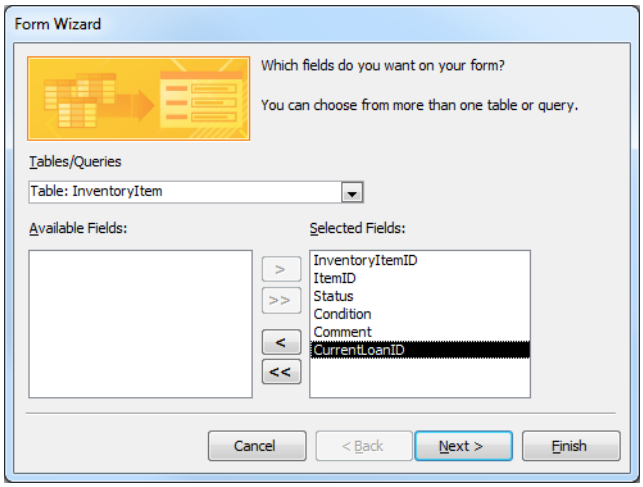


Figure 6-15. *Selecting the* `InventoryItem` *table*

2. In the second dialog box, select the Tabular layout, as shown in Figure 6-16. The Tabular layout is the best choice if you're planning to use a Continuous Form. It was designed specifically for that purpose.

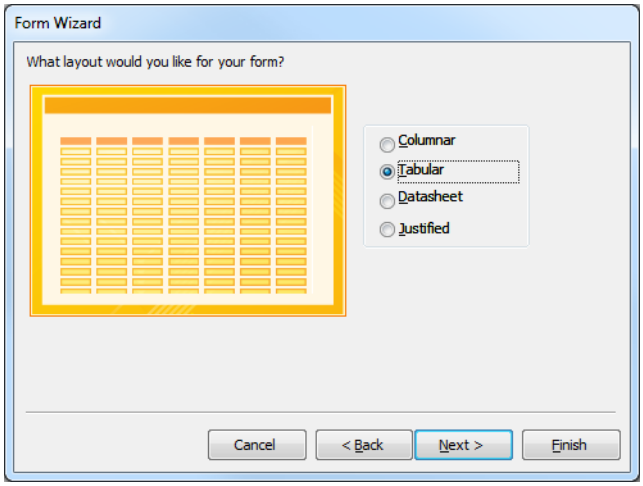


Figure 6-16. *Using the Tabular layout*

www.it-ebooks.info

3. In the final dialog box, leave the default form title as **InventoryItem**, as shown in Figure 6-17. Select the second radio option, which is to modify the form's design. This will cause the new form to be opened in the Design View instead of the Form View.

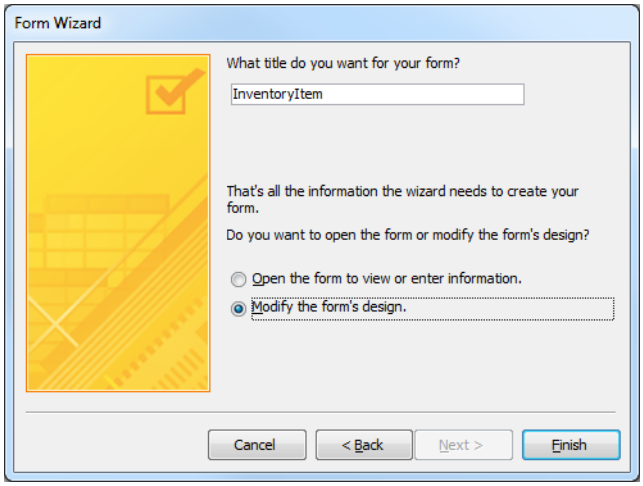


Figure 6-17. *Specifying the form title*

The form should now be displayed in the Design View and look similar to Figure 6-18.

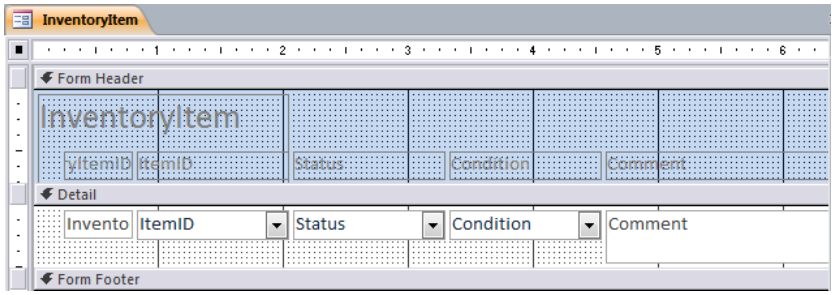


Figure 6-18. *The initial form in Design View*

The first thing you'll probably noticed is that the Label controls are in the Form Header rather that the Detail section. In a Continuous Form, only the Detail section is repeated for each record. The Form Header and Form Footer are only displayed once. Typically you will want to put the labels in the header to save "space" in the repeated section. This will help you keep the Detail as thin as possible. However, you don't have to do this; you can also put some or all of the labels in the Detail section if you want.

www.it-ebooks.info

■ **Tip** You cannot drag a control from one section of a form to another. If you want to move a label from the Form Header section to the Detail section, right-click the control and click the *Cut* link. Then right-click in a blank area of the Detail section and click the *Paste* link.

Also, the default layout has all of the fields on a single row, much like a datasheet. Again, you don't have to keep them that way; you can arrange them in any way you want to. In general, however, you will want to keep the Detail section thin, especially if you expect numerous records to be included.

USING THE MULTIPLE ITEMS TEMPLATE

Access often provides multiple ways to accomplish the same thing. Instead of using the Form Wizard, you could have selected the `InventoryItem` table in the Navigation pane and then click the Multiple Items button in the Create tab of the ribbon, as shown in Figure 6-19.

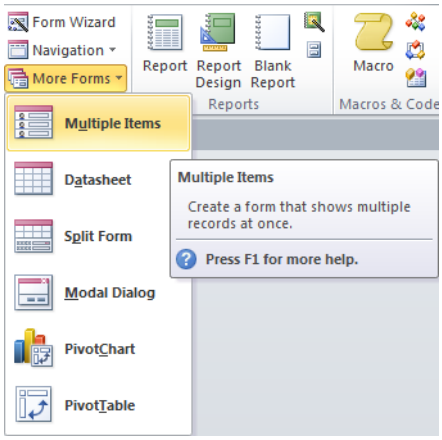


Figure 6-19. *Using the Multiple Items button to create a form*

This would create a new form that is very similar to the one created through the Form Wizard.

Designing the InventoryItem Form

It is helpful to think through how a form will be used; specifically, is it for view only or will it be used to add or update records. The first two forms that you created were designed to view, add, change, and delete records from the `Category` and `Media` tables. The `InventoryItem` form will be primarily used for viewing but will allow some restricted updates.

In Chapter 4, you created an append query that is used to insert a record into the `InventoryItem` table. This query takes a single parameter, the `ItemID`, and all the other fields will have default values.

www.it-ebooks.info

You will eventually add a button to an Item form that will execute this query to add a copy of that item to the inventory. Also, the ItemID should not be editable once the record is created. Likewise, the Status field is controlled by the data macros that you implemented in Chapter 3. The only user-modifiable fields on the form are Condition and Comment.

This InventoryItem form will be used as a subform, which I will explain later in this chapter. It will be used to show the InventoryItem records for a specific item. Since the associated item is inferred by the context in which this subform is placed, you can remove the ItemID from this form. Also, the CurrentLoanID field is not necessary for this purpose. The Status field was generated as a ComboBox control to allow the desired value to be selected. Because this should not be editable, you'll want to replace this control with a TextBox control. This will display the current value without implying to the user that it can be modified.

Modifying the Form Fields

From the Design View, make the following changes:

1. Delete the ItemID ComboBox control. Notice that its associated label was not also removed. Because they are in different sections, they are not linked together. You'll need to manually delete the Label control as well.
2. Delete the Status ComboBox control (leave the label).
3. From the Design tab of the ribbon, click the TextBox button. Then click in the Detail section, holding the mouse button down and dragging it to form a rectangle in the same location and about the same size as the previous dropdown list control. In the Other tab of the Property Sheet, enter **Status** for the Name property. In the Data tab, select Status from the Control Source property.
4. The new Textbox control will have added an associated Label control. Delete this, as it is not needed.
5. On both the InventoryItemID and Status controls, set the Locked property to Yes. You can find this in the Data tab of the Property Sheet.
6. Change the Caption property of the InventoryItemID label to **ID** and set the Text Align property to Left.
7. Remove the CurrentLoanID ComboBox control as well as its associated label.
8. Rearrange the controls in both the Detail and Form Header section to remove the empty spaces.

The form in Design View should look similar to Figure 6-20.

www.it-ebooks.info

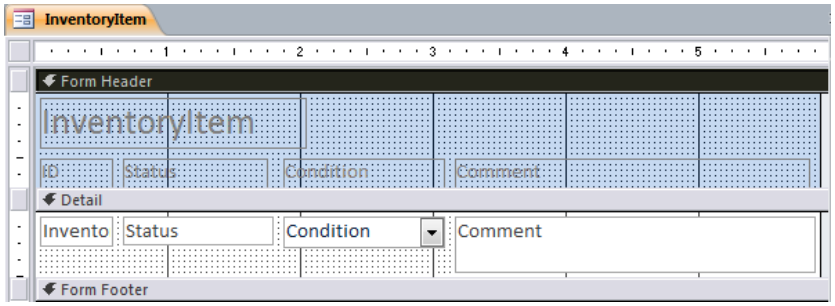


Figure 6-20. *The completed Design View*

Select the Form object in the Property Sheet and then select the Format tab. Notice that the Default View property is already set to Continuous Form. This is set by the Form Wizard whenever you use the Tabular layout.

■ **Note** You can change the Default View to Single Form if you want to see what the form would look like as a Single Form. This does not affect the design or layout of the form. Make sure you change it back to Continuous Form if you do change the Default View.

Save the form changes and select the Form View from the ribbon. The final form should look like Figure 6-21.

www.it-ebooks.info

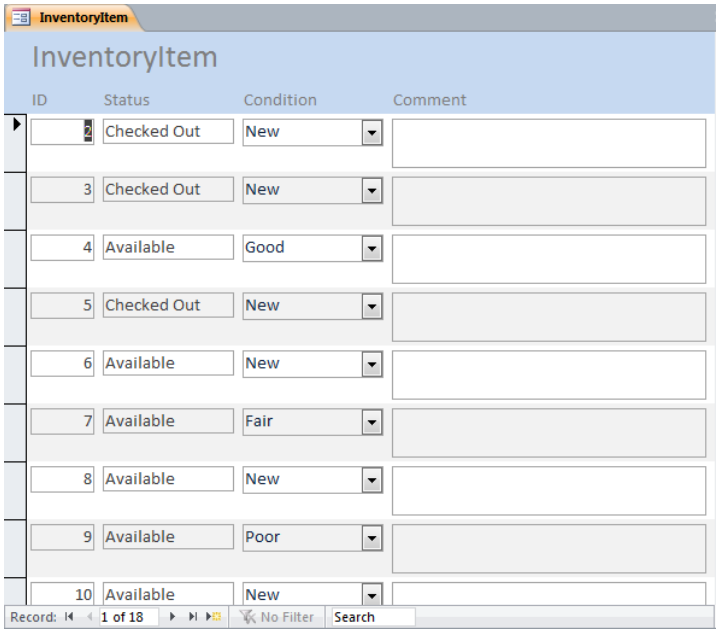


Figure 6-21. *The final* `InventoryItem` *form*

Understanding the Layout Options

Before I explain the final form that you'll create in this chapter, I want to review the layout options that are provided by the Form Wizard. You use the first dialog box to define the data source for the form. Each standard form uses a single table or query. You specify which one to use for the form and then decide which fields of that table or query to be included.

You use the second dialog box to indicate one of four layout options, which are:

- Columnar
- Tabular
- Datasheet
- Justified

When you select the associated radio button, the image changes to give you a visual representation of how your form will look.

Using the Tabular Layout Option

As I just demonstrated, the Tabular layout is used for generating a Continuous Form. The labels are placed in the Form Header and the data bound controls are aligned horizontally in a single row. This is presented in the Form Wizard with the graphic shown in Figure 6-22.

www.it-ebooks.info



Figure 6-22. *The Tabular layout*

As I mentioned earlier, Continuous Forms don't have to follow this pattern. You can include labels in the Detail section and arrange the data bound controls in any manner you wish. Also, when you chose the Tabular layout, the Default View property of the form is set to Continuous Form.

Using the Datasheet Layout Option

Similarly, the Datasheet layout should be used when you want to create a form that is viewed in the Datasheet View. The graphic shown in Figure 6-23 portrays this. The labels are at the top, like the Tabular layout, but there are also record selectors on each row.

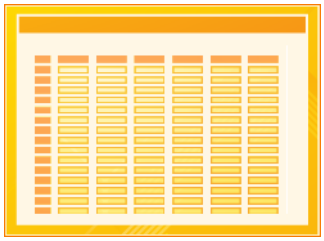


Figure 6-23. *The Datasheet layout*

When you use this option, the Default View property of the generated form is set to Datasheet View. However, you can choose to view the form using the From View. If you do, you'll notice that the form layout look just like forms generated with the Columnar layout, which I'll describe next. In fact, the Datasheet and Columnar options generate the exact same form, except that the Default View property is set to Datasheet View with the former option and Form View with the later.

Using the Columnar Layout Option

The last two layout options, Columnar and Justified are used when creating forms that will be viewed in the normal Form View. You used the Columnar layout when creating the `Category` form. This layout creates a label and its associated data-bound control side-by-side on the form. Each field is stacked vertically, which results in a column of labels and a column of controls. The image shown in Figure 6-24 represents this arrangement.

www.it-ebooks.info



Figure 6-24. *The Columnar layout*

This graphic implies that there can be multiple pairs of columns; however, the Form Wizard only creates a single pair of columns. You can arrange this into multiple columns in the Layout View using a Layout control, which I'll explain later.

Using the Justified Layout Option

The final layout choice is called Justified. It generates a label that is directly above its associated data-bound control for each field. Instead of being stacked in columns, these pairs are arranged left-to-right like text on a page. This is demonstrated in the associated graphic shown in Figure 6-25.



Figure 6-25. *The Justified layout*

In some ways this may seem like a more desirable arrangement, as well as the most efficient use of space.

■ **Caution** Keep in mind that if you plan to use a layout control, which I'll explain next, the Justified layout is not compatible. When converting to a layout, the fields will be re-arranged using Columnar format before being added to the layout.

---

www.it-ebooks.info

Using a Layout

Access provides a facility for easily arranging controls on a form that is often referred to as a layout. It is a grid that contains cells where you can insert controls. By placing labels and data-bound controls into a grid, you can easily format or resize an entire column or row and keep everything aligned properly.

You may have noticed that the controls in the `Media` form were placed in a grid. However, the `Category` and `InventoryItem` forms do not do this. These were generated using the Form Wizard, which does not use a layout. You'll fix that now.

1. Open the `Category` form using the Design View.
2. Select all the controls in the Detail section. You can do this quickly by dragging the mouse around a rectangle that includes all of the controls. Then, from the Arrange tab of the ribbon, click the Stacked button, as shown in Figure 6-26.

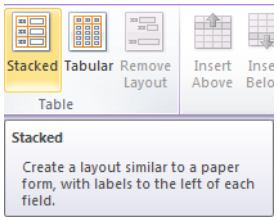


Figure 6-26. *Using the Stacked button in the Arrange tab*

3. You can switch to the Layout View and resize the grid columns, if necessary. Save the form and then switch to the Form View to see how the form looks.
4. Open the `InventoryItem` form in the Design View. In this form the labels are in the Form Header.
5. Select both the labels and the data-bound controls. For this form, click the Tabular button in the Arrange tab of the ribbon.
6. Switch to the Layout View and resize the columns. Notice that the labels in the Form Header are kept aligned with the associated control as you size each field. The final layout should look like Figure 6-27.

www.it-ebooks.info

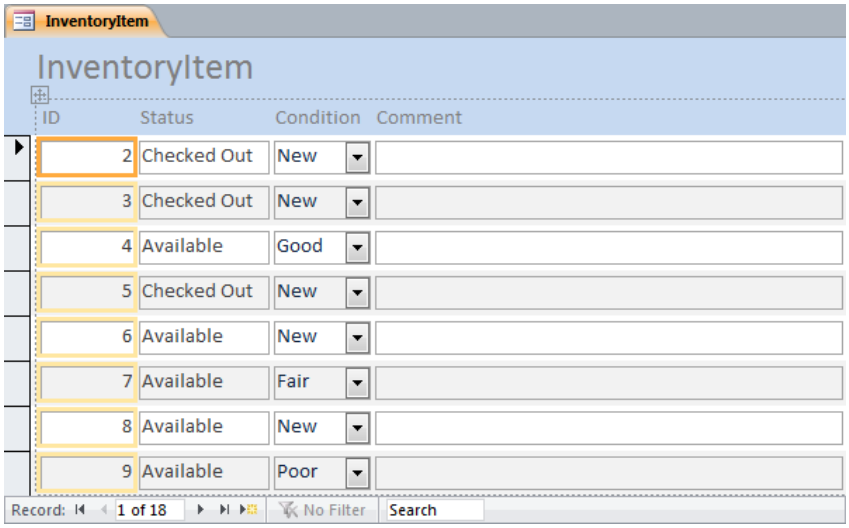


Figure 6-27. *The modified* `InventoryItem` *form layout*

7. Save the form changes, switch to the Form View, and verify the form layout.

Creating the Item Form

Now you'll add a form for the `Item` table, which will allow you to view, update, and create records in the `Item` table. You'll use the layout control to improve the arrangement of the controls. You will also imbed a sub form to view and update the associated `InventoryItem` records.

Using the Standard Form Template

Select the `Item` table in the Navigation pane and click the Form button in the Create tab of the ribbon as shown in Figure 6-28.

www.it-ebooks.info

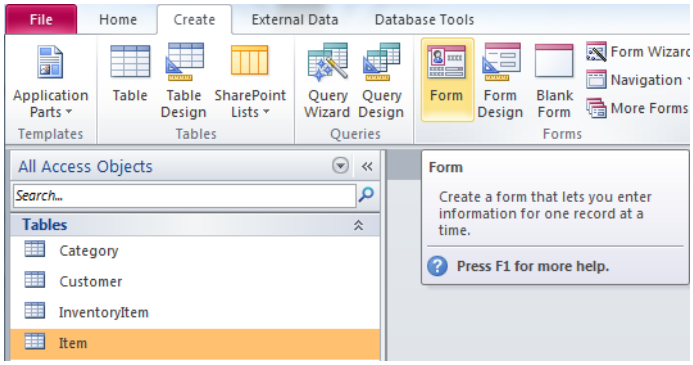


Figure 6-28. Using the Form button to create the `Item` *form*

■ **Caution** Just like with the Split Form link that you used earlier, it very important that you select the table before clicking the button. This does *not* launch a wizard where you can select the source table. With one click it will create a form based on the selected table.

This will generate a new form and display it using the Layout View that should look like Figure 6-29.

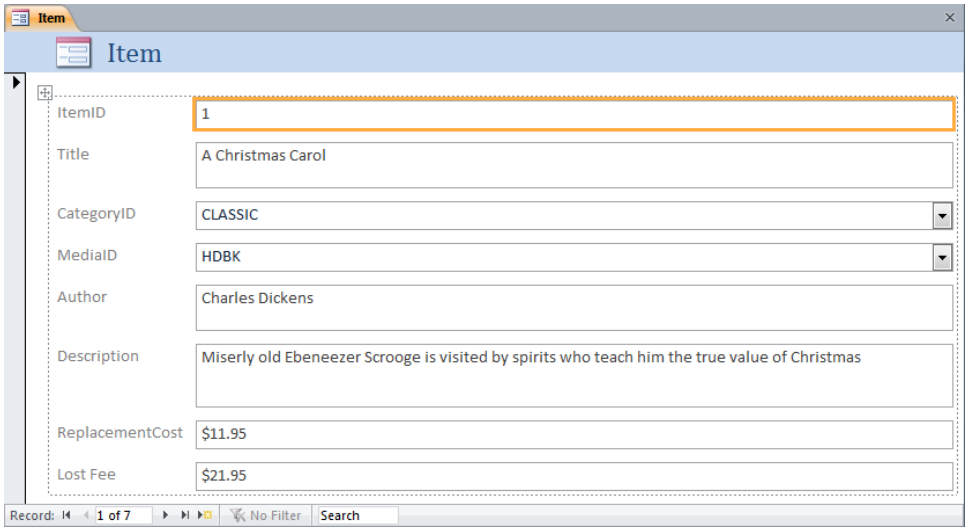


Figure 6-29. The initial `Item` *form in the Layout View*

www.it-ebooks.info

Arranging the Form's Layout

The default layout creates a column for labels and a column for the data-bound controls. Because of this, all of the controls have the same width. This is less than optimal; however, since the `ReplacementCost` does not need to be as wide as the `Description` field, for instance. I'll show you how to rearrange these controls to achieve a better presentation.

Select the Arrange tab of the ribbon. You'll notice a number of buttons, shown in Figure 6-30, that will help you modify the form's layout.

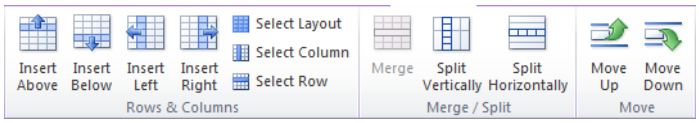


Figure 6-30. *The formatting buttons in the Arrange tab*

Using the layout control and the tools in the Arrange tab, perform the following steps:

1. The first step is to resize the second column to give you some room to work with. Select one of the controls, which will draw an orange border around it. Click on the right edge of this border and drag it to the left so the column is about an inch wide. Notice that all the other controls are resized as well.
2. Click the Insert Right button in the ribbon three times to add three more columns.
3. Select the `ItemID` control and its associated label and delete them. Notice that the row remains but the cells are now empty. To remove the row, right-click one of the empty cells and click the *Delete Row* link.
4. Select the `MediaID` control and its associated label and drag them to the previous row, just to the right of the `CategoryID` control. This will leave an empty row, which you should delete.
5. In the same way, drag the `LostFee` control and its label to the right of the `ReplacementCost` control and delete the empty row.
6. Now you'll re-order the rows, for example, the `Author` field should come before the `Title`. Select the `Author` control. You'll want to move the entire row so click the Select Row button in the ribbon. Then click the Move Up button twice. In the same way, move the `Description` field to the last row.
7. The `Author`, `Title`, and `Description` fields will need more space as I previously noted. You can accomplish this by merging multiple cells together. Select the `Author` control and two empty cells to the right of it (do not include the third empty cell). Then click the Merge button. Notice that there is now an orange border around all three cells.
8. Do the same for the `Title` control, merging a total of three cells.
9. For the `Description` control, merge all four cells together.

www.it-ebooks.info

10. You can resize some of the controls as appropriate. For example, you will probably want the `Description` control to be taller to allow for some longer descriptions. Because the Layout View displays the form with actual data, it's pretty easy to see how the data will fit into the controls. You can use the record navigation control at the bottom of the form to select different records to see how each will appear in the form.

11. Select one of the Label controls in the first column and then click the Select Column button in the ribbon. This will select the all of the labels in this column. In the Property Sheet, find the Text Align property in the Format tab and change its value to Right. Also select the `MediaID` and `LostFee` Label controls and change their Text Align property to Right.

The form should now look like Figure 6-31.

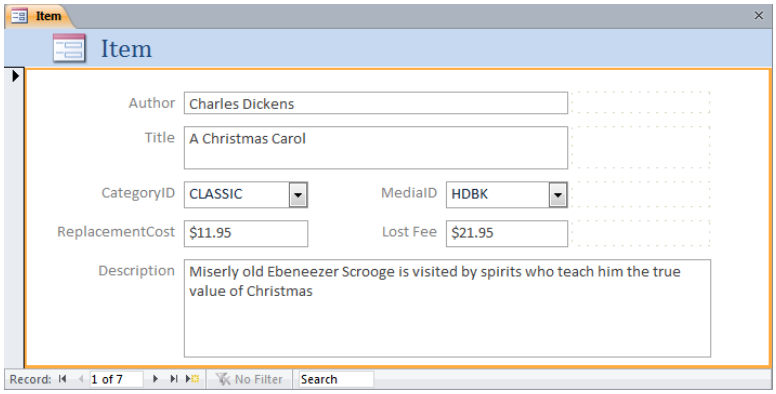


Figure 6-31. *The re-arranged* `Item` *form*

Adding a Subform

Now you'll add a subform to show the `InventoryItem` records for the selected `Item`. You will first need to provide a cell that the form can be placed in with the following steps:

1. Select the `Description` control and click the Insert Below button. This will create a new row of cells. Because the previous row had the last four cells merged together, the new row will be done the same way. This happens to be exactly what you want, in this scenario. If you needed individual cells, you could use the Split Horizontally button to split the merged cell.

2. Go to the Design tab of the ribbon and click the Subform button that is shown in Figure 6-32.

www.it-ebooks.info

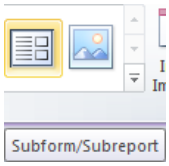


Figure 6-32. *Selecting the Subform control*

3. Select the empty cell that you just added. This will insert a Subform control in this cell and also add a Label control in the cell to the left of it, while the entire row is selected. Click some other control to unselect the row and then select the Subform control.

■ **Tip** These instructions assume you have the control wizards turned off. If the Subform Wizard starts just cancel the dialog box. I will explain the control wizard in a later chapter.

---

4. In the Data tab of the Property Sheet, select `Form.InventoryItem` for the Source Object property.
5. In the Other tab, change the Name property to **InventoryItem**. You'll need to enlarge this row so the entire subform can be seen. You may also need to make this cell wider. Also select the Label control and change its Caption property to **Inventory**.

■ **Note** The subform only shows a subset of the records in the `InventoryItem` table. Because of the relationship defined between the `Item` and `InventoryItem` tables, Access automatically filters the subform to records that are associated with the parent form.

---

6. Go to the Design View and select the `InventoryItem` form. In the Data tab of the Property Sheet, change the Allow Additions property to No. The user can view and modify records but they should not be allowed to add them.

■ **Tip** When you select the Subform control, an orange border will be drawn around it. You'll need to select the actual form, not the Subform control that contains the form. If you click inside the upper-left corner of the form (inside the Subform control), the orange border will disappear and there will be a small black square in the upper left corner. This indicates that the form has been selected. Also, the Property Sheet will indicate that the selected object is a Form.

---

www.it-ebooks.info

7. Save the form and enter **Item** as the form name when prompted. Switch to the Form View. The final form should look like Figure 6-33.

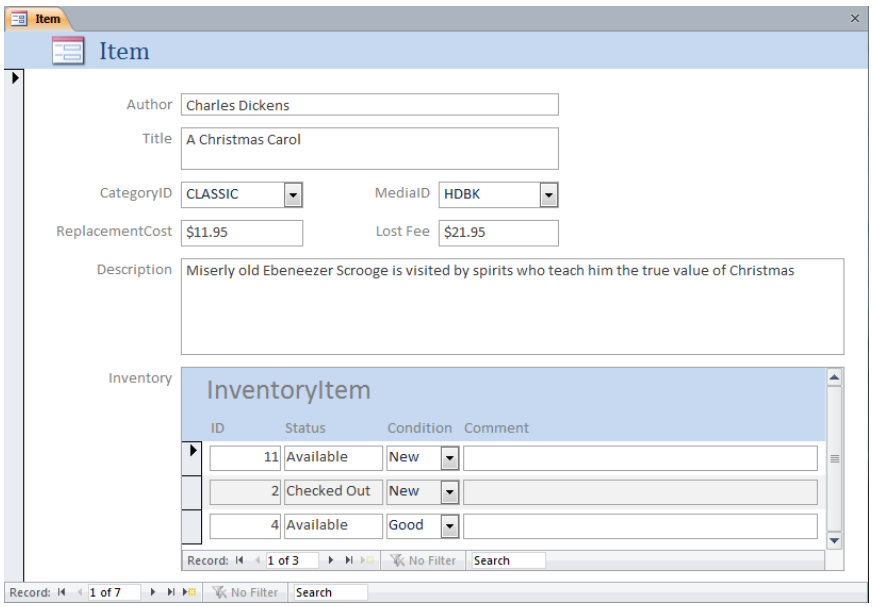


Figure 6-33. *The final version of the* `Item` *form*

Summary

In this chapter you created several useful forms without writing any code. The application now has forms that you'll use to view and modify categories, media types, and items. The key concepts that were covered include:

- Understanding both the Form View and Datasheet View
- Creating a Split Form, which combines both views
- Using a Continuous Form that combines the characteristics of both views
- Knowing which layout option to use in the Form Wizard
- Using the Layout View, a layout control, and the Arrange tab to format a form
- Embedding a subform

In the next few chapters, I'll build upon this foundation to create more complex forms that handle unique situations.

www.it-ebooks.info

CHAPTER 7

Creating a CheckOut Form

In the previous chapter, you created several forms using the design tools provided by Access. This is what Access does really well, and this is why it is such a popular platform for creating database solutions. It is really easy to create forms for your existing tables. The primary constraint, however, is that the standard forms can only access a single table. There are some ways around this. For example, you can use a query to join related tables and then the form can access all the fields in the query. This is really helpful, especially with a normalized database. You can also use subforms when a parent/child relationship exists between the tables.

As most of us have seen from some painful experiences, there are times when you need a form that just doesn't behave that way. The `CheckOut` form that you'll create in this chapter is a perfect example. It needs to do the following:

- Find and display `Customer` data
- Look up and display `Item` and `InventoryItem` information
- Insert records into the `Loan` table

In this chapter as well as the next, you'll build some complex solutions that will require Visual Basic for Applications (VBA) code to connect the pieces together. The forms themselves still rely on the foundational concepts presented in the previous chapter.

The process of building a checkout form will be described in the following steps:

1. Building a reusable `CustomerSearch` form.
2. Creating the `CheckOut` form, invoking the `CustomerSearch` form, and displaying the customer details.
3. Looking up and displaying an `InventoryItem` record.
4. Adding an item to the `CheckOut` form and finalizing the checkout process.

Implementing a Customer Search Feature

The first step during the checkout process is to identify the customer that is borrowing the items. In many cases, the customer will have a library card with them, and their `CustomerID` can be entered from that. We'll also need to allow a customer to be searched for based on name, address, phone, and so on. The search form will be implemented as a modal dialog so it can be reused in other places.

www.it-ebooks.info

Populating the Customer Table

The initial database that you created in Chapter 2 only has a few records in the `Customer` table. You'll need more records than that to test the search capability. To easily populate additional records, I have downloaded contact information for each of the members of the United States House of Representatives.

If you're starting each chapter with the online version of the previous chapter's `Library` database, then you already have these loaded. If not, you can download the `Library_06.accdb` file from `www.apress.com`, and use that as the starting point for this chapter's projects. If you want to keep using your own database, you can also download the `Customer.xlsx` file. Then copy and paste all the records (except the header) into your database. The `Customer` table should look like Figure 7-1.

Customer

Last Name	First Name	Address	City	State	ZIP Postal	Country Regic	PhoneNumber
Ackerman	Gary	2111 Rayburn House Office Building	Washington	DC	20515-3205	United States	(202) 225-2601
Adams	Sandy	216 Cannon House Office Building	Washington	DC	20515-0924	United States	(202) 225-2706
Aderholt	Robert	2264 Rayburn House Office Building	Washington	DC	20515-0104	United States	(202) 225-4876
Akin	W.	117 Cannon House Office Building	Washington	DC	20515-2502	United States	(202) 225-2561
Alexander	Rodney	316 Cannon House Office Building	Washington	DC	20515-1805	United States	(202) 225-8490
Altmire	Jason	332 Cannon House Office Building	Washington	DC	20515-3804	United States	(202) 225-2565
Amash	Justin	114 Cannon House Office Building	Washington	DC	20515-2203	United States	(202) 225-3831
Andrews	Robert	2265 Rayburn House Office Building	Washington	DC	20515-3001	United States	(202) 225-6501
Austria	Steve	439 Cannon House Office Building	Washington	DC	20515-3507	United States	(202) 225-4324

Figure 7-1. *`Customer` table populated with additional records*

Creating the CustomerSearch Dialog Box

From the Create tab of the ribbon, click the More Forms button, and then click the Modal Dialog link, as shown in Figure 7-2.

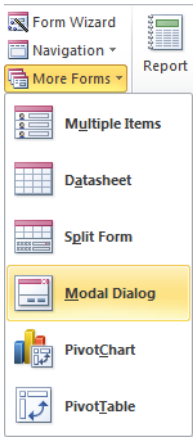


Figure 7-2. *Creating a modal dialog form*

www.it-ebooks.info

This creates a blank form that has OK and Cancel buttons on it. You'll add un-bound controls to the Form Header. The user will enter their search criteria in these fields. The Detail section will use standard data-bound fields that will display the records that match the specified criteria.

To display the Form Header section, right click on the form and click the Form Header/Footer link, as shown in Figure 7-3.

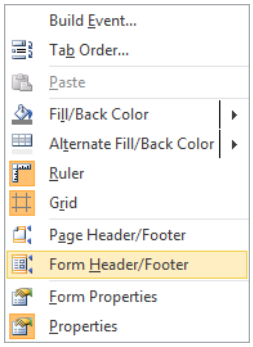


Figure 7-3*. Displaying the Form Header section*

The OK and Cancel buttons should be in the Form Footer. You can't drag controls from one section to another. Instead, use the cut and paste operations to move them to the footer.

Adding the Detail Fields

You'll start by adding the data-bound controls in the Detail section. In the previous chapter, this step was done for you, but you'll find it's fairly easy to do yourself. Follow these steps:

1. In the Property Sheet, select the Form object and the Data tab. Notice that the Record Source property is blank. There is a dropdown button that allows you to select an existing table or query. There is also a button with ellipses. This will launch the Query Designer that you used in Chapter 4. You can use this to create an ad-hoc query as the record source for this form. In this case, just select the `Customer` table from the dropdown list.
2. While you're here, set the Allow Additions and Allow Deletions properties to No. This form is for viewing only and should not be used for modifying records.

■ **Note** You can't set Allow Edits to No, because the user will need to edit the unbound controls that you will add later. Instead, you'll set the Locked property on the controls in the Detail section to prevent the user from modifying them.

---

www.it-ebooks.info

3. From the Create tab of the ribbon, click the Add Existing Fields button. This will display a list of the available fields in the selected record source (the `Customer` table). You'll double-click a field in this list to add it to the form. Add the following fields:
    - `CustomerID`
    - `LastName`
    - `FirstName`
    - `Address`
    - `PhoneNumber`
4. To quickly format the fields on the form, select all of the controls, including labels. In the Arrange tab of the ribbon, click the Tabular button. This will be a Continuous Form and you'll want the labels in the Form Header, which is what the Tabular button does. From the Property Sheet, select the Form object and the Format tab. Change the Default View property to Continuous Form.
5. Drag the labels to the bottom of the Form Header, and drag the data-bound controls to the top of the Detail section. Shrink the Detail section so there is just enough room for one row of controls. You'll also need to resize the data-bound controls, based on data that will be contained in each. You can switch to the Layout View to see how the actual data will appear.
6. Change the Caption property of the `CustomerID` label to **ID**. Save the form and enter the name **CustomerSearch** when prompted. The initial layout should be similar to Figure 7-4.

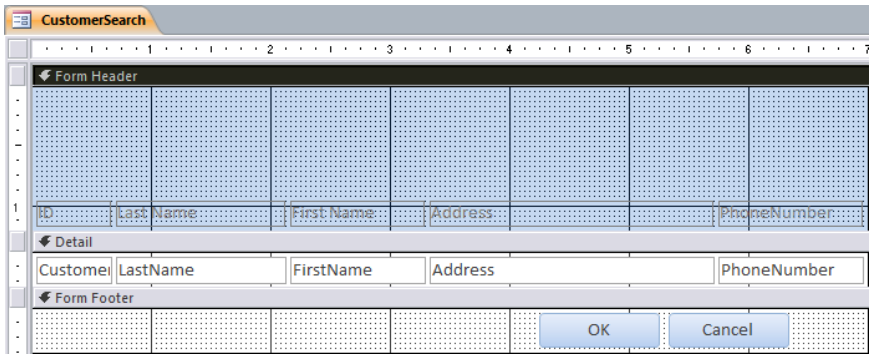


Figure 7-4. *Initial layout of the* `CustomerSearch` *form*

You'll need to make the following few adjustments to the Form's properties:

- In the Property Sheet, select the Form object.
- In the Format tab, change the Record Selectors property to Yes. Also set the Scroll Bars property to Vertical Only.

www.it-ebooks.info

- Select all of the data-bound controls and from the Property Sheet, set the Locked property (on the Data tab) to Yes. This will prevent the user from modifying the data.

7. Save the form changes. Try out the form by selecting the Form View. The dialog box should look like Figure 7-5.

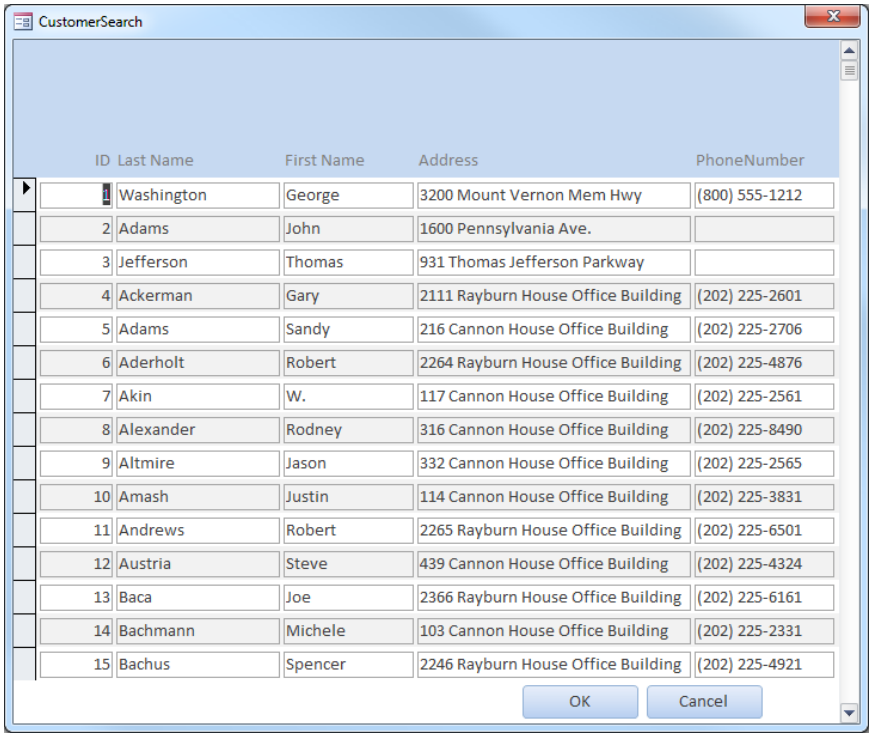


Figure 7-5. The initial implementation of the `CustomerSearch` *dialog*

Adding the Search Fields

At this point, the form returns all of the records in the `Customer` table. You'll now add unbound controls to the Form Header that the user will use to enter the search criteria. Follow these steps:

1. To add a TextBox control to a form, click the Text Box button in the Design tab of the ribbon.

www.it-ebooks.info

2. Draw a rectangle on the form where control should go. This will create a TextBox control and an associated Label control. You'll also need to change the name of the control. Because there are controls in the Detail section with similar names, prefix these controls with "txt," for example, **txtCustomerID**. You'll then edit the Caption property of the Label, which you can do directly on the form, jut edit the text that is displayed.

3. Add the following TextBox controls to the Form Header. Don't worry about the size or alignment; you'll take care of that with a layout control. The Form Header will look similar to Figure 7-6.

    - `CustomerID`
    - `Phone`
    - `Email`
    - `LastName`
    - `FirstName`
    - `Postal`

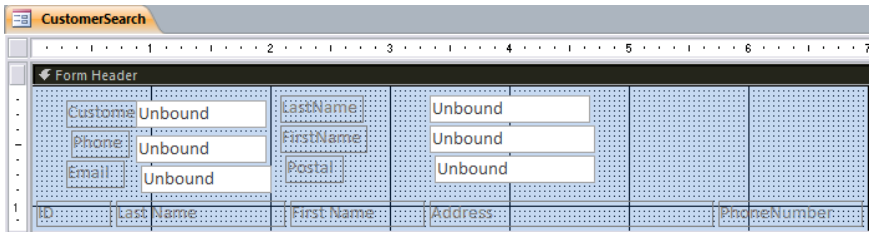


***Figure 7-6**. Initial Form Header layout with unbound controls*

4. Now select all of these controls and the associated labels, being careful not to include the column heading labels for the Detail section.

5. In the Arrange tab of the ribbon, click the Stacked button. This will stack all of the controls vertically. Click the Insert Right button in the ribbon four times to create four more columns. Click the Select Layout button to select all of the controls and drag everything to the left.

Follow these steps to arrange the control in the Form Header layout control:

1. Drag the `txtLastName` control and its label just to the right of the `txtCustomerID` control and remove the empty row (where the `txtLastName` control used to be).

2. In the same way, drag the `txtFirstName` control and its label just to the right of the `txtPhone` control and remove the empty row.

3. Drag the `txtPostal` control and its label to the right of the `txtEmail` control, leaving an empty cell between them. Delete the empty row.

www.it-ebooks.info

4. Select the `txtEmail` control and the empty cell next to it and click the Merge button.
5. Select all of the Label controls and in the Format tab of the Property Sheet, set the Text Align property to Right.
6. Select one of the column heading labels and click the Select Row button to select the rest of them. Position this row of labels so they are under the last unbound control.
7. Shrink the height of the Form Header to remove any unused space beneath the column headings.
8. Also shrink the width of the Form to remove any unused space.

Next, you'll need two command buttons. The first will execute the search and the second will clear the search criteria. Follow these steps:

1. In the Design tab of the ribbon, click the Button button and draw a rectangle in the Form Header. The Command Button Wizard will pop up, which allows you to select from some standard functions. Click the Cancel button, because you will be implementing this yourself.
2. Drag this control to the top-right corner cell of the layout grid. In the Format tab of the Property Sheet change the Caption property to **Search** and in the Other tab, change the Name property to **Search**.
3. In the same way, add a second button with the Caption and Name properties set to **Clear**. Drag this to the cell just below the Search button. The final layout should look like Figure 7-7.

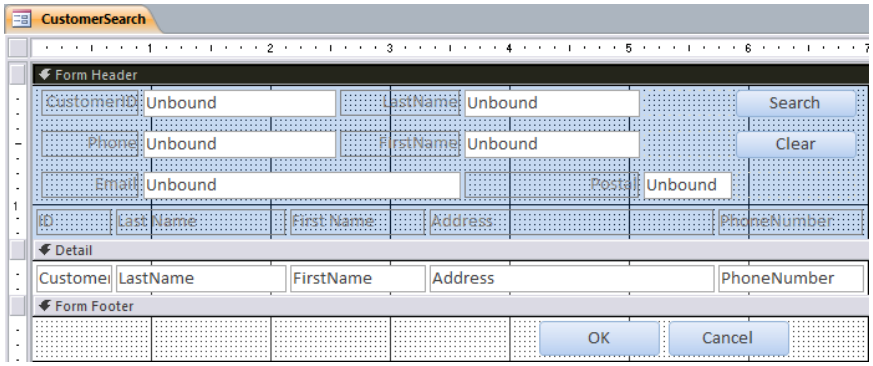


Figure 7-7. The final layout of the `CustomerSearch` form

Implementing the Search Logic

The search form is designed to display records from the `Customer` table. At this point, it is displaying all of the records. All you need to do now is apply a filter so the appropriate subset of records is returned. You'll do that using VBA code.

www.it-ebooks.info

Select the Search button. In the Property Sheet, select the Event tab to list all of the events that you can write a handler for. For a command button, you normally add code to the `OnClick` event. In the dropdown for this event, select Event Procedure and then click the ellipses next to it. This will display the VBA editor. In the `Search_Click` method, enter the implementation shown in Listing 7-1.

Listing 7-1. *The `OnClick` Event*

```
DoCmd.ApplyFilter "", _
       "([CustomerID] = [Forms]![CustomerSearch]![txtCustomerID] " & _
      "Or IsNull([Forms]![CustomerSearch]![txtCustomerID]))" & _
   "And ([LastName] Like [Forms]![CustomerSearch]![txtLastName] " & _
      "Or IsNull([Forms]![CustomerSearch]![txtLastName]))" & _
   "And ([FirstName] Like [Forms]![CustomerSearch]![txtFirstName] " & _
      "Or IsNull([Forms]![CustomerSearch]![txtFirstName]))" & _
   "And ([PhoneNumber] Like [Forms]![CustomerSearch]![txtPhone] " & _
      "Or IsNull([Forms]![CustomerSearch]![txtPhone]))" & _
   "And ([Email] Like [Forms]![CustomerSearch]![txtEmail] " & _
      "Or IsNull([Forms]![CustomerSearch]![txtEmail]))" & _
   "And ([ZipPostal] Like [Forms]![CustomerSearch]![txtPostal] " & _
      "Or IsNull([Forms]![CustomerSearch]![txtPostal]))", ""
```

This code calls the `ApplyFilter` method passing in the selection criteria. For each field that is in the Form Header, the logic checks to see if the database field matches what was specified or if a value was not specified. For the fields except `CustomerID` the comparison uses the `Like` operator. This allows for a partial value to be supplied, such as "Ad*".

Select the Clear button and for its `OnClick` event, select Event Procedure and click on the ellipses. In the VBA editor, enter the following code for the `Clear_Click` method. This code simply clears whatever values were supplied in the unbound controls.

```
txtCustomerID.Value = Null
txtLastName.Value = Null
txtFirstName.Value = Null
txtPhone.Value = Null
txtEmail.Value = Null
txtPostal.Value = Null
```

When the form is first loaded, it will display all the records in the `Customer` table, because the filter is not applied until the Search button is clicked. To keep it from doing that, you can set a default filter that will return no rows. In the Property Sheet, select the Form object and the Data tab. Enter **[CustomerID] = CLng('0')** for the Filter property. Also set the Filter On Load property to Yes. The `CustomerID` field is auto assigned and starts at 1, so it will never be 0. This is an effective and efficient way to keep any records from being loaded until the search criteria is entered.

Save the form and try it out by switching to the Form View. Try searching on various fields and using the wildcard character (*). You can also try entering a `CustomerID` value. The results should look like Figure 7-8.

www.it-ebooks.info

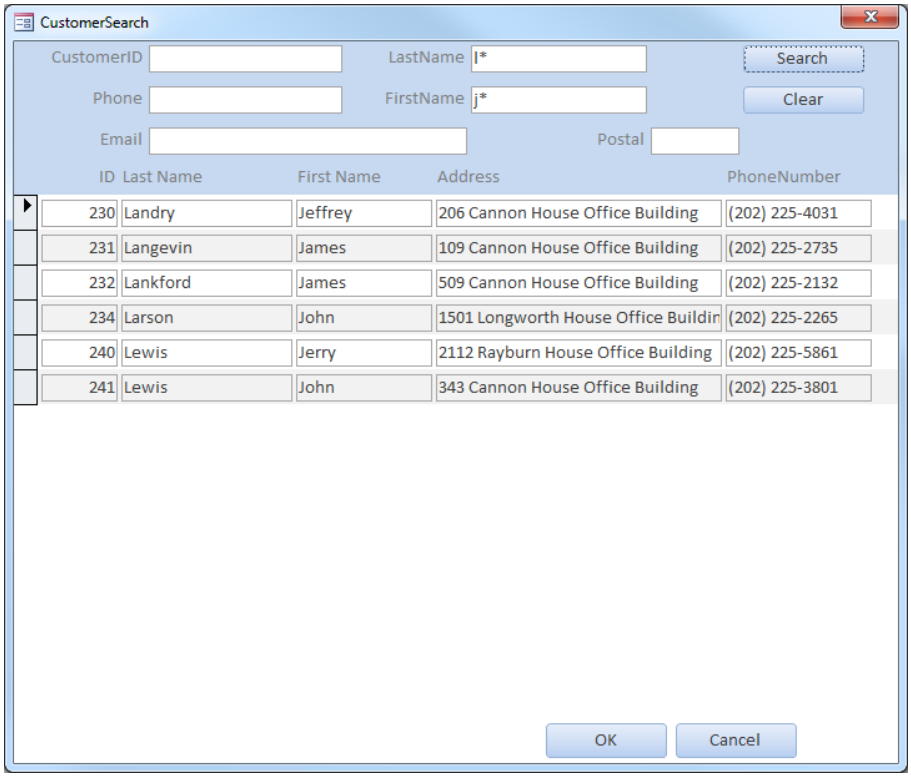


Figure 7-8. *The completed* `CustomerSearch` *form*

There's one more thing that you'll need to change to make this work. The `CustomerSearch` form will be launched from another form (the `CheckOut` form, in this case) for the purpose of selecting a customer. Once the user has closed the dialog, the main form will need to know if a customer was selected and, if so, which one. The solution to this is that the Cancel button will simply close the form. The OK button, however, will not close the form but merely hide it. To the user this will appear the same but hiding the form will leave it available to the calling form so the selected record can be determined.

We'll start by looking at the default implementation of the OK and Cancel buttons. Follow these steps:

1. Open the `CustomerSearch` form in the Design View.
2. Right-click the Cancel button and click the *Build Event* link. This will display the same Macro Designer that you used in Chapter 3 to create your data macros. The available actions are different, because you're working with Windows controls instead of database objects.

www.it-ebooks.info

■ **Tip** You can use this technique to view and edit the VBA event handlers as well. Right-click the Search button and click the *Build Event* link. This will display the VBA editor if the event handler is implemented in VBA.

3. This macro simply calls the `CloseWindow` action, which will close the dialog. The Save parameter is set to Prompt, which will prompt the users and ask if the changes (if there have been any) should be saved or discarded. Change that now and choose the No option. The warning icon is letting you know that this is an unsafe action because changes could be lost. For this form, we don't want any changes made, so this is OK. The macro should look like Figure 7-9.

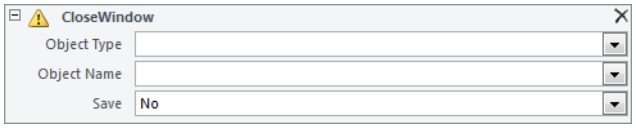


Figure 7-9. *Updated* `Cancel` *macro*

4. Click the Save button in the ribbon to save this change and then click the Close button to close the editor.
5. If you look at the event handler for the OK button, you'll see that it is exactly the same as for the Cancel button. Select the OK button. In the Event tab of the Property Sheet, change the On Click property from Embedded Macro to Event Procedure. Instead of executing a macro, the `OnClick` event handler will now invoke some VBA code. Click the ellipses to display the VBA editor.
6. In the `Command1_Click` method, enter the following code:

```
If (Me.CurrentRecord = 0) Then
   MsgBox "Please select a customer", vbExclamation, "No Customer Selected"
Else
   Me.Visible = False
End If
```

This code checks to see if they selected a record. If not, an error message is displayed. Otherwise, the Visible property is set to False, which will hide the form.

■ **Caution** The calling form must be sure to close the `CustomerSearch` form after it has determined the selected customer.

www.it-ebooks.info

Building a CustomerDisplay Form

The first thing that the user will do with the `CheckOut` form is to invoke the `CustomerSearch` dialog and select a customer. The selected customer should be displayed on the form. This is just for display purposes only; in fact, it would best to display this like a mailing label rather than a series of controls. In order to be able to reuse this on other forms, you'll build this as a stand-alone form now, which can be included on the `CheckOut` form as a subform. Follow these steps:

1. From the Create tab of the ribbon, click the Blank Form button.
2. Switch to the Design View. In the Property Sheet, select the Form object and the Data tab.
3. Select the `Customer` table for the Record Source property. Set the Allow Additions, Allow Deletions, and Allow Edits properties to No.
4. Click the Add Existing Fields button in the ribbon, and double-click each of the fields that are listed to add them to the form. These controls will be used for retrieving the data from the record and will not be visible. Delete each of the associated labels.
5. Select all of the data-bound controls. In the Format tab of the Property Sheet, set the Visible property to No. Resize the controls to be about 1 character wide and drag them to the top-left corner of the form. The form will look like Figure 7-10.

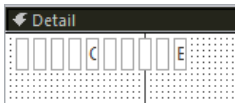


***Figure 7-10**. Initial `CustomerDisplay` form layout*

6. Now you'll add the controls that will display the formatted customer information. Add five TextBox controls to the form with the following names:
   - `lblName`
   - `lblAddress`
   - `lblRegion`
   - `lblEmail`
   - `lblPhone`
7. Delete all of the associated labels. Now select the five unbound controls and from the Arrange tab in the ribbon, click the Stacked button. This will create a single column of controls.

www.it-ebooks.info

8. Click the Insert Right button to add a second column. Drag the `lblPhone` control to the right of the `lblEmail` control, and delete the empty row. Select the `lblName` and the empty cell next to it, and click the Merge button. In the same way, merge the `lblAddress` and `lblRegion` control with the empty cell next to them.

9. Click the Select Layout button to select all of the controls. Drag the layout to the upper-left corner of the form. While the layout is still selected, click the Control Padding button in the ribbon and click the None link as shown in Figure 7-11.

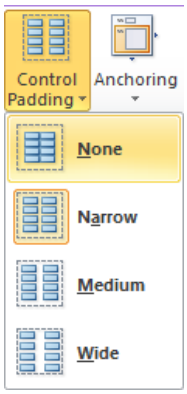


***Figure 7-11**. Selecting no cell padding*

10. While the layout is still selected, in the Format tab of the Property Sheet, clear the Border Color property (delete the text in the property value). When you tab off this property, the value will be changed to "#000000." Then set the Border Style to Transparent. When you click on the Border Color property, it should change to No Color. Also, set the Back Style property to Transparent. Finally, set the Height property to **.2."**

■ **Tip** You can make a TextBox control look like a Label by removing the border and making the background transparent, as you did here.

---

11. Resize the form to make it as small as possible, with no extra space below or to the right of the controls. Save the form and enter the name **CustomerDisplay** when prompted. The layout should look like Figure 7-12.

www.it-ebooks.info

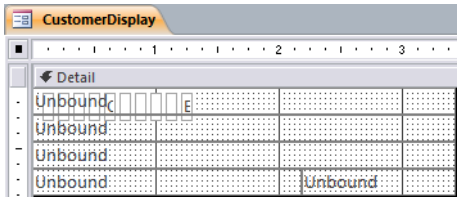


Figure 7-12. *The* `CustomerDisplay` *form layout*

12. Now you'll need to write some code to format the text in the unbound controls. In the Property Sheet, select the Form object and the Event tab. For the `OnCurrent` event, select Event Procedure, and then click the ellipses to display the VBA editor.

13. In the `Form_Current` method, enter the code shown in Listing 7-2.

Listing 7-2. *Implementation of the* `OnCurrent` *Event for the* `CustomerDisplay` *Form*

```
If (Len(LastName) > 0) Then
   lblName = LTrim(RTrim(FirstName) & " ") & RTrim(LastName) & "  " & _
      CStr(CustomerID)
   lblAddress = Trim(Address)
   lblRegion = RTrim(City) & ", " & LTrim(RTrim(StateProvince) & " ") & _
      RTrim(ZIPPostal)
   lblEmail = RTrim(Email)
   lblPhone = RTrim(PhoneNumber)
Else
   lblName = ""
   lblAddress = ""
   lblRegion = ""
   lblEmail = ""
   lblPhone = ""
End If
```

This code determines if a record was selected by checking the `LastName` field. If there is no record, the controls are all cleared. Otherwise, the appropriate fields are concatenated together to format a mailing label.

Save the code and save the form changes. Switch to the Form View. The format should look like Figure 7-13.

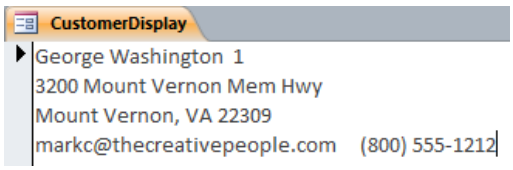


Figure 7-13. *The final output of the* `CustomerDisplay` *form*

www.it-ebooks.info

Go back to the Design View. This form should display a single record so you'll need to remove all the navigation controls. In the Property Sheet, select the Form object and the Format tab, and then set the following properties:

- Border Style: None
- Record Selectors: No
- Navigation Buttons: No
- Scroll Bars: Neither
- Control Box: No
- Close Button: No
- Min Max Buttons: None

In the Data tab, set the Filter property as [CustomerID] = CustomerID and set the Filter On Load property to Yes. When you drop this form onto another form, you'll set up the parent/child linkage so the correct record will be returned based on the parent form.

Creating the CheckOut Form

Now you are ready to create a `CheckOut` form that will use both the `CustomerSearch` and `CustomerDisplay` forms. The controls for the customer search feature will all be in the Form Header; you'll save the Detail section for the items being checked out.

Adding the Customer Controls

Now you'll add controls to the Form Header that you'll use for searching for and displaying the selected customer.

1. From the Create tab of the ribbon, click the Blank Form button. Switch to the Design View and then right-click the form and click the *Form Header/Footer* link. Expand the Form Header so you'll have some room to work with.
2. Add a TextBox control to the Form Header and change the Name property to **txtCustomerID**. Set the Caption of its associated label to **CustomerID:**. Set the Text Align property for the label to Right.
3. Add a command button to the Form Header and cancel the Command Button Wizard. Change the Name property to **Search** and set the Caption property to **Search…**
4. Select all three controls. From the Arrange tab of the ribbon, click the Stacked button. This should create two columns.
5. Click the Insert Right button on the ribbon to create a third column.
6. Drag the command button to the last column of the first row.
7. Merge all three cells of the second row. Enlarge the second row to be about 1-inch high.

www.it-ebooks.info

8. From the Design tab of the ribbon, click the Subform button and then click Form Header section. This will display the SubForm Wizard. Select the `CustomerDisplay`, as shown in Figure 7-14.

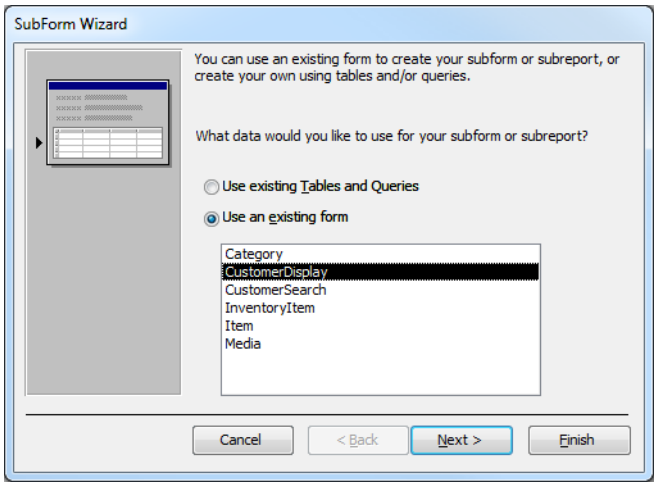


Figure 7-14. *Selecting the* `CustomerDisplay` *form*

■ **Tip** If the SubForm Wizard does not appear, you probably have the control wizards turned off. To enable them, expand the list of controls in the Design tab of the ribbon so all controls are displayed. Then click the *Use Control Wizards* link.

9. In the next dialog box, use the default value, CustomerDisplay, for the name of the subform. Click the Finish button.
10. In addition to the subform, a Label control is created, delete this. Drag the subform to the second row of the layout control.
11. Switch to the Layout View and resize the cells as appropriate.
12. Save the form and enter the name **CheckOut** when prompted.

The final layout should look like Figure 7-15

www.it-ebooks.info

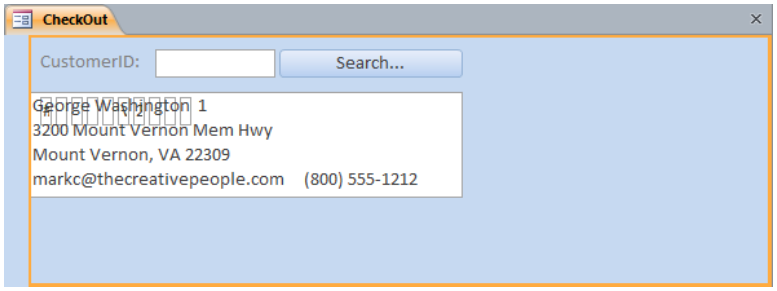


Figure 7-15. The layout of the Form Header section

Connecting the Controls

Now you'll tie the pieces together so this will function correctly. The first step is to link the subform, `CustomerDisplay`, with its parent form. The parent form, `CheckOut` has a `txtCustomerID` control that will specify the selected customer. The child form has a `CustomerID` data-bound control that is also used as the filter for the form. You'll now link these control together.

1. Go back to the Design View. In the Property Sheet, select the `CustomerDisplay` object and the Data tab.
2. Enter **txtCustomerID** for the "Link Master Fields" property. Likewise, enter **CustomerID** for the "Link Child Fields" property. The Property Sheet will look like Figure 7-16.

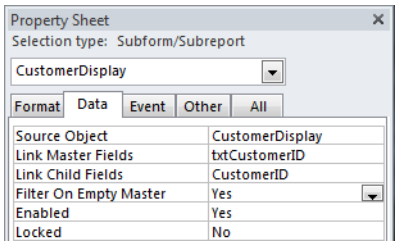


Figure 7-16. The subform data properties

3. Go to the Format tab and set the Visible property to No. This will keep the subform hidden until there is data to display.
4. Now you'll add code to handle the scenario where the user enters a customer ID directly into the form. Select the `txtCustomerID` control. In the Event tab of the Property Sheet, select Event Procedure for the On Lost Focus property. Then click the ellipses to edit the associated VBA code. Enter the following code for the `txtCustomerID_LostFocus` method:

```
CustomerDisplay!lblName = ""
CustomerDisplay!lblAddress = ""
```


www.it-ebooks.info

```
CustomerDisplay!lblRegion = ""
CustomerDisplay!lblEmail = ""
CustomerDisplay!lblPhone = ""

DoCmd.Requery "CustomerDisplay"

If (Len(CustomerDisplay!lblName) > 0) Then
   CustomerDisplay.Visible = True
Else
   CustomerDisplay.Visible = False
End If
```

This code first clears the labels and then forces a re-query against the database. It then checks to see if there is any data to display. If not, the subform is hidden. If an invalid customer ID is entered then no data will be displayed.

Save the code changes and the form. Try it out by switching to the Form View. Enter a number in the `CustomerID` field and then tab off the field. The form should look similar to Figure 7-17.

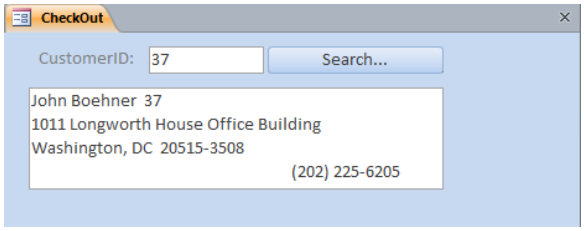


Figure 7-17. *The Form Header displaying the customer information*

Try entering an invalid ID and verify that the subform is hidden.

Invoking the Search Dialog

Now you'll add code to invoke the `CustomerSearch` dialog when the Search button is clicked. This code will also determine the customer that was selected.

1. Go back to the Design View. Right-click the Search button and click the *Build Event* link. Because there is no code currently associated with this event, the dialog box in Figure 7-18 is displayed for you to select the type of event handler.

www.it-ebooks.info

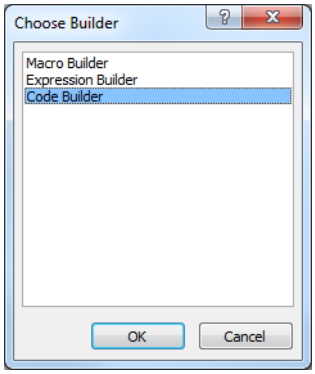


Figure 7-18. *Selecting the type of event handler*

2. Choose the Code Builder option and click the OK button.
3. Enter the code shown in Listing 7-3 for the implementation of the `Search_Click` method.

Listing 7-3. *Implementation of the* `Search_Click` *Method*

```
Dim sForm As String
sForm = "CustomerSearch"

' Open the search form
DoCmd.OpenForm sForm, acNormal, , , , acDialog

' If the form is not loaded, the user clicked the Cancel button
If (IsLoaded(sForm)) Then
   txtCustomerID = Forms(sForm)!CustomerID
   DoCmd.Close acForm, sForm
Else
   txtCustomerID = ""
End If

' Force a refresh if the CustomerDisplay subform
txtCustomerID_LostFocus
```

This code opens the `CustomerSearch` form, which is implemented as a modal dialog. Because it is modal, the `OpenForm` method does not return until the form has been closed or hidden. The code then checks to see if the form is still loaded. If it is, the `CustomerID` is obtained and the form is closed. Finally, the `LostFocus` event handler is called, which you just implemented. This will cause the `CustomerDisplay` subform to be updated with the selected customer, or hidden if no customer was selected.

This code uses a function called `IsLoaded`. You will need to implement that as well. Since this will be needed from several forms, you should implement this in the `Main` module so it can be shared by all the forms. To add a module to the VBA code, right-click the `Library` project and click the *Insert* ➤ *Module* links.

This will open up a new file. Add the following code to this file:

www.it-ebooks.info

```
Function IsLoaded(ByVal strFormName As String) As Boolean
 ' Returns True if the specified form is open in Form view or Datasheet view.
  Const conObjStateClosed = 0
  Const conDesignView = 0

  If SysCmd(acSysCmdGetObjectState, acForm, strFormName) <> conObjStateClosed Then
     If Forms(strFormName).CurrentView <> conDesignView Then
        IsLoaded = True
     End If
  End If
End Function
```

Save the code and enter **Main** when prompted for the file name. The `SysCmd` method is a built-in function provided by Access, which can be used for a number of purposes depending on the action that is specified in the first parameter. This code uses it to check the status of the specified form; to see if it is loaded. If it is, the code then checks to see if it is opened in Design View.

■ **Note** The `IsLoaded` function is provided by Microsoft. You can find more information about it at `http://msdn.microsoft.com/en-us/library/aa141272(office.10).aspx`. In my opinion, this should be provided as a built-in function. Because it is not, you'll need to make sure you add it to the `Main` module. If you forget, you'll get an error when your code executes.

Go to the Form View and click the Search button. Enter some search criteria and select one the records as shown in Figure 7-19. The record selector to the left of the row will indicate which record was selected.

www.it-ebooks.info

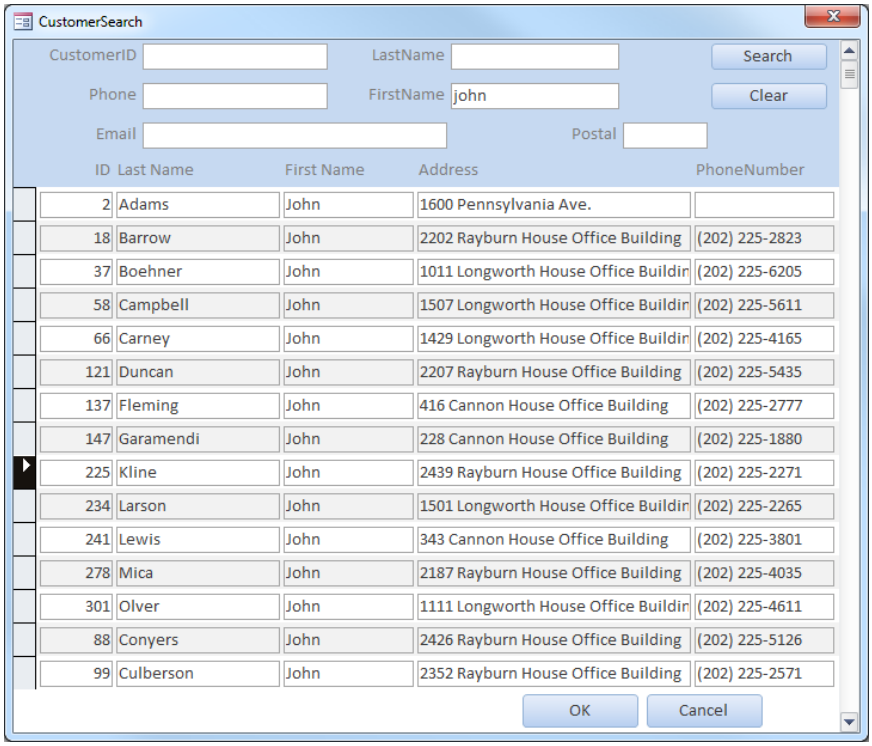


Figure 7-19. *Searching for a customer*

Click the OK button. You should see the selected customer displayed in the Form Header, as shown in Figure 7-20.

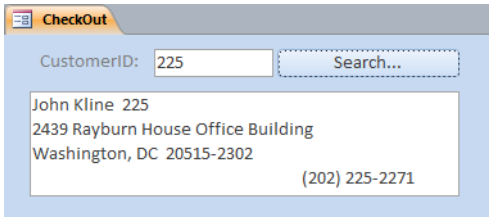


Figure 7-20. *The Form Header showing the selected customer*

Try searching for a customer and clicking the Cancel button. In this case, no customer data should be shown.

www.it-ebooks.info

Providing a CheckOut Feature

At this point the CheckOut form allows you to enter a customer ID or search for a customer and then display their information. Now that the preliminaries are out of the way, you can build a feature that will check out items to the selected customer.

The checkout process will start by entering an InventoryItemID. It is assumed that the item will have an inventory tag on it that can be scanned or entered so you won't need a search feature. When this ID has been entered, it's a good idea to look up the item and display details of the item, including its current condition and status. If the item is available, the form will allow the item to be checked out.

An item is checked out by creating a Loan record, which requires only the CustomerID and InventoryItemID fields. The data macros that you implemented in Chapter 3 will take care of any other updates that are needed, such as updating the InventoryItem record. The Loan record will be added to the Detail section of the CheckOut form. As additional items are checked out, they are added to the form so you can keep a running total of everything being checked out.

You will now implement this feature in the following steps:

1. Create a form for displaying details of an InventoryItem record.
2. Embed this as a subform in the CheckOut Form Header
3. Add logic to generate the Loan records when the item is checked out.
4. Implement some finalization code when the checkout process is complete.

Building an InventoryItemLookup Form

You'll now create a form that will display a single inventory item. You already have an InventoryItem form that is used as a subform on the Item form. This form will not work here, because it was designed to be used in the context of an Item record. You'll need a form that provides details from both the Item and InventoryItem tables for a single InventoryItem record.

You'll create a blank form and build an ad-hoc query as the record source. Then you'll add the fields to the form and use a layout control to arrange them. You'll set up a filter to display a single record and remove the selection and navigation control just like you did with the CustomerDisplay form.

From the Create tab of the ribbon, click the Blank Form button. Switch to the Design View.

Creating an Ad-Hoc Query

Now you'll design an ad-hoc query that will return the appropriate subset of records from the InventoryItem table.

1. In the Property Sheet, select the Form object and the Data tab. Click the ellipses next to the Record Source property. This will display a blank query in Design View.
2. Add the following tables to the query (the query should look like Figure 7-21):
   - InventoryItem
   - Item

www.it-ebooks.info

- Category
- Media

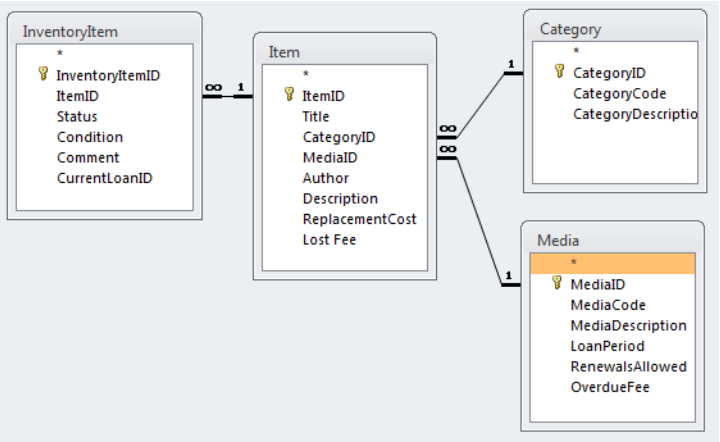


Figure 7-21*. The query design for the* `InventoryItemLookup` *form*

3. Double-click the following fields to add them to the query:
   - `InventoryItem.InventoryItemID`
   - `InventoryItem.Status`
   - `InventoryItem.Condition`
   - `InventoryItem.Comment`
   - `Item.Author`
   - `Item.Title`
   - `Item.Description`
   - `Category.CategoryDescription`
   - `Media.MediaDescription`
   - `Media.LoanPeriod`
4. Click the Close button. You'll see the dialog box shown in Figure 7-22. Click the Yes button to update the Record Source property.

www.it-ebooks.info

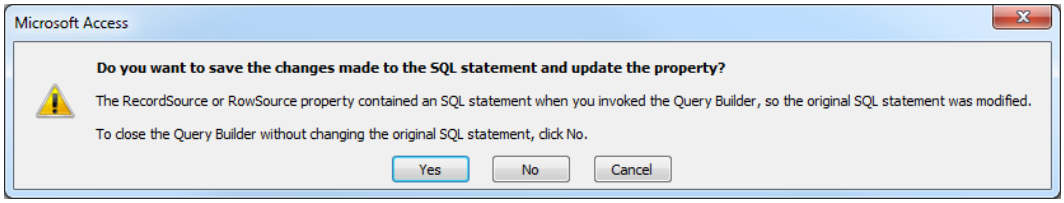


Figure 7-22. *Dialog box confirming the property change*

Adding the Fields to the Form

The fields from this query will now be added to the form and the initial form layout will be designed.

1. Click the Add Existing Fields button in the ribbon, which will list all of the available fields. Double-click each one to add it to the form.
2. Delete the label for the `InventoryItemID` control. The `InventoryItemID` field will be hidden and only used for linking with the parent form. Select this control and in the Format tab of the Property Sheet set the Visible property to No. Drag this control to the top-left corner of the form.
3. Select all of the controls except `InventoryItemID` and their associated labels. From the Arrange tab of the ribbon, click the Stacked button to create a layout control. Click the Insert Right button twice to add two more columns. Click the Insert Above button twice to add two blank rows at the top of the form.
4. To save space on the form, delete the labels for all of the fields except `Comment` and `LoanPeriod`. The data for these fields should be fairly self-explanatory.
5. Drag the `Title` field to the top-left cell. Then select this cell and the three cells to the right and click the Merge button.
6. Drag the `Author` control to the first cell in the second row. Merge this cell with the cell next to it (the `Author` field will use the first two cells of this row).
7. Delete the `Status` control. This was generated as a ComboBox control but we don't want this field editable so you'll replace it with a TextBox control. From the Design tab of the ribbon, click the Text Box button and then draw a rectangle in the form to create the control. Delete the associated label control. Select the new TextBox control and change its Name property to **Status**. For the Control Source property, select the `Status` field. Drag this control just to the right of the `Author` control.
8. Drag the `Condition` control to the right of the `Status` control. You'll leave this as a ComboBox control, because the user will be able to modify the condition of the item.
9. Drag the `Comment` control to the second cell in the third row. Its label should move to the first cell automatically. Merge this control with the two empty cells beside it.

www.it-ebooks.info

10. Drag the `CategoryDescription` control to the first cell in the fourth row. Drag the `MediaDescription` control to the second cell. Drag the `LoanPeriod` control to the fourth cell. Again, its label should move to the third cell.
11. Drag the `Description` control to the fifth row. Merge all the cells on this fifth row into a single cell.

Modifying the Form Design

Next you'll perform some cleanup, removing unused cells and setting an appropriate cell height.

1. Remove all of the empty rows.
2. Select one of the controls in the layout and click the Select Layout button to select all of the controls. Click the Control Padding button, and then click the None link.
3. You may have noticed that the blank rows at the top of the layout are not as high as the other rows. To assure uniformity, select all the cells and in the Format tab of the Property Sheet, set the Height property to **.22."** The `Description` control will need to be larger, however. Select only the `Description` control and resize it to be about 3 lines high.
4. Select the entire layout and drag it to the top-left corner of the form.
5. Go to the Layout View and resize the controls as necessary so the data fits properly.
6. Go back to the Design View and shrink the form size to remove all unused space.

Finalizing the Form Details

In the final step, you'll lock down the fields that should not be editable and finalize the form implementation.

1. Select the following controls and in the Data tab of the Property Sheet, set the Locked property to Yes. This will prevent the user from modifying these fields because the only fields that will be editable are `Condition` and `Comment`.
    - `Title`
    - `Author`
    - `Status`
    - `CategoryDescription`
    - `MediaDescription`
    - `LoanPeriod`
    - `Description`

www.it-ebooks.info

2. Save the form and enter the name **InventoryItemLookup** when prompted.
3. You will need an unbound control that you can use to indicate if a record was read successful. Add a TextBox control to the form and set its Name property to **lblTitle**. Also set its Visible property to No. Delete the associated label.
4. In the Property Sheet, select the Form object and the Event tab. For the On Current property select Event Procedure and click the ellipses. Enter the following code for the implementation of this event handler:

```
lblTitle = Title
```

This code simply updates the unbound control with whatever is in the `Title` field.

■ **Tip** The code that checks to see if a record was read successfully first sets one of the fields to a blank string. It then re-queries the form and checks to see if the value is still blank. You can't do this with a bound control, because setting it to blank will update the record. Instead you'll use the `lblTitle` control, which is copied from the bound `Title` control when a record is read.

The final layout should look like Figure 7-23.

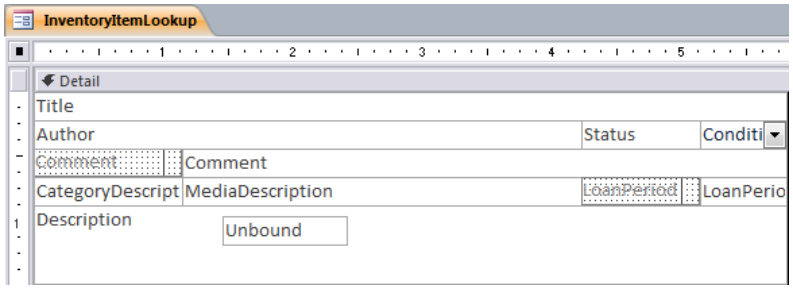


Figure 7-23. *The layout of the* `InventoryItemLookup` *form*

5. Just like the `CustomerDisplay` form that you designed earlier, this form should display a single record so you'll need to remove all the navigation controls. In the Property Sheet, select the Form object and the Format tab. Set the following property values:
    - Border Style: None
    - Record Selectors: No
    - Navigation Buttons: No
    - Scroll Bars: Neither

www.it-ebooks.info

- Control Box: No
- Close Button: No
- Min Max Buttons: None

6. Also, in the Data tab, set the Filter property as [**InventoryItemID**] = **InventoryItemID** and set the Filter On Load property to Yes. Test out the form change switching to the Design View. The form should look like Figure 7-24.

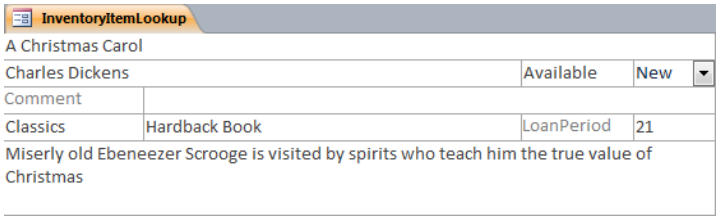


Figure 7-24. *The completed* `InventoryItemLookup` *form*

Linking the InventoryItemLookup Subform

Now you're ready to add the `InventoryItemLookup` form as a subform to the `CheckOut` form. This will be done just like you added the `CustomerDisplay` form. You'll add a TextBox control where the user can enter the ID of the inventory item. This will be linked to the child form so the details can be displayed.

1. Close the `InventoryItemLookup` form and then open the `CheckOut` form in the Design View.
2. You'll need to add some more cells to drop the new controls into. Go the Arrange tab of the ribbon. Select the existing layout control and click the Insert Right button five times and the Insert Below button once.
3. Add a TextBox control to the Form Header and set its Name property to **txtInventoryItemID**. Enter the Caption for the associated label as **InventoryID:** and set the Text Align property to Right. Drag the `txtInventoryItemID` control to the top row, leaving two blank cells to the right of the `Search` button. The label should move to the left of this control, leaving a single blank cell after the `Search` button.
4. Merge the last four cells of the last two rows into a single cell. This will leave a blank column between the existing customer controls and the new inventory item controls.
5. From the Design tab of the ribbon, click the Subform button and then click inside the Form Header. This will start the Subform Wizard. Select the `InventoryItemLookup` form, as shown in Figure 7-25.

www.it-ebooks.info

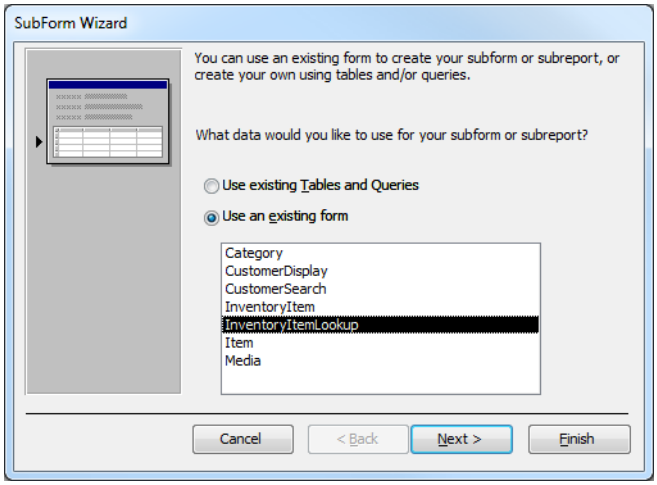


Figure 7-25. Selecting the existing `InventoryItemLookup` form

6. In the next dialog box, accept the default name, which should be InventoryItemLookup.
7. Delete the associated label that was generated and drag the subform to the merged cell at the bottom right of the Form Header.
8. Switch to the Layout View and resize the cells so the subform fits properly. The layout should look like Figure 7-26.

■ **Tip** To increase the vertical size of the merged cell without affecting the cells used by the `CustomerDisplay` form, select the blank cell underneath the `CustomerDisplay` form and change its height. Likewise, to increase the horizontal size, select the cell to the right of the `txtInventoryItemID` control and increase its width.

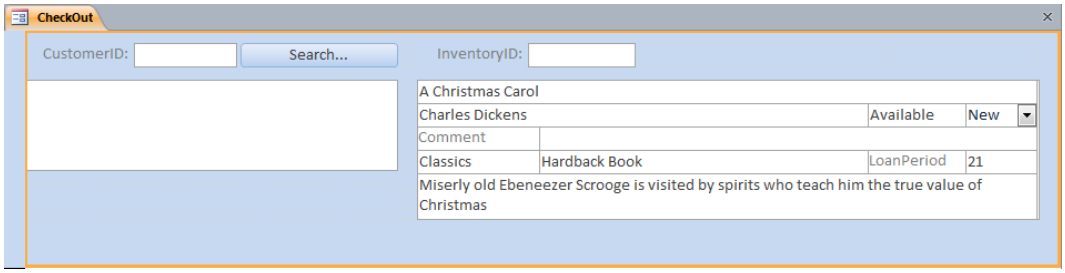


Figure 7-26. The layout of the `CheckOut` Form Header

www.it-ebooks.info

9. Go back to the Design View and select the InventoryItemLookup subform. In the Data tab of the Property Sheet, enter **txtInventoryItemID** for the Link Master Fields property and **InventoryItemID** for the Link Child Fields property. This will establish the link between the parent and child forms. Also, in the Format tab, set the Visible property to No.

10. The last step is to implement the LostFocus event for the txtInventoryItemID control. Select this control and in the Event tab of the Property Sheet, select Event Procedure for the On Lost Focus property. Then click the ellipses to display the VBA code.

11. For the txtInventoryItemID_LostFocus method, enter the following implementation:

```
InventoryItemLookup!lblTitle = ""

DoCmd.Requery "InventoryItemLookup"

If (Len(InventoryItemLookup!lblTitle) > 0) Then
   InventoryItemLookup.Visible = True
Else
   InventoryItemLookup.Visible = False
End If
```

This code should be familiar to you since it's very similar to the LostFocus event that you implemented for the txtCustomerID control. It clears the unbound TextBox control (lblTitle), re-queries the form and then checks to see if the control is still blank.

Save the code and save the form. Test out the CheckOut form by switching to the Form View. Enter an InventoryItemID and verify that it displays the data correctly. Try entering a comment and changing the condition. Also try entering an invalid InventoryItemID and verify that no information is displayed.

Designing the CheckOut Details

When an item is checked out, a Loan record is created. The Detail section will contain these Loan records; as items are checked out, they will be added to the Detail section of the form. The record source for the form will be the Loan tabled, filtered to only include those records loaned out to the current customer.

However, the customer could have items that were previously checked out so the query needs to only retrieve the current items. To accomplish that, you'll need to modify the Loan table to add an InProgress field. You'll set the default value to be Yes so all new records will show as in progress. When the checkout has completed, you'll reset this flag.

Altering the Loan Table

Open the Loan table in the Design View. Add a new field named **InProgress** and select Yes/No for the Data Type. In the Field Properties window, set the Default Value property to **1**, as shown in Figure 7-27.

www.it-ebooks.info

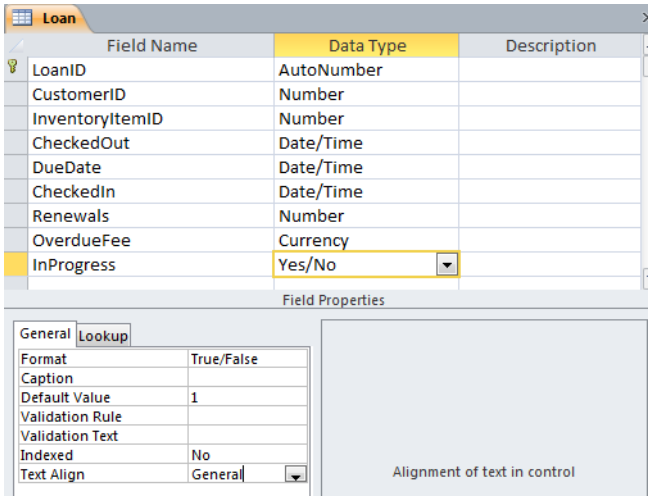


Figure 7-27. *Adding the* `InProgress` *field*

Creating a LoanDetail Query

The next step is to create a query that returns all in progress records for the current customer. From the Create tab of the ribbon, click the Query Design button. Add the following tables to the query:

- `Loan`
- `InventoryItem`
- `Item`

There are two relationships between the `Loan` and `InventoryItem` tables; delete the one between `Loan.LoanID` and `InventoryItem.CurrentLoanID`. (This only affects the current query and not the underlying table relationships.) Add the following fields to the query:

- `Loan.CustomerID`
- `Loan.InventoryItemID`
- `Loan.CheckedOut`
- `Loan.DueDate`
- `Loan.InProgress`
- `Item.Title`

For the `CustomerID` field, enter **[Forms]![CheckOut]![txtCustomerID]** for the criteria. For the `InProgress` field, enter **True** for the Criteria. Save the query and enter the name as **LoanDetail** when prompted. The query design should look like Figure 7-28.

www.it-ebooks.info

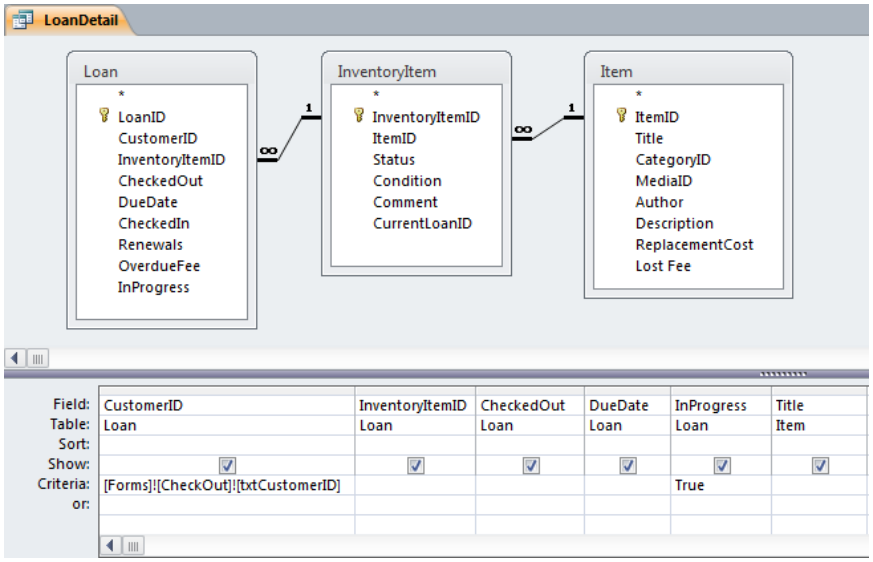


Figure 7-28. The query design for `LoadDetail`

Designing the Detail Section

Now you'll set up the new `LoanDetail` query as the record source for the `CheckOut` form and add data-bound controls to the Detail section. This will display the items that have been checked out so far.

1. Open the `CheckOut` form in the Design View. In the Property Sheet, select the Form object and the Data tab. For the Record Source property, select the `LoanDetail` query.
2. Click the Add Existing Fields button in the ribbon. Double-click each of the following fields to add it to the form:
   - `InventoryItemID`
   - `Title`
   - `DueDate`
   - `CustomerID`
3. Select all of these controls. From the Arrange tab of the ribbon, click the Tabular button. The labels will be placed in the Form Header.
4. Select all of the labels and drag them to the bottom of the Form Header. Select all of the data-bound controls and drag then to the top of the Detail section. Select both the controls and the labels and drag them to the left edge of the form.
5. Make the `Title` and `DueDate` controls wider.

www.it-ebooks.info

6. The `InventoryItemID` and `CustomerID` controls were created as ComboBox controls. These will be read-only fields, so you'll need to replace them with TextBox controls. Delete both controls.
7. Add a TextBox control to the Detail section. Delete the associated label. Change the Name property to **InventoryItemID**, and select `InventoryItemID` as the Control Source.
8. In the same manner add another TextBox control, delete the associated label, set the Name property to CustomerID, and select `CustomerID` as the Control Source.
9. Drag these controls into the cells where the ComboBox controls were.
10. Delete the label for the `CustomerID` field. Select the `CustomerID` control and set the Visible property to No. The `CustomerID` control is used for inserting records and does not need to be displayed.
11. Select all the data-bound controls in the Detail section and set the Locked property to Yes.
12. In the Property Sheet, select the Form object and set the Default View property to be Continuous Form.

Adding the Header and Footer Controls

There are just a few more things to finish up the form design. You'll need a CheckOut button in the Form Header that will create the `Loan` record. You'll also need a Complete button in the Form Footer that will execute the final step of the process. You'll also add a control to the Form Footer to keep a running count of the number of items being checked out.

1. Add a command button to the Form Header. Cancel the Command Button Wizard and drag the control to the top-right cell. Set the Name property to **CheckOut** and the Caption property to **CheckOut** as well. Set the Visible property to No.
2. Add a command button to the Form Footer. Again, cancel the wizard and enter the Name and Caption properties as **Complete**.
3. Add a TextBox control to the Form Footer. Enter the Caption of the associated label as **Items checked out:** and set the Text Align property to Right. Change the control name to **txtCount**. Enter **=Count(*)** for the Control Source property.
4. Select all three fields in the Form Footer. In the Arrange tab of the ribbon, click the Stacked button. Click the Insert Right button twice to add two more columns.
5. Drag the `Complete` button the right-most cell in the first row and delete the empty row.
6. Drag the layout control to the top-left corner of the Form Footer. Resize the third cell to move the `Complete` button to the right.

www.it-ebooks.info

The final layout should look like Figure 7-29.

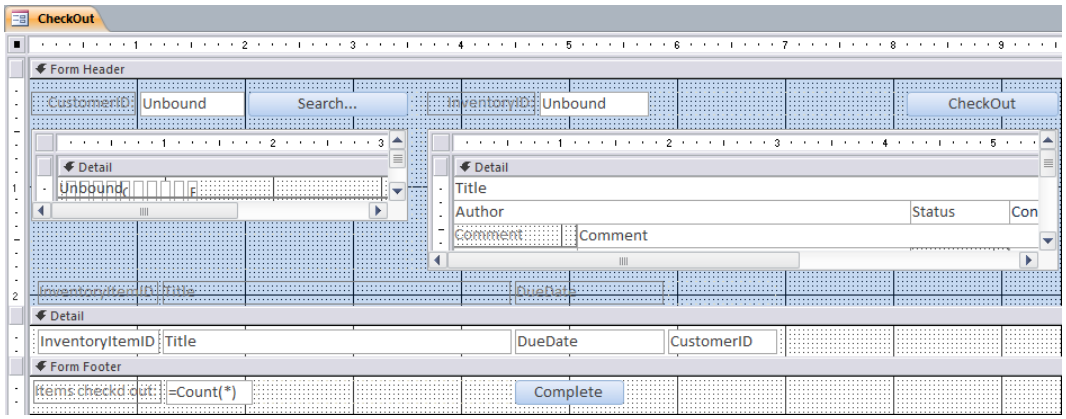


Figure 7-29. The layout of the CheckOut form

Implementing the CheckOut Logic

Now you're ready to implement the checkout process. At this point, the user can select a customer and an inventory item. That's all you need to create a Loan record. The first thing you'll need to do is hide the CheckOut button until both a CustomerID and InventoryItemID have been entered. Then the CheckOut button needs to be implemented so it inserts a record into the Loan table. Finally, you'll implement the Complete button that will finalize the process.

Enabling the CheckOut Button

Edit the VBA code for the txtInventoryItemID LostFocus event and add some code to hide or display the CheckOut button. Listing 7-4 shows the modified code and the new lines are shown in bold.

Listing 7-4. The modified LostFocus Event

```
InventoryItemLookup!lblTitle = ""

DoCmd.Requery "InventoryItemLookup"

If (Len(InventoryItemLookup!lblTitle) > 0) Then
    InventoryItemLookup.Visible = True
    If (Len(CustomerDisplay!lblName) > 0) Then
        CheckOut.Visible = True
        If (InventoryItemLookup!Status = "Available") Then
            CheckOut.Enabled = True
        Else
            CheckOut.Enabled = False
        End If
    Else
```


www.it-ebooks.info

```
        CheckOut.Visible = False
    End If
Else
    InventoryItemLookup.Visible = False
    CheckOut.Visible = False
End If
```

The new code makes the button visible if there is an inventory item displayed in the `InventoryItemLookup` subform. The button is then enabled if the item is available. Otherwise, the button is disabled. If the user scans an item that is not available for some reason, they'll see a grayed CheckOut button that will indicate the item cannot be checked out. The details in the subform such as `Status` and `Comment` should explain why it is not available.

Implementing the CheckOut Button

If the item is available to be loaned out, the user can click the `CheckOut` button. This will insert a record into the `Loan` table and perform some clean-up to prepare for additional records to be checked out.

Right-click the `CheckOut` button and click the *Build Event* link. At the prompt choose Code Editor. Enter the implementation of the `CheckOut` button using the code from Listing 7-5.

■ **Tip** The `OnClick` event is the default event for a command button so you can use the shortcut approach to creating an event handler. For other events, you should select the event in the Property Sheet and click the ellipses like you have done before.

Listing 7-5. *The Implementation of the `CheckOut` Button*

```
' Go to a new record
DoCmd.GoToRecord acDataForm, "CheckOut", acNewRec
CustomerID = txtCustomerID
InventoryItemID = txtInventoryItemID

' Cause the focus to move off the current record, which
' will cause it to be saved
DoCmd.GoToRecord acDataForm, "CheckOut", acNext
DoCmd.GoToRecord acDataForm, "CheckOut", acLast

' Reset the form controls
txtInventoryItemID = ""
DoCmd.Requery "InventoryItemLookup"
InventoryItemLookup.Visible = False
CheckOut.Visible = False
```

This code manipulates the form to insert a record rather than executing a SQL command directly. The `GoToRecord` method allows you to navigate through the records on the form just like the navigation controls at the bottom of the form. The last parameter of this function species the navigation option

www.it-ebooks.info

such as first, next, or last. In this case, you used the acNewRec constant, which goes to the record just after the last one. This is where you would add a new record to the form.

The form is now on a new, blank record. The code then sets the values of the two required fields, CustomerID and InventoryItemID. It uses the unbound controls in the Form Header, which contain the correct values. The GoToRecord method is called again to go to the next record. Just as with navigating a form manually, when you navigate off the current record, the updates are saved to the database. This causes the record to be inserted into the Loan table. GoToRecord is called one more time to make the row that was just added the current record.

The code then clears the txtInventoryItemId control. It also hides the InventoryItemLookup subform and the CheckOut button.

Implementing the Complete Button

After all the items have been entered, the user will use the Complete button to finalize the process. This clears the InProgress flag and then resets the form so it's ready for the next customer.

Right-click the Complete button and click the *Build Event* link. At the prompt, choose Code Editor. Enter the implementation of the Complete button using the code from Listing 7-6.

***Listing 7-6**. The Implementation of the Complete Button*

```
Dim sSQL As String

If (Len(txtCustomerID) > 0) Then
    sSQL = "UPDATE Loan SET Loan.InProgress = False WHERE Loan.InProgress=True " & _
           "AND Loan.CustomerID=" & txtCustomerID & ";"
    Application.CurrentDb.Execute sSQL, dbFailOnError
End If

txtCustomerID = ""
txtInventoryItemID = ""

Me.Requery

CustomerLookup.Visible = False
InventoryItemLookup.Visible = False
CheckOut.Visible = False

txtCustomerID.SetFocus
```

As I mentioned earlier, the Loan records that are added during this "session" need to be displayed on the form without including items that were previously checked out. You did this by only including records where the InProgress flag was true. This is the default value of this field so all new records have the InProgress flag set. Now that the checkout process is complete, you'll need to clear these flags. This is done by executing a SQL command.

The remainder of the code clears the input fields and hides the sub forms. The Me.Requery statement causes the main form to refresh its data. Since the txtCustomerID field has been cleared, it will not find any matching records and the form will be empty.

www.it-ebooks.info

Testing the Application

The CheckOut form is now complete, so let's test it.

1. Open the form in the Form View. Select a customer by entering the ID or using the Search feature.
2. Then enter an InventoryItemID. Try some that are available and some that are not to see if the CheckOut button is enabled correctly.
3. Then try clicking the CheckOut button and add a few items to the form. The form should look like Figure 7-30.

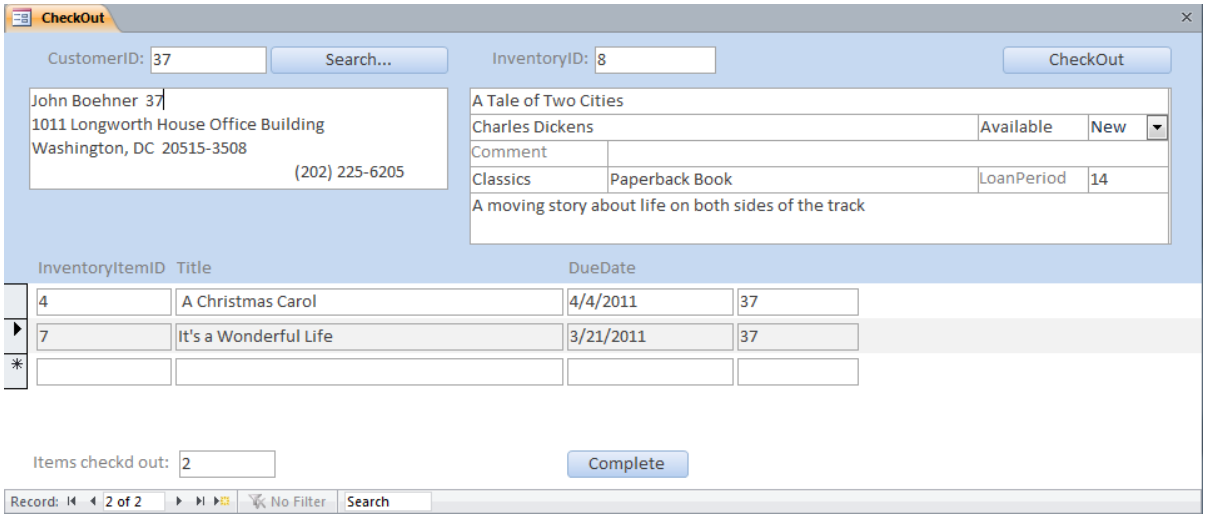


Figure 7-30. *The final CheckOut form with items added*

4. When you're done, click the Complete button and verify that the form is reset and ready for the next customer. You should also look at the Loan table and verify the records that were added by the form.

I have included the complete code file for the CheckOut form in Listing 7-7.

Listing 7-7. *The Complete Code Module for the CheckOut Form*

```
Option Compare Database

Private Sub CheckOut_Click()
    DoCmd.GoToRecord acDataForm, "CheckOut", acNewRec
    CustomerID = txtCustomerID
    InventoryItemID = txtInventoryItemID
```


www.it-ebooks.info

```
        ' Cause the focus to move off the current record, which
        ' will cause it to be saved
        DoCmd.GoToRecord acDataForm, "CheckOut", acNext
        DoCmd.GoToRecord acDataForm, "CheckOut", acLast

        ' Reset the form controls
        txtInventoryItemID = ""
        DoCmd.Requery "InventoryItemLookup"
        InventoryItemLookup.Visible = False
        CheckOut.Visible = False
    End Sub

    Private Sub Complete_Click()
        Dim sSQL As String

        If (Len(txtCustomerID) > 0) Then
            sSQL = "UPDATE Loan SET Loan.InProgress = False " & _
                   "WHERE Loan.InProgress=True " & _
                   "AND Loan.CustomerID=" & txtCustomerID & ";"
            Application.CurrentDb.Execute sSQL, dbFailOnError
        End If

        txtCustomerID = ""
        txtInventoryItemID = ""

        Me.Requery

        CustomerDisplay.Visible = False
        InventoryItemLookup.Visible = False
        CheckOut.Visible = False

        txtCustomerID.SetFocus
    End Sub

    Private Sub Search_Click()
       Dim sForm As String
       sForm = "CustomerSearch"

       ' Open the search form
       DoCmd.OpenForm sForm, acNormal, , , , acDialog

       ' If the form is not loaded, the user clicked the Cancel button
       If (IsLoaded(sForm)) Then
          txtCustomerID = Forms(sForm)!CustomerID
          DoCmd.Close acForm, sForm
       Else
          txtCustomerID = ""
       End If

       ' Force a refresh if the CustomerDisplay subform
```


www.it-ebooks.info

```
   txtCustomerID_LostFocus

End Sub

Private Sub txtCustomerID_LostFocus()
   CustomerDisplay!lblName = ""
   CustomerDisplay!lblAddress = ""
   CustomerDisplay!lblRegion = ""
   CustomerDisplay!lblEmail = ""
   CustomerDisplay!lblPhone = ""

   DoCmd.Requery "CustomerDisplay"

   If (Len(CustomerDisplay!lblName) > 0) Then
      CustomerDisplay.Visible = True
   Else
      CustomerDisplay.Visible = False
   End If
End Sub

Private Sub txtInventoryItemID_LostFocus()
    InventoryItemLookup!lblTitle = ""

    DoCmd.Requery "InventoryItemLookup"

    If (Len(InventoryItemLookup!lblTitle) > 0) Then
        InventoryItemLookup.Visible = True
        If (Len(CustomerDisplay!lblName) > 0) Then
            CheckOut.Visible = True
            If (InventoryItemLookup!Status = "Available") Then
                CheckOut.Enabled = True
            Else
                CheckOut.Enabled = False
            End If
        Else
            CheckOut.Visible = False
        End If
    Else
       InventoryItemLookup.Visible = False
       CheckOut.Visible = False
    End If
End Sub
```

Summary

In this chapter you created a form that allows items to be loaned out to a customer. Along the way, you implemented a number of techniques that you will undoubtedly find an application for in many future projects. Some of these will also be used in subsequent chapters of this book.

www.it-ebooks.info

The new design approaches that you implemented include:

- Implementing a search form as a modal dialog
- Creating a reusable data-bound subform
- Manipulating the form records through VBA code
- Executing a SQL command

You also used the layout control and the Layout View to format several forms.

www.it-ebooks.info

CHAPTER 8

Creating a Customer Admin Form

In the previous chapter, you created a `CheckOut` form that demonstrated some advanced techniques with Access forms. Now, I'll continue along the same theme, digging a little bit deeper into how forms work and explaining ways to configure them. I'll also demonstrate the TabControl, which is useful for organizing a lot of information.

In this chapter, you'll build a customer administration form that can be used to create and update customer records, show a history of what they have previously checked out, and display the items that are currently due. This form will also allow you to renew items that are checked out. You will reuse some of what was developed in Chapter 7, including the `CustomerSearch` form. Hopefully, I will instill in you the benefits of designing reusable forms.

Building the Customer Profile Tab

The form that you'll build will use a TabControl and the first page will show the customer's name and contact information. It will also allow you to update this information and to set up a new customer. Once you have that working, I'll show you how to add other features in subsequent pages.

You'll start by creating a stand-alone form for adding and updating customer information. You will then drop this on the main form along with the existing `CustomerDisplay` form. You'll then need to write some code to tie these together.

Creating a CustomerUpdate Form

In the last chapter, you created a `CustomerDisplay` form that presented the customer's name, address, email, and phone in a compact form that looks much like a mailing label. This was handy to drop on a form without taking up much space. However, this was not designed to allow updates; you'll need a different form for that, which you'll design now.

You'll use the standard form template to create a new form based on the `Customer` table.

1. Select the `Customer` table in the Navigation pane then click the Form button in the Create tab of the ribbon. The initial form, which is displayed in the Layout View, should look like Figure 8-1.

www.it-ebooks.info

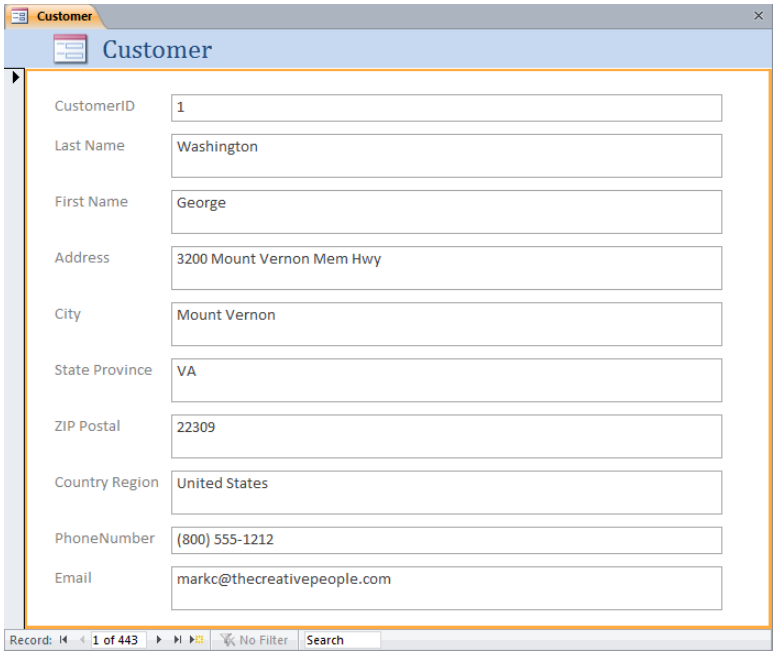


Figure 8-1. *Initial* `CustomerUpdate` *form*

2. Select one of the data-bound controls and drag the right edge to the left to shrink the width to about 1 inch.

3. Finally, add two more columns using the Insert Right button on the Arrange tab of the ribbon.

The `CustomerID` field will not be displayed, as I discussed in Chapter 6. This surrogate key is not meaningful to the end user, so you won't clutter the form with this information. However, you'll need the `CustomerID` control on the form, because it will be used to link this form to a parent form. You will hide this control and delete the associated label. Even though the control is not visible, it still takes up a cell in the layout control. Access will not allow you to put two controls in the same cell. To resolve this, you'll need to remove this control from the layout control. You can then move it to wherever you want it, even on top of another control. It doesn't really matter, because it is not visible.

■ **Note** You cannot remove a control from the layout while in the Layout View. You'll need to first switch to the Design View.

---

1. Switch to the Design View. Select the `CustomerID` control. From the Arrange tab in the ribbon, click the Remove Layout button, as shown in Figure 8-2.

www.it-ebooks.info

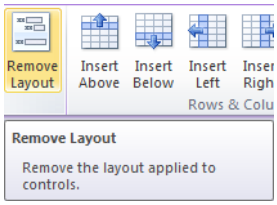


Figure 8-2. *Removing the* `CustomerID` *control from the layout*

2. When you removed the control, its associated label is also removed. Because all the cells on that row are now empty, the row is deleted, moving all the controls up one row. This can be a little confusing, because the `CustomerID` control is still where it was but now the `LastName` control is on top of it. Drag the `LastName` control to one of the empty cells on the right as shown in Figure 8-3 and you'll be able to see the `CustomerID` control.



Figure 8-3. *Temporarily moving the* `LastName` *control*

■ **Tip** At run time, the `CustomerID` control is not visible. However, when designing the form, you'll need to be able select it and modify its properties. What I suggest with these "floating" controls is that you resize them to make them small enough to completely fit inside of one of the layout cells with space around it. Then bring the control to the front so other controls are not hiding it. With this approach, you can see and manipulate the control without hiding any of the visible controls.

---

3. Select the layout control and drag it down about half an inch to expose the `CustomerID` control. Delete the associated label for the `CustomerID` control.
4. Shrink the `CustomerID` control to about half of its original height and width. In the Format tab of the Property Sheet, set its Visible property to No.
5. Right-click the `CustomerID` control and click the *Position* ➤ *Bring to Front* links, as shown in Figure 8-4. This will keep other controls from hiding it.

www.it-ebooks.info

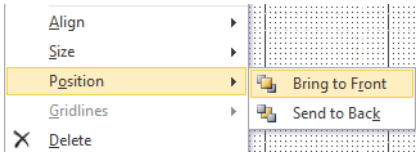


Figure 8-4. *Bringing the* `CustomerID` *control to the front*

6. Drag the `CustomerID` control to the bottom of the form. You'll move it to a permanent location once the form layout is complete.
7. Switch back to the Layout View.
8. Change some of label captions (these can be edited directly on the form):
   - Country Region: **Country**
   - State Province: **State**
   - Zip Postal: **Postal**
   - PhoneNumber: **Phone**
9. Select all the cells in the layout and, in the Format tab of the Property Sheet, set the Height property to **.22."**
10. Select all the label controls and set their Text Align property to Right.
11. Drag the entire layout control to the top-left corner of the form.
12. Arrange the controls and size the cells, as shown in Figure 8-5.

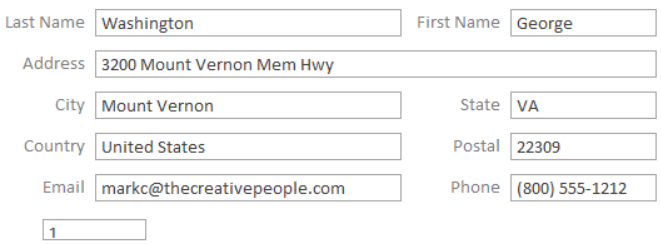


Figure 8-5. *The modified* `CustomerUpdate` *form layout*

13. When you try to drag the `CustomerID` control on top of one of the cells in the layout, Access will try to insert a row in the layout. You'll see an orange line appear in between two rows indicating where a row will be created to accept the control. This is not what you want. Instead, drag it near the layout, just to the right of the `Address` control. Click the left edge of the `CustomerID` control and drag it to the left until it is well inside the `Address` control. Then click the right edge and shrink the control back to its original size.

www.it-ebooks.info

14. Finally, shrink the height and width of the form to remove any unused space. You may notice that it won't let you shrink the width. You can't make the form smaller than the collection of controls placed on it. The Form Header has a Label control that is as large as the original form; you'll need to first resize this to make it smaller than the layout control in the Detail section.

15. Save the form and enter the name **CustomerUpdate** when prompted.

Open the form in Form View, which should look like Figure 8-6.

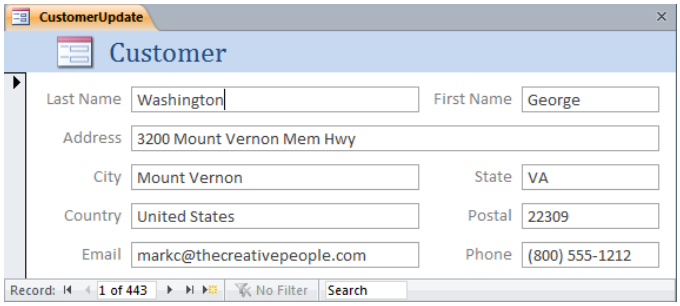


Figure 8-6. *Completed design of the* `CustomerUpdate` *form*

Configuring the Form Controls

In Chapter 7, I gave some instructions for setting various form properties without explaining what they were for. I'll now explain the form controls that Access generates for you, how they work, and how to configure them using the form Property Sheet. Figure 8-7 shows the `CustomerUpdate` form, and I have circled and numbered the controls.

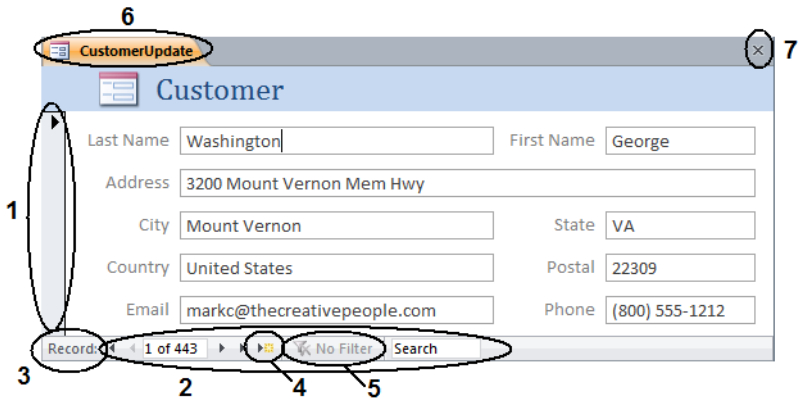


Figure 8-7. *Identifying the form controls*

www.it-ebooks.info

1. Record Selector: The record selector serves two purposes. In the Datasheet View, the record selector indicates which row is currently selected. In Form View, this is not needed, because only a single record is displayed. However, the second purpose is to indicate if the data has changed. If any of the fields on the selected record have changed and the change has not been saved yet, the arrow will change to a pencil, indicating a change has been made. You can also click on the record selector to force the changes to be saved. The `CustomerDisplay` form you created in Chapter 7 was read-only so the record selector was unnecessary. To remove this, set the Record Selectors property to No.
2. Navigation Controls: The navigation controls are at the bottom of the form. The Navigation Buttons property determines if this row of controls is visible or not. The "1 of 443" text lets you know there are 443 records currently available and you're on the first one. There are buttons to go to the first, previous, next, and last records. I've circled the area where most of the controls are, but the entire row is used for navigation functions. If the Navigation Buttons property is set to No, the areas marked as 3, 4, and 5 are not displayed either.
3. Navigation Caption: The word "Record" is defined by the Navigation Caption property. If no value is specified, the default text, "Record," is used. If you wanted this to be "Customer," just enter that text in the Navigation Caption property.
4. New Record: The control just to the right of the Last Record button is used to create a new record. This is disabled if the Allow Additions property is set to No.
5. Filter: Access allows the end user to define ad-hoc filters, which is a really nice feature. The control here indicates if a user-defined filter has been defined. If not, the control will display "No Filter." If a filter has been applied, the text will be "Filtered." You can temporarily toggle a filter, which will display all the records. In this case, the text on this control will be "Unfiltered." If a user-defined filter has been defined, you can toggle between Filtered and Unfiltered by clicking on this control. You can turn off the filtering feature by setting the Allow Filters property to No.

■ **Note** Filters defined at design time or programmatically are not affected by the Allow Filters property. You can turn this feature off to prevent the user from using filters and still define a default filter at design time and apply a filter using code.

---

6. Caption: You can set the Caption property to specify what text will appear in the form tab. If none is specified, the form will use the contents of the Name property. In the Design View, this is ignored and the Name property is used on the tab heading.

www.it-ebooks.info

7. Close: The close button at the top-right corner of the form closes the form. You can disable this by setting the Close Button to No. This will prevent you from closing the form in any view except the Design View.

■ **Note** When a form is used as a subform, neither the Caption nor the Close Button properties are used. These do not apply to child forms.

Now with this understanding, you'll configure the `CustomerUpdate` form that you just created. The record selector would be useful for indicating when there are unsaved changes. It also works as a Save button. Just like the `CustomerDisplay` form, this form will be configured to show the record associated with the parent form. So there's no need for the navigation controls.

Set the following properties; the remaining form properties can be left at their default values.

- Record Selectors: Yes
- Navigation Buttons: No
- Filter: `[CustomerID] = CustomerID`
- Filter On Load: Yes

Finally, you'll need to remove the Form Header that was generated by the template. It contains a logo image and a Label control. Delete both of these controls and the drag the Detail section up to shrink the Form Header so there is no space between them. Save the form and switch to the Form View. The form should look like Figure 8-8.



Figure 8-8. *The final* `CustomerUpdate` *form*

Creating the CustomerAdmin Form

Now you're ready to create the `CustomerAdmin` form and implement the first page. You'll provide a search facility in the Form Header just like the `CheckOut` form. When a customer is selected, the existing `CustomerDisplay` form will be included in the `Profile` page to show their contact information. This page will also have a `Modify` button. If the user needs to change this information, the `Modify` button will show the new `CustomerUpdate` form that they will use to update the data.

www.it-ebooks.info

If no customer was selected, a `New` button will appear that will allow the user to create a new `Customer` record. It will use the same `CustomerUpdate` form to enter the details for the new customer. When the update is complete, the user will click the `Finish` button, which will hide the `CustomerUpdate` form and refresh the `CustomerDisplay` form.

Designing the Form Header

In the Form Header you'll add a TextBox control and a `Search` button to enter or find the desired customer, just like you did with the `CheckOut` form in the last chapter.

1. From the Create tab of the ribbon, click the Blank Form button. Switch to the Design View and then right-click the form and click the *Form Header/Footer* link. Expand the Form Header to about 1-inch high.
2. Add a TextBox control to the Form Header and change the Name property to **txtCustomerID**. Set the Caption of its associated label to **CustomerID:**. Set the Text Align property for the label to Right.
3. Add a command button to the Form Header and cancel the Command Button Wizard. Change the Name property to **Search** and set the Caption property to **Search…**

■ **Tip** You can disable the wizards, such as the Command Button Wizard, when adding controls to the form. Sometimes they're helpful, and other times they're not. It's pretty easy to just cancel them if you don't want a wizard for a particular control. But you can also turn these off. In the Design tab of the ribbon, click the dropdown icon in the Controls section, which will expand this area and show all the available controls. Just click the *Use Control Wizards* link as shown in Figure 8-9. To enable the wizards, repeat this step to toggle them back on.

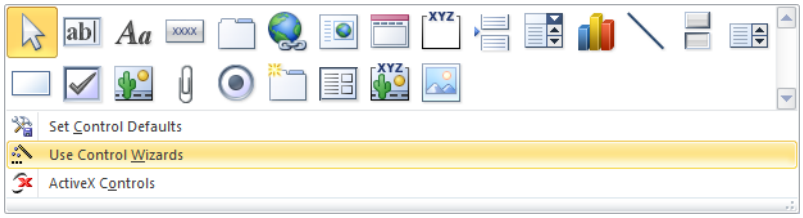


Figure 8-9. *Disabling the control wizards*

4. Select all three controls in the Form Header. From the Arrange tab of the ribbon, click the Stacked button. This should create two columns.
5. Click the Insert Right button on the ribbon to create a third column.
6. Drag the command button to the last column of the first row. Delete the empty second row.

www.it-ebooks.info

7. Select the entire layout and drag it to the top-left corner of the Form Header. Shrink the Form Header so there is no empty space beneath these controls.

The Form Header layout should look like Figure 8-10.

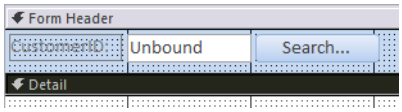


Figure 8-10. *The layout of the Form Header*

Using a Tab Control

A tab control in Access works basically the same as in any .NET application. Each page is like its own mini-form. You design the controls on each page independently from the other pages.

■ **Note** In a tab control, you will have multiple pages of controls. The *tab* is the small portion at the top that contains the label describing the data for that page. The object below the tab is called the *page*. Think of a file folder, where the tab is the part that sticks out where you add a label to identify it. Often the *term* tab is used to refer to the *page* as well. We can generally infer based on the context what is being referred to. I will call them pages; if you're used to them being called tabs, I hope this will not be confusing.

One thing to keep in mind when designing a tab control is that the pages all need to be the same size. More specifically, the size of all of the pages will be defined by the size of the largest one. As you plan what should be on each page try to keep this size rule in the back of your mind. This is a relatively minor point; it is far more important to keep the pages organized logically. Don't put a control on a page just because there is more room on that page. Let's get started.

1. From the Design tab of the ribbon, click the Tab Control button. Then draw a rectangle in the Detail section that is about the size of the entire section. This will create a TabControl with two tabs (pages). The label on the tab will be, initially, the name of the Page control associated with the tab. In my case, the first one is `Page11`. (I must have created 10 pages before I added this one.)
2. Click just below the tab to highlight a rectangle around the page. The Property Sheet should indicate that the `Page11` object is selected (or whatever your page was named).
3. In the Format tab, change the Caption property to **Profile**. In the Other tab, change the Name property to **Profile**.

www.it-ebooks.info

■ **Caution** When adding a control to a page, make sure that the page is selected first. An orange rectangle that is just below that tabs is used to show the page is selected. If there are already controls on the page, you can just select one of these controls. That will also put the page in focus. If the orange rectangle goes around the tabs then you have selected the TabControl not a particular page on that control. If the control is not included on a specific page, it won't be really obvious…until you go to another tab and you notice that the control is still visible. Let's just say I've learned from experience.

Designing the Profile Page

The `Profile` page will contain the `CustomerDisplay` form that you implemented in Chapter 7 and the `CustomerUpdate` form that you just created. You will also need a few command buttons to tie everything together.

1. Make sure the control wizards are enabled and the `Profile` page is selected. Click the Subform button in the ribbon, and then click inside the `Profile` page. In the Subform Wizard, select the `CustomerDisplay` form as shown in Figure 8-11 and click the Next button.

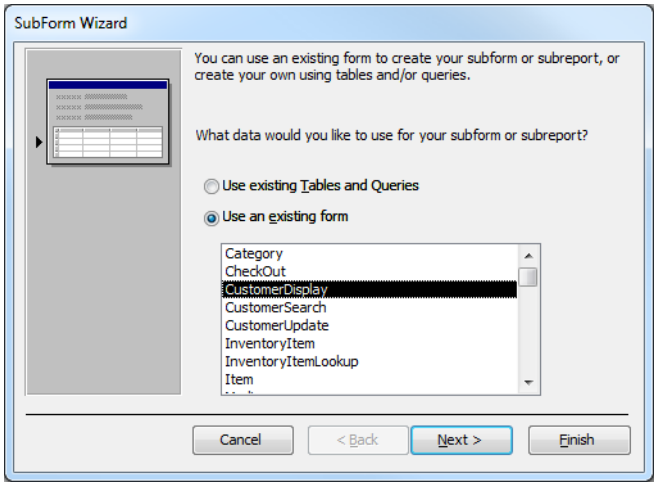


Figure 8-11. Selecting the `CustomerDisplay` form

2. In the next dialog box, use the default name, **CustomerDisplay** and click the Finish button. The wizard also generated an associated label; delete this.

3. Select the Subform control and, in the Arrange tab of the ribbon, click the Stacked button. Because there is only one control, the layout will have a single row and column. Click the Insert Below button twice to add two rows. Click the Insert Right button once to add another column.

www.it-ebooks.info

4. Merge both cells in the last row into a single cell.
5. From the Design tab of the ribbon, click the Subform button again and then click on the `Profile` page. In the Subform Wizard, select the `CustomerUpdate` form as shown in Figure 8-12 and then click the Next button.

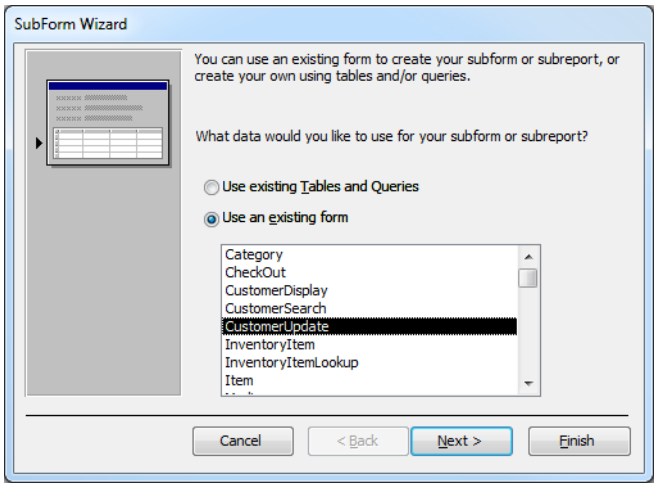


Figure 8-12. *Selecting the* `CustomerUpdate` *form*

6. In the next dialog box, use the default name, **CustomerUpdate** and click the Finish button. Again, delete the associated label.
7. Drag this Subform to the bottom-left cell of the layout control.
8. Switch to the Layout View and size the cells to fit the subforms correctly.

■ **Tip** The cell for the `CustomerDisplay` subform should already be sized correctly. To adjust for the `CustomerUpdate` subform without affecting the `CustomerDisplay` subform, increase the vertical size of the last row and the horizontal size of the top-right cell.

---

9. Drag the entire layout to the top-left corner of the `Profile` page.
10. You will add three command buttons to the row of cells between the two subforms. Select the left cell of this row and click the Split Horizontally button. This will split the cell into two cells. Select one of these and click the Split Horizontally button again. Now you'll have three cells where there used to be one.

www.it-ebooks.info

11. Change the width of these three cells so they are about the same size. This will adjust the size of the subforms so you'll need to keep an eye of that as well.
12. Create three command buttons, one in each cell. Enter the following Name and Caption properties:
    - `ModifyCustomer`: **Modify...**
    - `NewCustomer`: **New...**
    - `Finish`: **Finish**
13. You may need to adjust the height of the row containing the command buttons.
14. Select all the controls (the two subforms and the three buttons) and in the Format tab of the Property Sheet, set the Visible property to No. You will display these controls with VBA code as appropriate.
15. Save the form and enter the name **CustomerAdmin** when prompted.

The form layout should look like Figure 8-13.

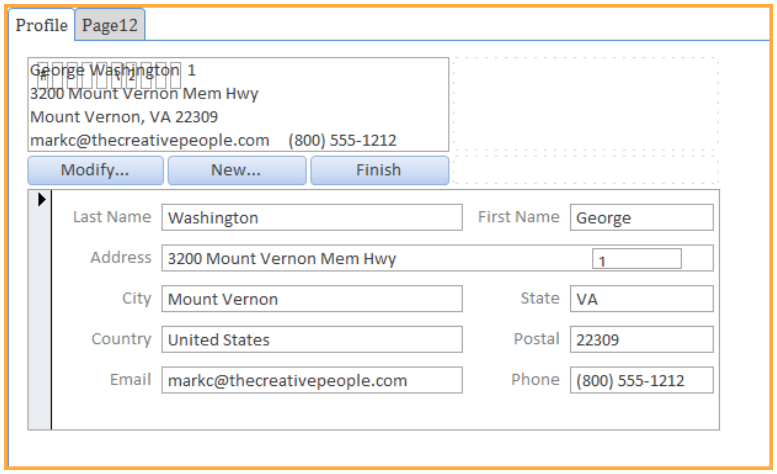


Figure 8-13. *The layout of the* `Profile` *page*

Implementing the Connecting Code

The last step of making this all work is implementing the code that connects the forms. First you'll set up the links between the parent page and the sub forms. Then you'll implement several event handlers.

1. Select the `CustomerDisplay` subform. In the Data tab of the Property Sheet, enter **txtCustomerID** for the Link Master Fields property and **CustomerID** for the Link Child Fields property.

www.it-ebooks.info

2. Select the `CustomerUpdate` subform and enter the exact same values for these two properties. You probably noticed that, as soon as you did this, the data disappeared from the subforms. That's because there is no value defined for the `txtCustomerID` control in the Form Header. Now you just need to implement the VBA code. Most of this should be familiar, because it is similar to what you implemented in Chapter 7.

■ **Tip** So far, to implement an event handler you selected the Event Procedure option on the Property Sheet for the appropriate event and then clicked the ellipses, which generated the method and displayed the VBA editor. As an alternative, you can right-click the command button and choose the *Build Event* link, which accomplished the same thing. If you know the name of the events for which you want to generate handlers, you can go directly to the VBA code. From the Design tab of the ribbon, click the View Code button. This will display the VBA editor and generate a code file for the form, if necessary. Then you can create the event handlers in VBA without having to enter anything in the Property Sheet.

---

3. From the Design tab of the ribbon, click the View Code button. This will launch the VBA editor and display an empty file for this form. Enter the methods shown in Listing 8-1.

Listing 8-1. *Event Handlers for the* `Profile` *Page*

```
Private Sub txtCustomerID_LostFocus()
   CustomerDisplay!lblName = ""

   DoCmd.Requery "CustomerDisplay"

   If (Len(CustomerDisplay!lblName) > 0) Then
      CustomerDisplay.Visible = True
      ModifyCustomer.Visible = True
      NewCustomer.Visible = False
   Else
      CustomerDisplay.Visible = False
      NewCustomer.Visible = True
      NewCustomer.SetFocus
      ModifyCustomer.Visible = False
   End If
End Sub

Private Sub Search_Click()
   Dim sForm As String
   sForm = "CustomerSearch"

   ' Open the search form
   DoCmd.OpenForm sForm, acNormal, , , , acDialog
```


www.it-ebooks.info

```
    ' If the form is not loaded, the user clicked the Cancel button
    If (IsLoaded(sForm)) Then
       txtCustomerID = Forms(sForm)!CustomerID
       DoCmd.Close acForm, sForm
    Else
       txtCustomerID = ""
    End If

    ' Force a refresh if the CustomerDisplay subform
    txtCustomerID_LostFocus

End Sub

Private Sub ModifyCustomer_Click()
    CustomerUpdate.Visible = True
    Finish.Visible = True
    Finish.SetFocus
    ModifyCustomer.Visible = False
End Sub

Private Sub NewCustomer_Click()
    CustomerUpdate.Visible = True
    Finish.Visible = True
    Finish.SetFocus
    NewCustomer.Visible = False
End Sub

Private Sub Finish_Click()
    ModifyCustomer.Visible = True
    CustomerUpdate.Visible = False
    ModifyCustomer.SetFocus
    Finish.Visible = False

    ' Force a refresh
    txtCustomerID = CustomerUpdate!CustomerID
    txtCustomerID_LostFocus
End Sub
```

The implementation of `txtCustomerID_LostFocus` is almost identical to the version from Chapter 7 except that it has to take care of showing or hiding the `Modify` and `New` buttons. If a customer was found, it shows the `Modify` button and if not, the `New` button is shown. In either case, only one is visible; whichever one is shown, the other is hidden.

The implementation of `Search_Click` is identical to what you implemented in Chapter 7. The last three methods handle the `OnClick` event of the three buttons. The first two, `ModifyCustomer_Click` and `NewCustomer_Click` are basically the same. They show the `CustomerUpdate` form and the `Finish` button and hide themselves.

The implementation of `Finish_Click` is a little more interesting. It hides the `CustomerUpdate` form and itself and shows the `Modify` button. That puts everything back like it was before the first `Modify` button was clicked. The last part of this method gets the `CustomerID` from the `CustomerUpdate` form and updates the TextBox control in the Form Header. This is done to handle new records. The ID is not

www.it-ebooks.info

generated until the record is saved. So, once the insert has completed, the ID is retrieved and stored in the header. The `txtCustomerID_LostFocus` handler is also called to refresh the `CustomerDisplay` form.

Notice that there is no code to save the record. Remember, when you move the focus off of the updated record, the changes are automatically saved. The `Finish` button is on the parent form, so, when this is clicked, the focus moves off the `CustomerDisplay` form and the record is saved.

Testing the Profile Page

Save the code file and the `CustomerAdmin` form. Test it out by switching to the Form view. After selecting a customer and clicking the `Modify` button, the `CustomerUpdate` form appears, as shown in Figure 8-14.

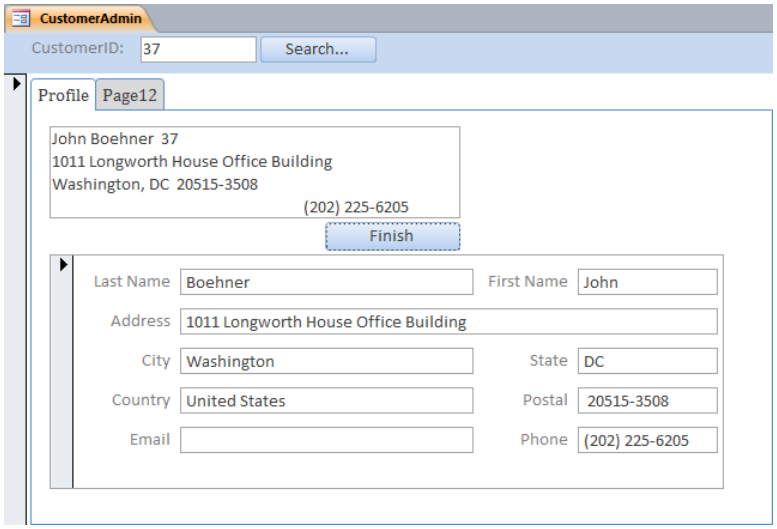


Figure 8-14. *The implemented* `Profile` *page*

If you change anything, the arrow changes to a pencil. You can undo your changes by pressing the Esc key. To save your changes, click on the `Finish` button. The `CustomerUpdate` form will be hidden and the `CustomerDisplay` form will be refreshed with the updated data.

Now try creating a new customer. Clear the `CustomerID` field, and tab off of the field. Because there will be no matching customer, the form will be blank except for the `New` button. Click the `New` button and an empty `CustomerUpdate` will be displayed as shown in Figure 8-15.

www.it-ebooks.info

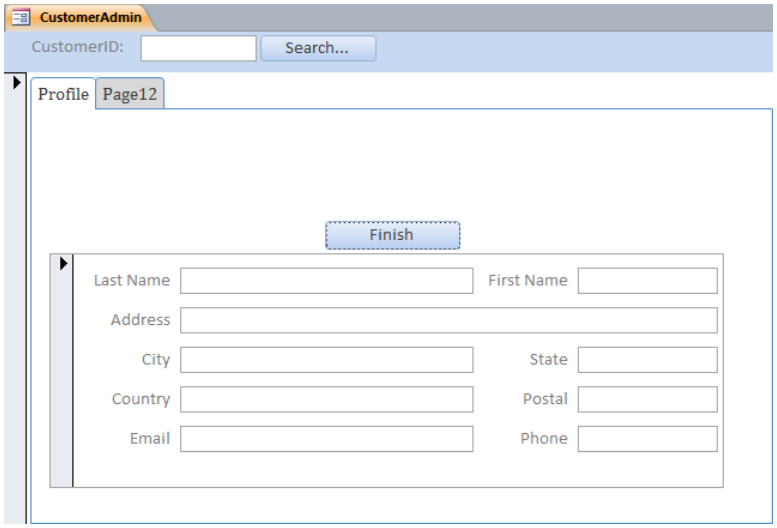


Figure 8-15. *Adding a new customer*

Enter a name and contact info and click the `Finish` button. The form should be updated to show this information in the `CustomerDisplay` form as shown in Figure 8-16.

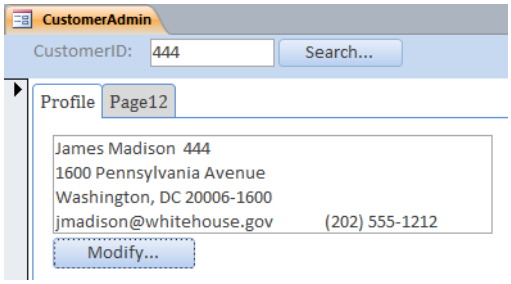


Figure 8-16. *Updating the* `CustomerDisplay` *form*

Building the Items on Loan Tab

The next page in the `CustomerAdmin` form will list all the items that are on loan for that customer. They can select one to see more details and have the option to renew the item. This is a perfect application for a Split Form, which you used in Chapter 6. The only problem is, you cannot use a Split Form as a subform. To work around this problem, you'll create two subforms on this page; a Datasheet View to list the items and a Form View to display the selected record. Then you'll need to keep the two in sync with VBA code.

www.it-ebooks.info

Enhancing the LoanDetails Query

In the last chapter, you created a `LoanDetails` query to support the `CheckOut` form. You'll use the same query to retrieve the items that are currently on loan. For the purposes here, you'll need to also join the `Category` and `Media` tables and add a few more fields to the query results.

Open the `LoanDetails` query in the Design View. Click the Show Table button in the ribbon and add the following tables to the query:

- `Category`
- `Media`

The join relationships to these tables should be set up for you. The query design should look like Figure 8-17.

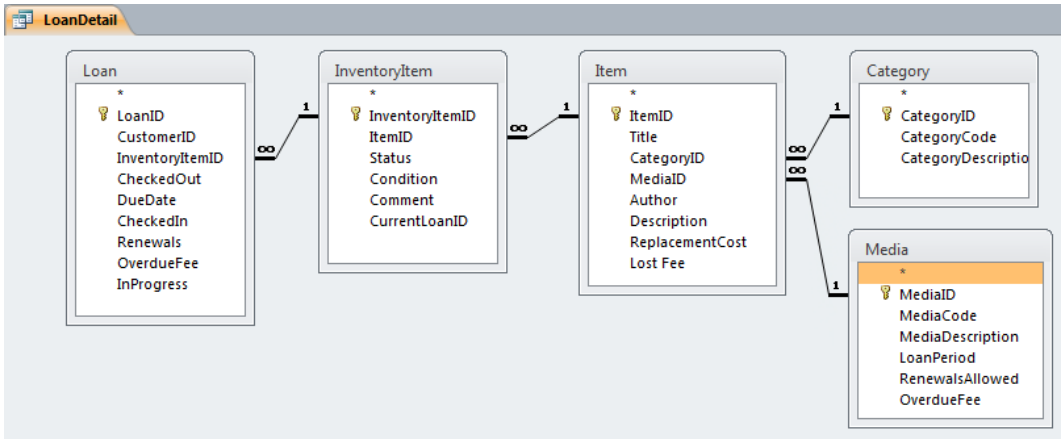


Figure 8-17. *The modified* `LoanDetails` *query design*

Double-click on the following fields to add them to the query results:

- `Loan.LoanID`
- `Loan.CheckedIn`
- `Loan.Renewals`
- `Loan.OverdueFee`
- `Item.ItemID`
- `Item.Author`
- `Item.Description`
- `Category.CategoryDescription`
- `Media.MediaDescription`

www.it-ebooks.info

- Media.LoanPeriod
- Media.RenewalsAllowed

You should also define the sort order for this query. In the CheckedOut field, select Descending for the Sort property. This will return the items in the order they were checked out with the most recent first. Save the query changes and close the LoanDetail query.

Designing a CustomerLoan Form

The next step is to design a form that will be used for the two subforms. You'll actually design a single form that will be used for both. One copy of the form will use the Form View and the other will use the Datasheet View. In Chapter 6 I told you that the layout of the Datasheet View was independent of the Form View. Now, you'll have a good opportunity to see that in action.

Creating the Initial CustomerLoans Form

You'll start by creating a standard data-bound form using the Form button in the ribbon. This is the quickest way to generate a simple form.

1. Select the LoanDetail query in the Navigation pane and click the Form button in the Create tab of the ribbon. This will generate a form with a label and data-bound control for each field in the query.
2. Select the InProgress control and delete it, which should also remove its associated label. You will not need this field on either version of this form.
3. There are several fields that will not be shown on the Form View, but are needed for the subsequent form logic that you will write. Just like with the CustomerID field in the CustomerUpdate for that you implemented earlier, you will remove these from the layout and make them invisible.
4. First, shrink the two columns in the layout to make the size of the form a little easier to work with. Then switch to the Design View. Select the following controls and their associated labels:
   - CustomerID
   - InventoryItemID
   - LoanID
   - ItemID
5. In the Arrange tab of the ribbon, click the Remove layout button. Select the layout and drag it to the right to reveal the controls that were removed from the layout. Delete their associated labels.

www.it-ebooks.info

6. The `CustomerID` and `InventoryItemID` controls were generated as ComboBox controls. You'll need to replace them with TextBox controls. Delete both ComboBox controls, and then add two TextBox controls. Delete the associated labels. Set the Name property of the new controls to **CustomerID** and **InventoryItemID**. In the Data tab of the Property Sheet, set the control source to the corresponding field in the query.
7. Select all four data-bound controls and set their Visible flag to No.

Arranging the Controls

Now let's work on arranging the remaining controls. This will be used as a continuous form so you should make efficient use of space to create a form that is as small as possible.

1. First, change some of the label captions as follows:
   - `Renewals:` Renewals
   - `RenewalsAllowed:` Allowed Renewals
2. Then delete the labels for the following controls (the text in these fields is fairly self-explanatory):
   - `Title`
   - `Author`
   - `Description`
   - `CategoryDescription`
   - `MediaDescription`
3. Switch to the Layout View. You will need a total of four columns in the layout. Select one of the data-bound controls and click the Insert Right button (in the Arrange tab of the ribbon) twice.
4. Drag the `CheckedOut` and `DueDate` controls and their labels to empty cells on the lower-right part of the layout.
5. Merge all the cells in the top row into one cell and drag the `Title` control to this cell.
6. Merge the first three cells in the second row and drag the `Author` field to this row.
7. Drag the `CheckedOut` control and its label to the second and first cells of the third row. Drag the `DueDate` control and its label just below this.
8. Drag the `CategoryDescription` control to the right of the `CheckedOut` control. Drag the `MediaDescription` control to the right of the `DueDate` control.
9. The `CheckedIn` and `Renewals` controls can stay where they are. Drag the `OverdueFee` and its label down one row. Drag the `RenewalsAllowed` control and its label just above the `OverdueFee` control.

www.it-ebooks.info

10. Drag the `LoanPeriod` control to the right of the `Renewals` control, and drag its label just above it.
11. Merge the cell that has the `Description` control with all the other cells on that row.
12. Merge the empty cells in the rightmost column into a single cell.

Modifying the Form Layout

Lastly, you'll add some fine tuning to the layout of the form and configure the filter and navigation properties.

1. Select the entire layout, click the Control Padding button in the ribbon, and click the None link. While everything is still selected, from the Format tab of the Property Sheet, set the Height property to **.22."**
2. Select all the labels in the first column and set the Text Align property to Right.
3. Drag the entire layout to the top-left corner of the form.
4. Remove the controls in the Form Header, which should remove the Form Header as well.
5. Create a command button and place it in the cell just below the `LoanPeriod` control. Set the Name and Caption properties to **Renew**. This form is designed to display a single record and should not allow any changes. You won't need the record selector or the navigation controls.
6. Set the following form properties in the Property Sheet:
   - Allow Additions: No
   - Allow Deletions: No
   - Allow Edits: No
   - Allow Filters: No
   - Record Selectors: No
   - Navigation Buttons: No
7. Save the form and enter the name **CustomerLoans** when prompted.

Switch to the Form View. The form should look like Figure 8-18.

www.it-ebooks.info

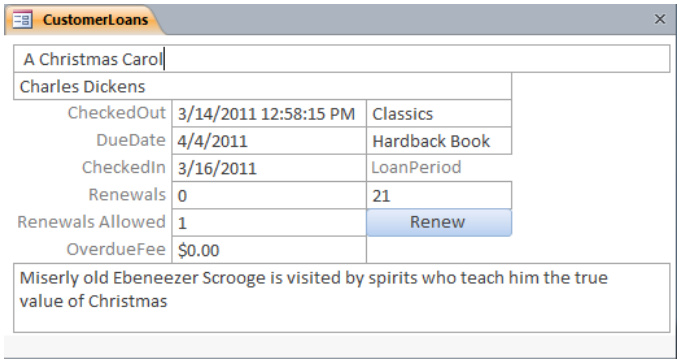


Figure 8-18. *The* `CustomerLoans` *form layout*

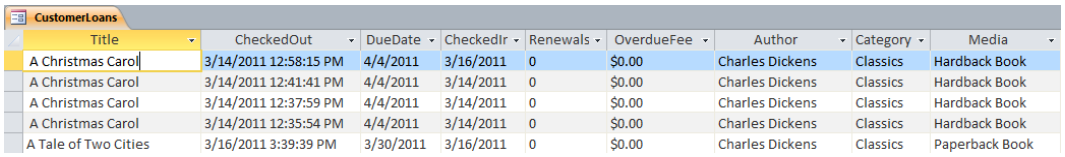
Configuring the Datasheet View

Now you'll configure the layout of the Datasheet View, which should not affect the Form View that you just designed.

1. Go to the Design View. In the Property Sheet select the Form object and the Format tab.
2. Set the Allow Datasheet View property to Yes and switch to the Datasheet View.
3. To change the order of the column, select a column and then drag it to the desired position. Using this method, put first nine fields in the following order. The rest of the fields should come after these nine.
    - `Title`
    - `CheckedOut`
    - `DueDate`
    - `CheckedIn`
    - `Renewals`
    - `OverdueFee`
    - `Author`
    - `CategoryDescription`
    - `MediaDescription`
4. For the remaining fields the order is not important. Shrink the column width down to as small as possible.
5. Select the `CategoryDescription` column. In the Other tab of the Property Sheet set the Datasheet Caption property to **Category**. In the same way, select the `MediaDescription` column and set the Datasheet Caption property to **Media**.

www.it-ebooks.info

The Datasheet View should look like Figure 8-19.



CustomerLoans

Title	CheckedOut	DueDate	CheckedIr	Renewals	OverdueFee	Author	Category	Media
A Christmas Carol	3/14/2011 12:58:15 PM	4/4/2011	3/16/2011	0	$0.00	Charles Dickens	Classics	Hardback Book
A Christmas Carol	3/14/2011 12:41:41 PM	4/4/2011	3/14/2011	0	$0.00	Charles Dickens	Classics	Hardback Book
A Christmas Carol	3/14/2011 12:37:59 PM	4/4/2011	3/14/2011	0	$0.00	Charles Dickens	Classics	Hardback Book
A Christmas Carol	3/14/2011 12:35:54 PM	4/4/2011	3/14/2011	0	$0.00	Charles Dickens	Classics	Hardback Book
A Tale of Two Cities	3/16/2011 3:39:39 PM	3/30/2011	3/16/2011	0	$0.00	Charles Dickens	Classics	Paperback Book

Figure 8-19. *The `CustomerLoans` form Datasheet View*

Save the form changes and go back to the Form View to make sure that that looks as expected.

Designing the Items on Loan Page

Now you're ready to design the second page of the `CustomerAdmin` form. This will be a fairly simple matter of dropping two copies of the `CustomerLoans` form onto this page. Then you'll write a little VBA code to connect everything together.

1. Close the `CustomerLoans` form and open the `CustomerAdmin` form in the Layout View.
2. Click the second tab (mine is called `Page12`, yours may be a different name). In the Other tab of the Property Sheet, change the Name property to **OnLoan**. In the Format tab, set the Caption property to **Items on Loan**.
3. In the Design tab of the ribbon, click the Subform button and then click on the `OnLoan` page. This will add a Subform control and an associated label. Delete the label. Select the Subform control and change its Name property to **CustomerLoans**.
4. In the Data tab of the Property Sheet, select the `CustomerLoans` form for the Source Object property.

■ **Note** These instructions assume you have the wizards turned off. If the wizards are on, just select the `CustomerLoans` as the source form and use the default name.

---

5. Enter **txtCustomerID** in the Link Master Fields property and **CustomerID** in the Link Child Fields property. The Property Sheet will look like Figure 8-20.

www.it-ebooks.info

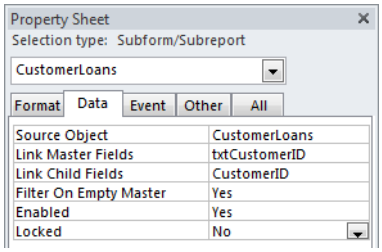


Figure 8-20. *The Data tab of the Property Sheet*

6. Add another Subform control to the `OnLoan` page. Follow the exact same steps, except set the Name property to **CustomerLoansForm**. The form should go below the first Subform control.

7. Now you'll arrange these to take up the full width and height of the page. The layout control should have two columns. The cells in the left column had the labels but are now empty. Merge the two cells on the top row into one cell. Drag the lower control to the left column.

■ **Caution** When dragging the Subform control, be careful not to click inside the control and select a control inside the subform. If you do, when you drag the selected object you will be re-arranging the controls on the subform.

---

8. The lower control will have the Form View version and so make sure you have it sized so the entire form will fit correctly. The upper control can use whatever space is left. The layout should look like Figure 8-21.

www.it-ebooks.info

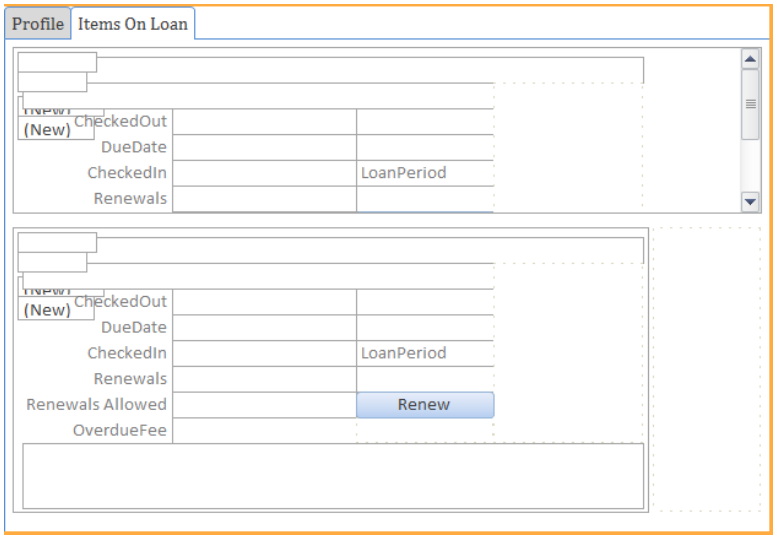


Figure 8-21. *The layout of the* `OnLoan` *page*

Connecting the Pieces

There are a couple of interesting challenges to deal with. First, you probably noticed that both copies of the `CustomerLoans` form use the Form View. That's because this is determined by the Default View property on the form. At design-time, this can only have one value. You'll add code to the `Form_Load` event to switch the view on one of the forms.

The other issue is not as obvious. With the other subforms you've used, you set the form's Filter property which was based on a control on the subform. You then linked that control with a control on the parent form. In this case, however, the filter for the two forms needs to be different. The filter on the Datasheet View will use the `CustomerID` because it will include all the `Loan` records for the customer. In the Form View, the filter will use the `LoanID`, because it will only display a single record. Again, you'll need to specify the filters at run-time.

Along the same lines, you'll need to configure the navigation controls also. These were disabled for the Form View but will be necessary for the Datasheet View.

To sort this out, switch to the Design View. In the Design tab of the ribbon, click the View Code button to display the VBA editor. This should open the code file for the `CustomerAdmin` form. Add the method shown in Listing 8-2.

Listing 8-2. *Implementation of the* `Form_Load` *Event Handler*

```
Private Sub Form_Load()
    ' Set the form properties for the Datasheet View
    CustomerLoans.Form.Filter = "[CustomerID] = CustomerID " & _
                                "And IsNull([CheckedIn]) = True"
    CustomerLoans.Form.FilterOn = True
    CustomerLoans.Form.NavigationButtons = True
    CustomerLoans.Form.NavigationCaption = "Loan"
    CustomerLoans.Form.AllowFilters = False
```


www.it-ebooks.info

```
    CustomerLoans.Form.AllowAdditions = False

    ' Switch to the Datasheet View
    ' The RunCommand function requires the form to be in focus
    Me.CustomerLoans.SetFocus
    DoCmd.RunCommand acCmdSubformDatasheetView

    ' Go back to the first page
    Me.Profile.SetFocus
End Sub
```

The form properties at design-time were configured for the Form View. As I indicated, this code configures the properties for the copy that is used for the Datasheet View. It sets the filter and configures the navigation controls. The filter limits the view to loans made by the current customer and that have not been checked in.

■ **Tip** Notice the code uses `CustomerLoans.Form` syntax. This is something that is often overlooked. The `CustomerLoans` object is a Subform object, not the actual form. You need to use its `Form` property to access properties of the form.

The second part of this code switches the form to the Datasheet View. It uses the `RunCommand` method, which executes a menu or ribbon command. It does the same thing as clicking the Datasheet View button on the ribbon. For this to work, the form has to have the focus, or, in other words, is the current form. The code first uses the `SetFocus` method to ensure that it is. After the view is changed, the focus is put back on the first page of the form.

Open the `CustomerLoans` form in the Design View. In the Design tab of the ribbon, click the View Code button. This should create a code file for this form. Enter the code shown in Listing 8-3.

Listing 8-3. *Implementation of the* `CustomerLoans` *Code File*

```
Private Sub DisplayCurrentLoan()
    ' If this is the Datasheet View synchronize the Form View
    If (CurrentView = acCurViewDatasheet And IsNull(CheckedIn)) Then

        ' Find the Form View
        Dim loanForm As Form_CustomerLoans

        ' Ignore any error if the form cannot be found
        On Error Resume Next
        Set loanForm = Parent.CustomerLoansForm.Form
        If (Not loanForm Is Nothing) Then
            loanForm.Filter = "[LoanID] = " & LoanID
            loanForm.FilterOn = True
        End If
    End If
End Sub
```


www.it-ebooks.info

```
Private Sub Form_Click()

    DisplayCurrentLoan
End Sub

Private Sub Form_Current()
    DisplayCurrentLoan

    ' Enabled the Renew button
    If (CurrentView = acCurViewFormBrowse) Then
        If (Renewals < RenewalsAllowed And DueDate > Now() - 1) Then
            Renew.Enabled = True
        Else
            Renew.Enabled = False
        End If
    End If

End Sub

Private Sub Renew_Click()
    Dim sSQL As String

    If (LoanID > 0) Then
        ' Renew the loan
        sSQL = "UPDATE Loan SET Loan.DueDate = FormatDateTime(Now()+" & _
               LoanPeriod & ",2)  WHERE LoanID=" & LoanID & ";"
        Application.CurrentDb.Execute sSQL, dbFailOnError

        ' Find the Datasheet View
        Dim loanForm As Form_CustomerLoans

        ' Ignore any error if the form cannot be found
        On Error Resume Next
        Set loanForm = Parent.CustomerLoans.Form
        If (Not loanForm Is Nothing) Then
            loanForm.Requery
        End If

    End If

    Renew.Enabled = False

    Requery
End Sub
```

The `DisplayCurrentLoan` method synchronizes the Form View with the record selected in the Datasheet View. It only does that when the CurrentView property indicates this is the Datasheet View instance. Keep in mind that both copies of the form with be executing these event handlers. It uses the parent form's reference to the other Subform control and then set's the Filter property so the Form View instance displays the currently selected loan.

www.it-ebooks.info

The `On Error Resume Next` tells the runtime to essentially ignore any errors. It is conceivable that this form could be reused somewhere else where there is no Form View version of it to synchronize. In this case, the error that will result will be ignored.

The `DisplayCurrentLoan` method is executed in both the `OnCurrent` and `OnClick` events. `OnCurrent` is raised when the form is first displayed. `OnClick` is raised when the user selects a different record in the view. In both cases, the Form View will be synchronized with the currently selected loan.

If this is the Form View, the `OnCurrent` event handler also checks to see if the `Renew` button should be enabled. It will be enabled if the item is not overdue and if the allowed number of renewals has not been exceeded.

The `Renew_Click` method is called when the user clicks the `Renew` button. It executes a SQL statement that updates the `DueDate`. The `DueDate` is determined by adding the appropriate `LoanPeriod` to the current date. It then finds the Datasheet Form and calls its `Requery` method, which refreshes the data. This code also calls the current form's `Requery` method. This ensures that both views are updated.

Testing the Page

Open the `CustomerAdmin` form in the Form View and select a customer; their information should be displayed. Now go to the Items On Loan tab and see the items they have checked out as demonstrated in Figure 8-22. (If there are no items currently on loan, you can use the `CheckOut` form that you implemented in Chapter 7 to create some loans for this customer.)

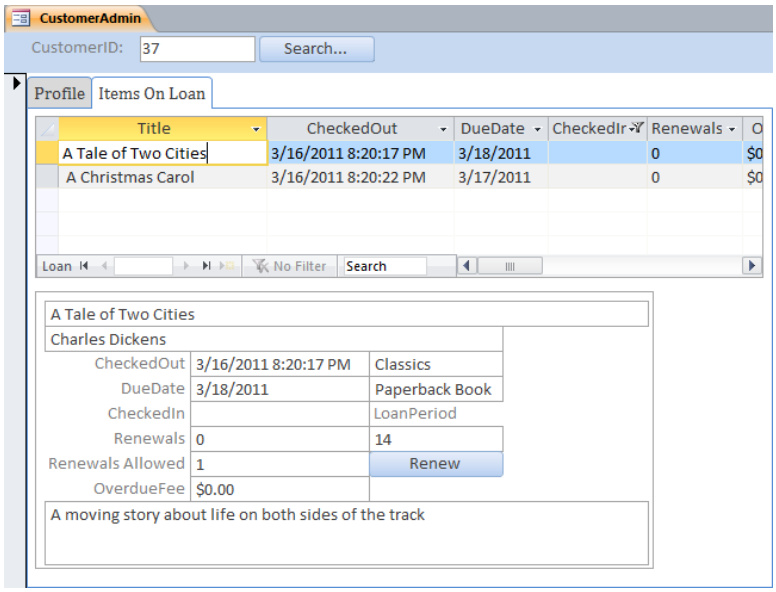


Figure 8-22. *The `OnLoan` page*

Try selecting different records and verify that the Form View is synchronized with the selected record. Try renewing an item to verify that the `Renewals` field is incremented and the `DueDate` is updated.

www.it-ebooks.info

Building the Loan History Tab

The Loan History tab will show all loans made by the customer, including those that have been returned and those still outstanding. This is really easy to implement. You just need to drop the `CustomerLoans` form on the page. In the `Form_Load` event you'll configure this as a Datasheet View and set the appropriate filter.

1. Open the `CustomerAdmin` form in the Layout View. Right-click the Items On Loan tab and click the *Insert Page* link as shown in Figure 8-23.

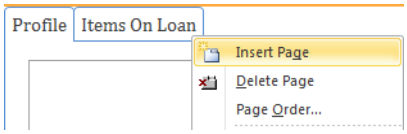


Figure 8-23. *Adding a new tab*

2. In the Other tab of the Property Sheet set the Name property to **History**. In the Format tab set the Caption property to **Loan History**.
3. In the Design tab of the ribbon, click the Subform button and then click on the `History` page. Delete the label that is created.
4. Select the Subform control and set its Name property to **CustomerLoansHistory**.
5. In the Data tab, select `CustomerLoans` for the Source Object property. Enter **txtCustomerID** in the Link Master Fields property and **CustomerID** in the Link Child Fields property. Enlarge the layout to take up the entire area of the page.
6. Go to the VBA editor and modify the `Form_Load` event handler in the `CustomerAdmin` code file. The complete event handler is shown in Listing 8-4 and the new code is shown in bold.

Listing 8-4. *Modified version of the* `Form_Load` *Event Handler*

```
Private Sub Form_Load()
    ' Set the form properties for the Datasheet View
    CustomerLoans.Form.Filter = "[CustomerID] = CustomerID " &
                                "And IsNull([CheckedIn]) = True"
    CustomerLoans.Form.FilterOn = True
    CustomerLoans.Form.NavigationButtons = True
    CustomerLoans.Form.NavigationCaption = "Loan"
    CustomerLoans.Form.AllowFilters = False
    CustomerLoans.Form.AllowAdditions = False

    ' Set the form properties for the loan history tab
    CustomerLoansHistory.Form.Filter = "[CustomerID] = CustomerID"
    CustomerLoansHistory.Form.FilterOn = True
    CustomerLoansHistory.Form.NavigationButtons = True
    CustomerLoansHistory.Form.NavigationCaption = "Loan"
    CustomerLoansHistory.Form.AllowFilters = False
```


www.it-ebooks.info

```
    CustomerLoansHistory.Form.AllowAdditions = False
    CustomerLoansHistory.Form.OrderBy = "[CheckedOut] DESC"

    ' Switch to the Datasheet View
    ' The RunCommand function requires the form to be in focus
    Me.CustomerLoans.SetFocus
    DoCmd.RunCommand acCmdSubformDatasheetView

    ' Switch the history form to a Datasheet View
    Me.CustomerLoansHistory.SetFocus
    DoCmd.RunCommand acCmdSubformDatasheetView

    ' Go back to the first page
    Me.Profile.SetFocus
End Sub
```

The new code that you've added sets the filter and navigation properties on the `CustomerLoansHistory` form just like you did earlier for the `CustomerLoans` form. It sets the OrderBy property to show the item most recently checked out first. It also uses the `RunCommand` method to change the form to use the Datasheet View.

Now try out this new feature. Open the `CustomerAdmin` form in the Form View. Select a customer and then go to the Loan History tab. The page should look similar to Figure 8-24.

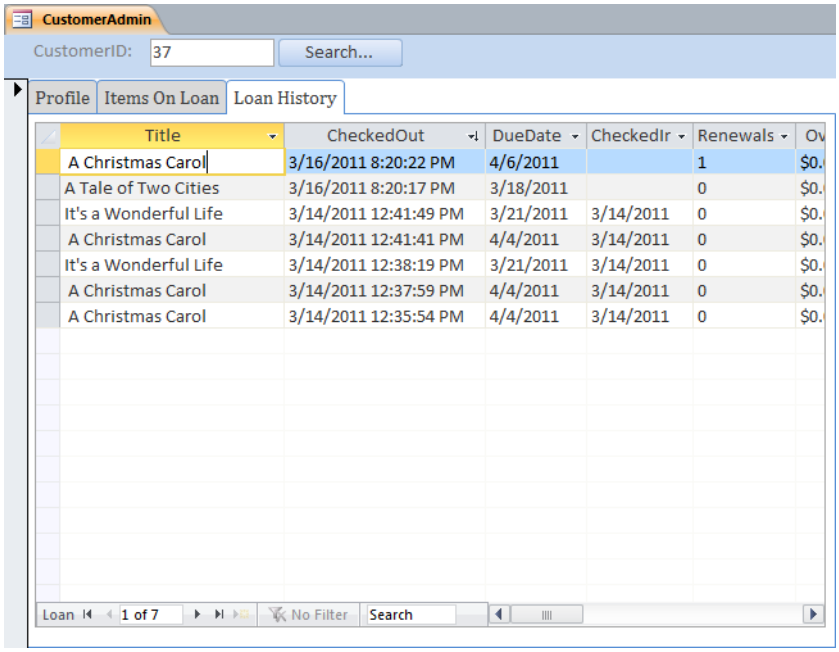


Figure 8-24. The `History` page

www.it-ebooks.info

■ **Tip** Controls on a form must have unique names. Although pages of a TabControl are designed independently from the other pages, they are still part of the main form. This means that the control names for all of the pages must be unique. For instance, if you have a Modify button on several pages, they must have different names. They can have the same caption so they appear the same but must have unique names. If you have a lot of pages with similar controls, you might want to prefix each control name with the Page name.

Summary

In this chapter you built a `CustomerAdmin` form using a TabControl to allow multiple pages. Each page has one or more subforms on it. With this pattern you can easily add new pages for additional features. For example, you could add a page to manage requested items or a page to show fines that are due or have been paid.

The Form Header provides a feature for entering or searching for the appropriate customer. By linking this with the various subforms, they automatically retrieve the appropriate records for the selected customer. After looking up the customer, all of the pages will re-query their content.

Some of the specific topics that were covered include:

- Dealing with floating controls (hidden controls not anchored with the layout control)
- Understanding the form properties that control record navigation
- Simulating a Split Form using two subforms
- Configuring form properties at run-time
- Supporting multiple instances of the same form
- Invoking the `RunCommand` method

As we progress through these chapters, you're building a collection of reusable forms that can be easily dropped on to another form or page. A good example is the Loan History page. With just setting a few properties and writing a few lines of code, an existing form was re-purposed to solve a new requirement.

You are also building your repertoire of design and implementation techniques. In the next chapter, you'll add some more. In particular you'll add data-bound images to your database and embed a web browser that will take the application to another level.

www.it-ebooks.info

CHAPTER 9

Enhancing Product Administration

In this chapter, you'll add the final piece of significant form development. So far you have completed the following:

- Provided forms for setting up categories and media types, plus a simple form for defining items (Chapter 6)
- Implemented a check-out feature for creating new loans (Chapter 7)
- Created a customer administration form (Chapter 8)

Now you'll significantly enhance the `Item` form to provide the following:

- Data-bound images
- Item search capability
- Embedded web browser

Using Data-Bound Images

The saying goes, "a picture is worth a thousand words." I think in the technologically advanced society that we live in, pictures are nearly essential. Even if the images do not provide any "required" information, the "wow" factor alone will necessitate them. Undoubtedly, you will be asked to include images in your applications.

To be clear, in this chapter I'm addressing data-bound images; support for static images uses an entirely different approach, which I will cover in Chapter 11. By data-bound images, I mean images that are part of your data. For example, in the `Library` application that you're building, you will provide the ability to store an image for each item in the database.

Image Support in Access

Access provides the following three methods for including images in your database:

www.it-ebooks.info

- Storing bitmap files in the database
- Storing OLE objects in the database
- Storing images in the file system and storing a reference in the database

I will now briefly explain each of these techniques.

Storing Bitmaps in Access

The simplest approach is to store the bitmap images directly in the Access table. Add a field with a data type of `OLE Object` and include this field on a form, just like you would a `Text` field. To store the image, right-click the data-bound control on the form and click the *Insert Object* link. In the dialog box, select the Create From File option and browse to the file containing bitmap image, as shown in Figure 9-1.

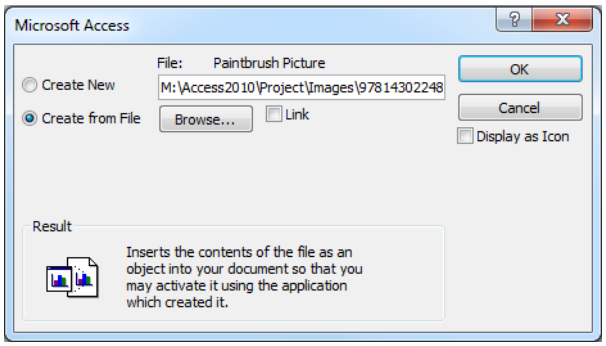


Figure 9-1*. Loading a bitmap image*

The images must be bitmaps (.bmp) or device independent bitmaps (.dib). Once loaded in the database, your forms will display the images automatically. The advantage of this approach is that it's easy and requires no coding.

The disadvantages, however, are significant. The primary one is size. Bitmaps do not use any compression and can be three or four times the size of the same image in other file types such as JPEG, TIFF or PNG. While theoretically the file size limit of Access databases is quite high, generally the performance deteriorates as the file grows. Another disadvantage is that tables with image (`OLE Object`) fields cannot be published as a web database.

Storing OLE Objects

As an alternative, you could store images that use compressed formats such as JPEG, which may take less space than bitmaps. There are numerous issues with this approach.

Object Linking and Embedding (OLE) is a technology that allows you to store data in a document that is rendered by another application. For example, you can embed an Excel spreadsheet inside a Word document. When you view or modify that spreadsheet in Word, you are actually running an Excel

www.it-ebooks.info

application. Excel is running as an OLE Server, rendering the embedded data on behalf of the Word application.

Similarly, when you store an OLE object in Access, an OLE Server will be launched to render the object. For this to work, there must be an OLE Server available to handle the type of object that was loaded. Earlier versions of Microsoft Office (earlier than Office 2003) included a Photo Editor application, which also served as an OLE Server. Image files saved as JPEG and GIF, for example, were rendered by the Photo Editor application.

With Office 2003 and later, the Photo Editor application was replaced with the Microsoft Picture Manager. This does not provide the OLE Server support that the Photo Editor did, so Access can no longer display these types of files. You can still store them, but Access does not know how to display them.

■ **Note** You can install the old Photo Editor application from the Office 2000 or Office XP installation disk. Of course there is no 64-bit version of this application, so this is not an option if you are using the 64-bit version of Office 2010.

Storing Images in the File System

In the Access table, you'll use a `Text` field that will store the file path and name of the image. This takes virtually no space in the database. This is the most practical solution for many scenarios and the only one that is compatible with web databases.

There are some disadvantages with this approach, however, including the following:

- You'll need to do a little bit of work to display the image on a form. You'll use an un-bound image control and set its path/filename using VBA code.
- Because the images are not stored in the database, you'll need to make sure they are copied along with the database when deploying the solution.

These limitations are manageable and, considering the alternatives, this will be the optimum choice in almost any environment.

Adding a Picture to the Item Table

You'll now add images to the items in your database using the recommended approach I just described. You'll add a `Text` field to the `Item` table that will store the filename of the associated item.

■ **Tip** This design will allow a single image for each item. If you needed to support two images – say, front and back cover – you would add two fields to store both filenames. If an undetermined number of images were required, you would create a child table and store the filenames there.

---

www.it-ebooks.info

1. Open the `Item` table using the Design View.
2. Add a new field named **Picture** with a Data Type of `Text`. The default Field Size of 255 is fine. You can leave all the other default properties.
3. While you're here, add another field named **URL** and select the `Hyperlink` Data Type. You will use this field later in the chapter.
4. Save the table changes. The table design should look like Figure 9-2.

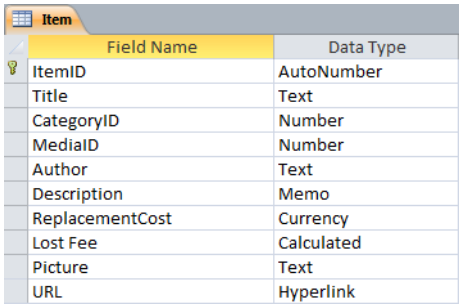


Figure 9-2. *The updated `Item` table design*

Modifying the Item Form

Now you'll add these two fields to the existing `Item` form so you can enter data for them. You will also add an Image control that will display the image on the form.

1. Open the `Item` form in the Layout View. Click the Add Existing Fields button in the Design tab of the ribbon. Drag the `Picture` field from the Field List pane to below the `ReplacementCost` control. An orange bar should appear indicating where the control will be added, as shown in Figure 9-3.

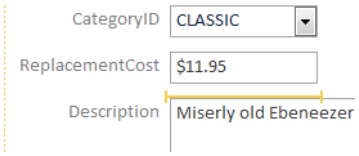


Figure 9-3. *Dragging the `Picture` field onto the `Item` form*

2. In the Other tab of the Property Sheet change the name of this control to **txtPicture**.
3. Merge the `Picture` control with the empty cell next to it.
4. In the same way, drag the `URL` field to just below the `Picture` control. After the row has been added, you may need to move the `URL` control and its label to the two leftmost cells on that row. Merge the `URL` control with the two empty cells next to it.

www.it-ebooks.info

5. At the top-right corner there are six empty cells; merge these into one cell. This is where the image will be displayed.
6. In the Design tab of the ribbon, click the Image button, as shown in Figure 9-4. Then click on the large empty cell. After adding the Image control, a dialog will appear for you to select the image. Just cancel this dialog as this will be defined programmatically.

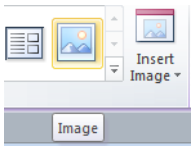


Figure 9-4. *Clicking the Image button*

■ **Caution** Just to the right of the Image button is another button labeled Insert Image. This is not part of the list of form controls. It is used to add a static image to your form, which I'll explain in Chapter 11.

---

7. In the Property Sheet, enter the name for this control as **imgPicture**.
8. Save the form changes. The updated form design should look like Figure 9-5.

www.it-ebooks.info

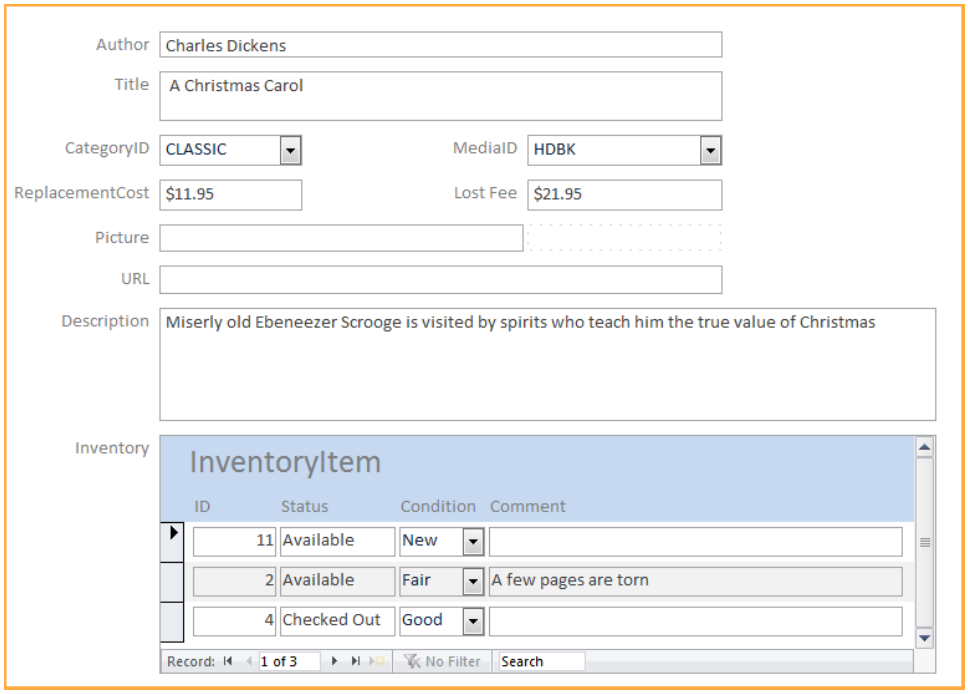


Figure 9-5. *The modified form layout*

Displaying an Image

You have added a Text control to the form for entering the filename of the associated picture. You also have an Image control that will display this file. Now you'll need to connect the two controls with a little bit of VBA code.

Adding the Code to Display an Image

Switch to the Design View. In the Design tab of the ribbon, click the View Code button, which will create a code file for this form and display the VBA editor. Open the `Main` module and add code shown in Listing 9-1.

Listing 9-1. *Implementation of the* `GetFullImagePath` *Function*

```
Function GetFullImagePath(ByVal sImage As String) As String

Dim sPath As String
Dim n As Integer

    sPath = CurrentProject.FullName
```


www.it-ebooks.info

```
    n = InStrRev(sPath, "\", Len(sPath))
    sPath = Left(sPath, n)

    GetFullImagePath = sPath + "Images\" & sImage

End Function
```

This function gets the `CurrentProject.FullName` property, which specifies the full path and filename of the Access file that is being executed. It then strips off the filename to get just the path. It then adds the hardcoded `Images` subfolder and the image file name that was passed in as a parameter. The final string is provided to the caller as the return value.

With this implementation, you can deploy the application anywhere, as long as you put the images in an `Images` subfolder. Any place that needs to retrieve the images should use this function. If you decide later to change the location of the images, you only need to modify this function.

Now add the code shown in Listing 9-2 to the code file for the `Item` form.

Listing 9-2. *Implementation of the* `Item` *Form File*

```
Private Sub Form_Current()
    DisplayImage
End Sub

Private Sub txtPicture_LostFocus()
    DisplayImage
End Sub

Private Sub DisplayImage()

    If (Len(txtPicture) > 0) Then
        On Error GoTo ErrorExit
        imgPicture.Picture = GetFullImagePath(txtPicture)
    Else
        imgPicture.Picture = ""
    End If
    Exit Sub
ErrorExit:
        imgPicture.Picture = GetFullImagePath("Static\NotFound.tif")
End Sub
```

This code implements an event handler for both the form's `OnCurrent` event and the `txtPicture` control's `LostFocus` event. The `OnCurrent` event is raised whenever a record is displayed on the form. If the filename is changed, the `LostFocus` event will also fire. Both of these event handlers call the `DisplayImage` method, which sets the Picture property of the Image control. If no picture is defined for the item, the Picture property is set to an empty string, which clears the control.

Because the images are stored outside of the database, it's a good idea to handle the situation where the specified filename is not valid. If an error occurs accessing the file, the `On Error Goto ErrorExit` statement moves the execution to the `ErrorExit` label. This displays a static image.

www.it-ebooks.info

■ **Note** Make sure you have an `Images\Static` subfolder that contains a `NotFound.tif` file, or you will generate an error trying to display the static image. You can download my `Static` folder from `www.apress.com`. I will cover that in more detail in Chapter 11.

Loading the Image Files

Now you'll need to set up a folder and store some image files. Then you will enter the filenames into your database.

1. Create an `Images` subfolder where your Access file is. If you have loaded your `Item` table with my rather eclectic list of items (or are using the downloaded database at the start of each chapter), you can download pictures of these items from `www.apress.com`. Otherwise, you should be able to get some images from the Internet.
2. Open the `Item` form using the Form View, which will display the first item.
3. Enter the filename for the associated picture. When you tab off this field, the picture should appear, as shown in Figure 9-6.

www.it-ebooks.info

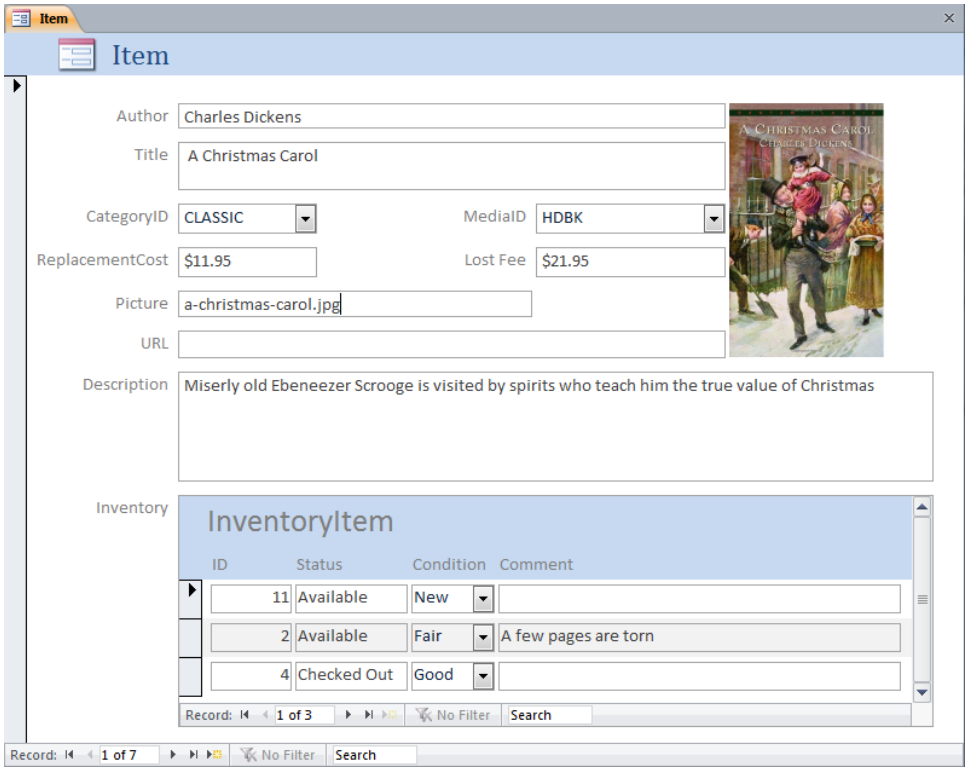


Figure 9-6. *The* `Item` *form displaying a picture*

4. Enter a filename for each of the other items so you'll have some data to test with later.

Implementing Item Search

Now you'll implement a form for searching the `Item` table. It will use a modal dialog like the `CustomerSearch` form that you implemented in Chapter 7. In the Form Header, you'll use a TabControl so you can implement multiple ways to search.

The Detail section will use the Continuous Form view so you can arrange the search results, including images. You'll simulate a record selector using conditional formatting.

www.it-ebooks.info

Importing Item Data

It's difficult to test the search feature with just a handful of records. To load a larger sample of items, I have created an Access database with the entire list of titles published by Apress. Download this `Item.accdb` file from `www.apress.com`. I'll show you how to import these records into your `Item` table.

1. You'll need a new category for the items you're importing. Open the `Category` form in the Form View and add a new category, as shown in Figure 9-7.

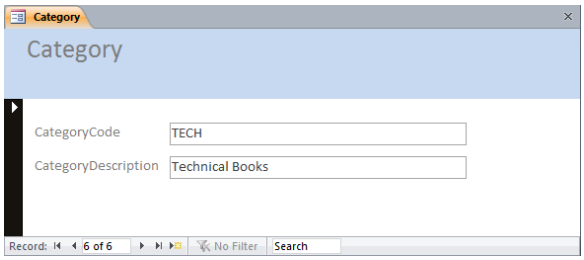


Figure 9-7. *Adding a category for technical books*

2. From the External Data tab of the ribbon, click the Access button, which will display the dialog box shown in Figure 9-8.

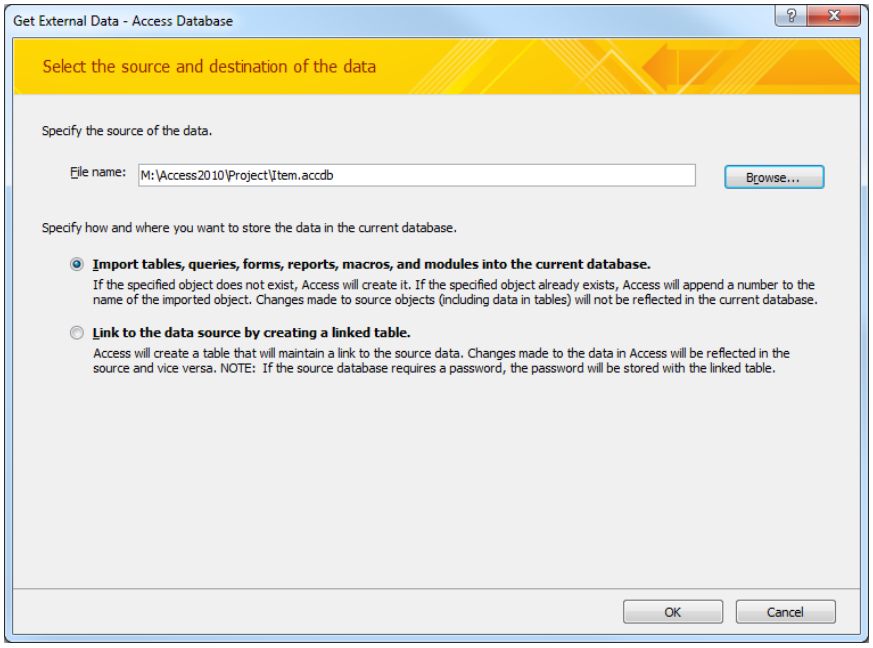


Figure 9-8. *Selecting the external Access database options*

www.it-ebooks.info

3. Browse to the `Item.accdb` file that you downloaded and select the first option, which is to import the data rather than link to it. The next dialog box, shown in Figure 9-9, allows you to select the objects that you want to import. Select the `Item` table.

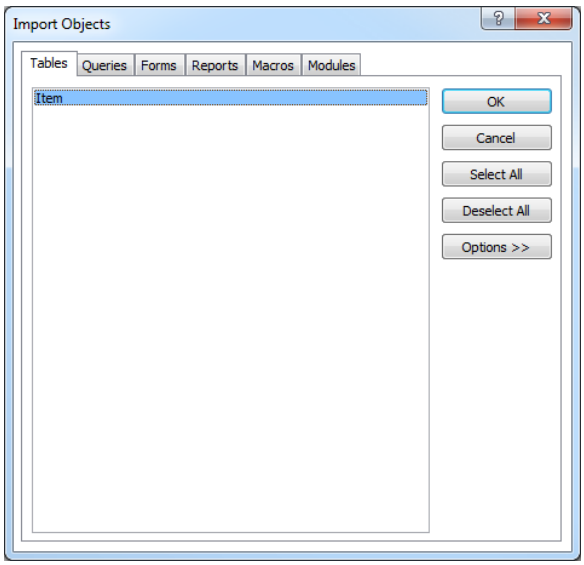


Figure 9-9. *Selecting to import the* `Item` *table*

4. Click the OK button, which will start the import. When the confirmation dialog appears, just close it. This will have created a new table named `Item1`, because an `Item` table already existed. Now you'll create a query to copy the records from `Item1` to `Item`.

5. From the Create tab of the ribbon, click the Query Design button. Select the `Item1` table and close the Show Table dialog box. Click the Append button in the ribbon to change this to an append query. This will display the Append dialog box to select the table to be appended to. Select the `Item` table, as shown in Figure 9-10.

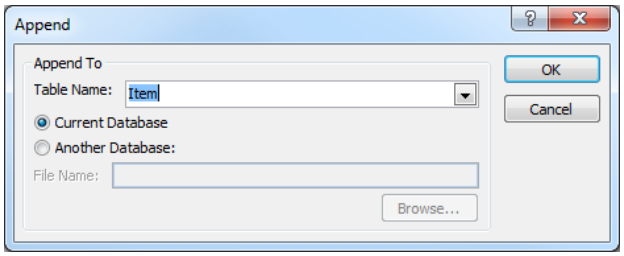


Figure 9-10. *Appending to the* `Item` *table*

www.it-ebooks.info

6. Double-click each of the columns in the `Item1` table, except for `ItemID` and `LostFee`. You won't specify a value for `ItemID` because this is an `AutoNumber` field and the database will assign unique IDs for you. Also, the `LostFee` column is calculated based on the `ReplacementCost`, so you'll let the data macro handle that for you. The completed query design should look like Figure 9-11.

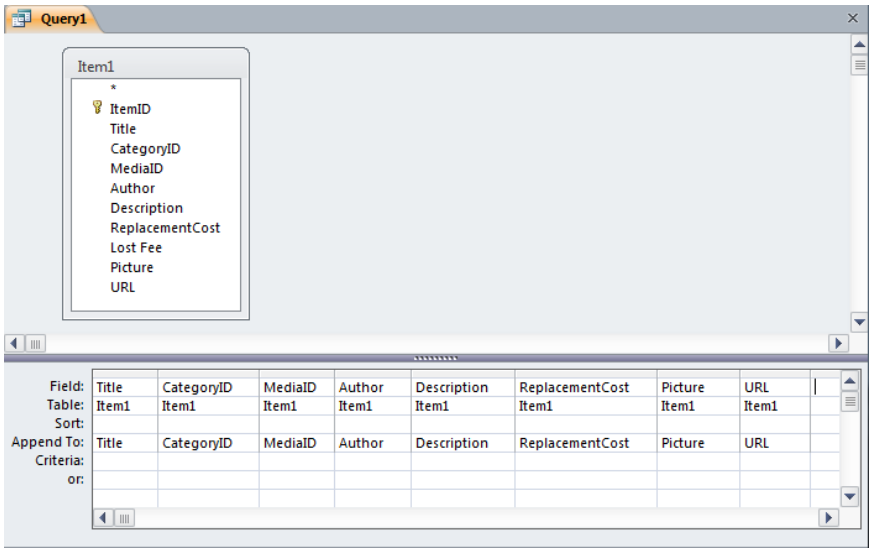


Figure 9-11. The completed append query

■ **Tip** The records in the `Item1` table have the `CategoryID` set to 6. It's possible that your Technical Books category has a different ID. If so, instead of using the value from the `Item1` table, in the Field row, replace `CategoryID` with an expression that has the appropriate value for your database.

7. Run the query and you should see a confirmation dialog, shown in Figure 9-12, telling you that 1,344 records were added to the `Item` table.

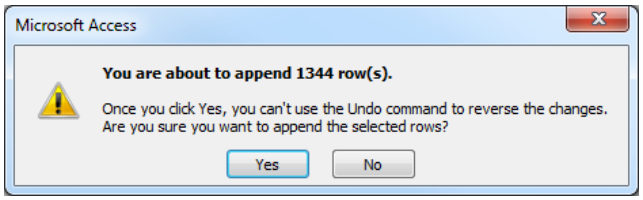


Figure 9-12. Confirmation of the records added

www.it-ebooks.info

8. Close the query without saving it. Delete the `Item1` table that was imported. Open the `Item` form in the Form View and page down through some of the new records. You should see the Image Not Found graphic, because the referenced file is not in your `Images` folder. You can also download the image files from `www.apress.com`.

Designing the Search Form

You'll start by designing the layout of the `ItemSearch` form. You'll use the Modal Dialog template and then place controls on the Form Header and Detail section.

1. From the Create tab of the ribbon, click the More Forms button and then click the *Modal Dialog* link. Right-click the form and click the *Form Header/Footer* link.
2. Cut and paste the OK and Cancel button from the Detail section to the Form Footer. Shrink the Detail section to about 1-inch high and expand the Form Header to be about 1½-inch high.
3. Click the Tab Control button in the Design tab of the ribbon then draw a rectangle that takes up the entire Form Header.
4. Change the Name and Caption properties on the first page to **Basic**. The Name and Caption properties of the second page should be **Advanced**.
5. Add a TextBox control at the top of the Form Header. Delete the associated label that is generated. Change the Name property to **txtSelectedItemID** and set the Visible property to No. This control will be used to keep track of which item in the Detail section has been selected.

The Form Header should look like Figure 9-13.

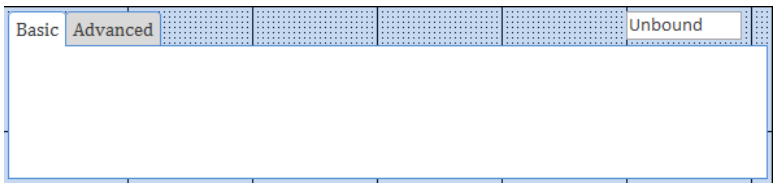


Figure 9-13. *The initial Form Header layout*

Designing the Basic Search Page

The basic search feature will accept a single keyword or phrase. You'll provide a ComboBox control to select which fields should be searched in such as `Author`, `Title`, or `Description`. You'll provide another ComboBox control to select which media types should be included.

1. Add a TextBox control to the `Basic` page (make sure the page is selected before you add the TextBox). Set the Name property to **txtKeywords** and set the Caption of the associated label to **Keywords.**

www.it-ebooks.info

2. Select the `txtKeywords` control and its associated label. From the Arrange tab of the ribbon, click the Stacked button. This will create a layout control with two columns. Click the Insert Right button three times to create a total of five columns. Click the Insert Below button to add a second row.
3. Select the `txtKeywords` control and the two columns to the right and merge them into one cell.
4. Make sure the control wizards are turned on by clicking the dropdown icon to see all the form controls. The Use Control Wizards icon should be highlighted.
5. Click the Combo Box button and then click the `Basic` page. In the first Combo Box Wizard dialog box, select the second option as shown in Figure 9-14.

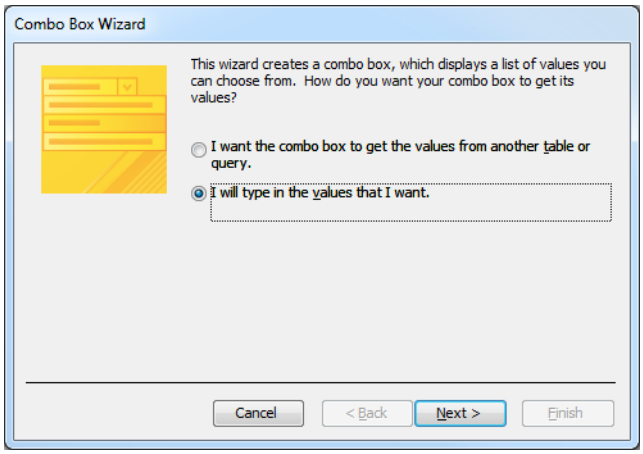


Figure 9-14. *Selecting the option to specify the combo options*

6. In the next dialog box, enter the following values, as shown in Figure 9-15.
   - **<Any field>**
   - **Author**
   - **Title**
   - **Description**

www.it-ebooks.info

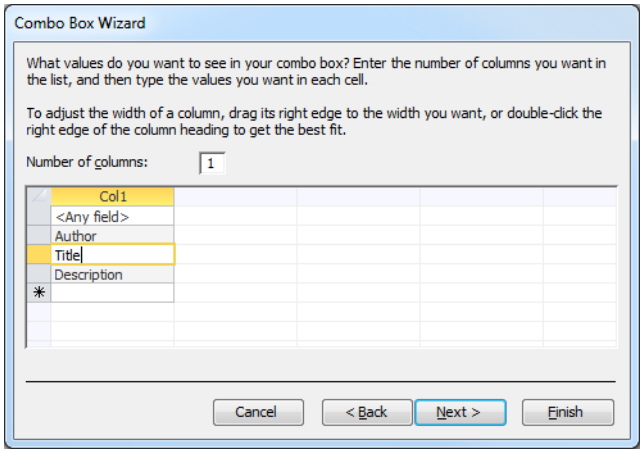


Figure 9-15. Specifying the allowed values

7. In the final dialog box enter **Look In** as the label text. Set the Name property of this control to **cbField**. In the Data tab, set the Default Value property to "**<Any field>**." Set the Limit To List property to Yes. Set the Allow Value Lists Edits to No. Drag the control to the second cell of the bottom row.

8. Add another ComboBox control to the `Basic` page. This time, in the Combo Box Wizard, select the first option, which is to get the values from a table or query.

9. In the second dialog box, select the `Media` table, as shown in Figure 9-16.

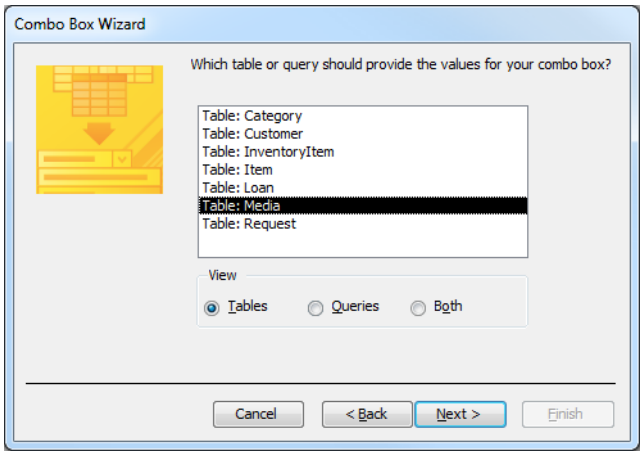


Figure 9-16. Selecting the `Media` table for the combo box source

www.it-ebooks.info

10. In the third dialog box, shown in Figure 9-17, select the `MediaID` and `MediaDescription` columns to be included.

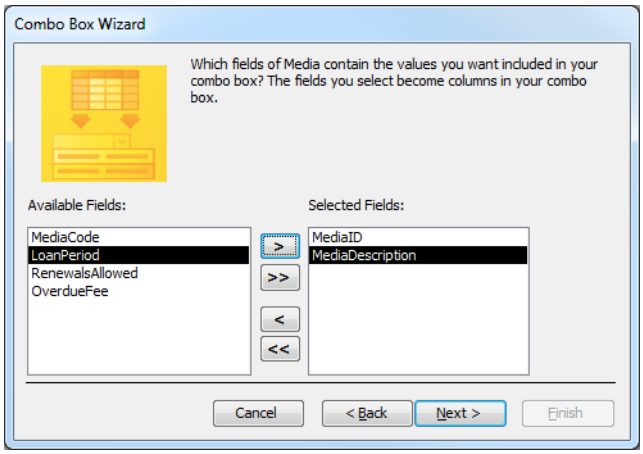


Figure 9-17. Selecting the columns to be included

11. In the fourth dialog box, shown in Figure 9-18, select the `MediaDescription` field for the sort option.

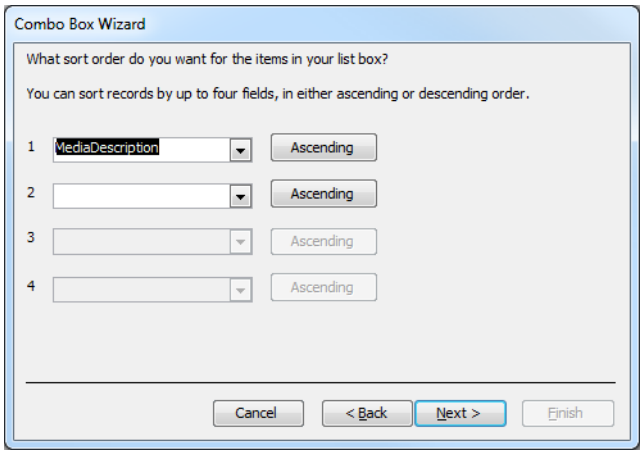


Figure 9-18. Sorting by the `MediaDescription` column

12. The fifth dialog box, shown in Figure 9-19, shows a preview of what the Combo Box will look like. Leave all the default settings.

www.it-ebooks.info

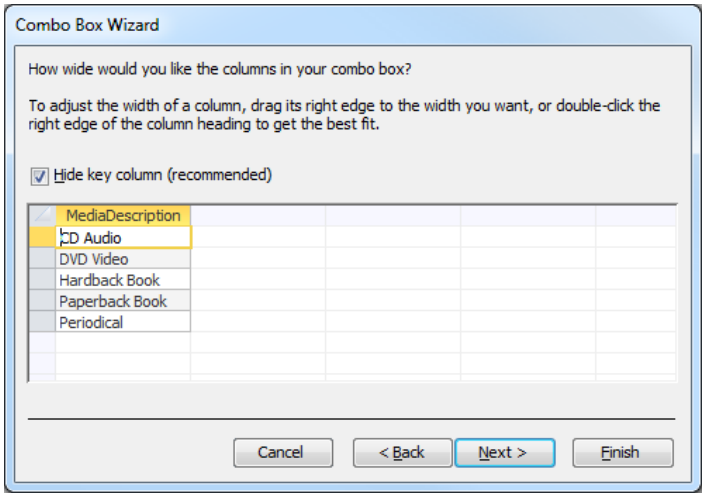


Figure 9-19. *The Combo Box preview*

13. In the final dialog box enter **Include** as the label text. Set the control's Name property to **cbMedia**. In the Data tab, set the Default Value property to **0**. Set the Allow Value Lists Edits to No. Drag the control to the fourth cell of the bottom row.

This `cbMedia` Combo Box will allow you user to select one of the media types. You also want to provide an option to include all of the media types in the search, which will require another row in the dropdown list. To do that you'll modify the query that populates this control.

1. Select the `cbMedia` control and in the Data tab of the Property Sheet, select the Row Source property. Click the ellipses, which will launch the Query Builder. Select the SQL View. Replace the existing SQL with the following code:

```
SELECT [Media].[MediaID], [Media].[MediaDescription] FROM [Media]
UNION SELECT 0, "<All media>" FROM [Media]
ORDER BY [MediaDescription];
```

■ **Tip** This SQL uses a `UNION` clause to add an additional hard-coded value to the values supplied by the `Media` table. The value 0, with a description of <All media>, will indicate that the search should look in all media types.

---

2. Close the Query Builder and click the Yes button when prompted to update the property.
3. Select all the labels and set the Text Align property to Right.

www.it-ebooks.info

4. Add a command button to each of the cells in the far-right column. Just cancel the control wizards when they launch. For the top button, enter **BasicSearch** for the Name property and **Search** for the Caption. For the lower button, enter the **BasicClear** for the Name property and **Clear** for the Caption.

Save the form and enter the name **ItemSearch** when prompted. The layout of the Form Header should look like Figure 9-20.

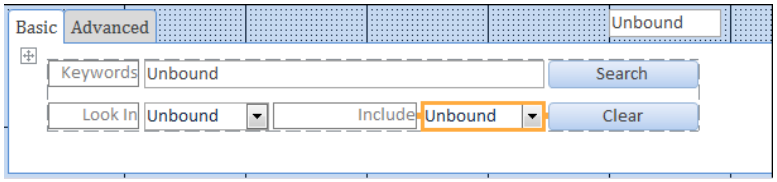


Figure 9-20. *The layout of the Form Header*

■ **Note** To add other search methods, such as the Advanced tab, you would add unbound controls for specifying the search criteria. You then implement a Search button that formats a filter based on those criteria. I will leave the implementation details for you to work out on your own.

Designing the Detail Section

Now you'll design the results section of the `ItemSearch` form. It will use the Continuous Form View, so all of the controls that you place here will be repeated for each record returned by the search. Just like with the `CustomerSearch` form you created in Chapter 7, the detail section will contain data-bound controls. You will limit the records that are displayed with a filter that is generated based on the specified search criteria.

1. In the Property Sheet, select the Form object and the Data tab. For the Record Source property select the `Item` table.
2. Set the following form properties:
   - Filter On Load: Yes
   - Allow Additions: No
   - Allow Deletions: No
   - Allow Filters: No
   - Allow Edits: Yes (this is needed to be able to enter the search criteria)
   - Default View: Continuous Forms
   - Record Selectors: No

www.it-ebooks.info

- Navigation Buttons: Yes
- Close Button: No
- `Filter: [ItemID] = CLng('0')` (This will prevent items from displaying until the search criteria is entered.)

3. In the Design tab of the ribbon, click the Add Existing Fields button. Double-click the `ItemID` and `Picture` fields to add them to the Detail section. Delete their associated labels. Prefix the Name property of both of these controls with "txt." For example, the Name property of the `Picture` control should be **txtPicture**. Set the Visible property to both of these controls to No. Drag both of these controls to the top-left corner of the Detail section.
4. Click the Add Existing Fields button in the ribbon. Double-click the following fields to add them to the form and then delete the associated labels:
   - `Author`
   - `Title`
   - `Description`
5. In the Other tab of the Property Sheet, change the Name property for all three controls to prefix them with "txt." For example, **txtAuthor**.
6. Select all three of these controls and set their Locked property to Yes to prevent the user from modifying these fields.
7. With these controls still selected, from the Arrange tab of the ribbon, click the Stacked button to create a layout control. Click the Insert Left button to add a column to the left of the existing column. Merge all three cells of the left column into one cell.
8. Select the entire layout, click the Control Padding button, and then click the *None* link. This will remove the spaces between the controls.
9. Drag the layout to the top-left corner of the Detail section.
10. Add an Image control to the left cell of the layout. Set the Name property of this control to **imgPicture**. In the Data tab, for the Control Source property enter **=GetFullImagePath([txtPicture])**. This binds the Picture property of the Image control to the `txtPicture` control that specifies the filename. The layout should look like Figure 9-21.

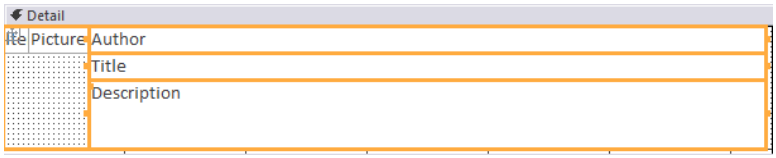


Figure 9-21. *The layout of the Detail section*

www.it-ebooks.info

11. Right-click the Cancel button and click the Build Event link. This should display the Macro Designer. The existing macro calls the Close Window action. Change the Save parameter from Prompt to No. Save the macro changes and close the Macro Designer.

12. Select the OK button. In the Event tab of the Property Sheet, change the On Click property from Embedded Macro to Event Procedure. Change the Name property to Close.

Adding the VBA Code

In the Design tab of the ribbon, click the View Code button, which will generate a code file for this form and display the VBA Editor. Enter the code shown in Listing 9-3.

Listing 9-3. *Initial Implementation of the* `ItemSearch` *Form*

```
Private Sub Close_Click()
   If (Me.CurrentRecord = 0) Then
      MsgBox "Please select an item first", vbExclamation, "No Item Selected"
   Else
      Me.Visible = False
   End If
End Sub

Private Sub Form_Current()
    txtSelectedItemID = Me.ItemID
End Sub

Private Sub BasicSearch_Click()
    Dim s As String
    Dim sFilter As String

    If (IsNull(Me.txtKeywords) Or Len(Me.txtKeywords) <= 1) Then
        MsgBox "Please enter a keyword", vbExclamation, "No Keyword Specified"
    Else
        s = "*" + Me.txtKeywords + "*"

   sFilter = "(([Author] Like '" + s + "' And ([Forms]![ItemSearch]![cbField] " & _
             "= '<Any field>' Or [Forms]![ItemSearch]![cbField] = 'Author'))" & _
      "Or ([Title] Like '" + s + "' And ([Forms]![ItemSearch]![cbField] " & _
      "= '<Any field>' Or [Forms]![ItemSearch]![cbField] = 'Title'))" & _
      "Or ([Description] Like '" + s + "' And ([Forms]![ItemSearch]![cbField] " & _
      "= '<Any field>' Or [Forms]![ItemSearch]![cbField] = 'Description')))" & _
      "And ([MediaID] = [Forms]![ItemSearch]![cbMedia] Or " & _
      "[Forms]![ItemSearch]![cbMedia] = CLng('0'))"

        DoCmd.ApplyFilter "", sFilter, ""
    End If

End Sub
```


www.it-ebooks.info

```
Private Sub BasicClear_Click()
    txtKeywords.Value = Null
    cbField = "<Any field>"
    cbMedia = cbMedia.DefaultValue
End Sub
```

The Cancel button simply closes the form. However, the event handler for the OK button (implemented in the `Close_Click` method) merely hides the form. This allows the calling form to retrieve the selected item. This is the same way that you implemented the `CustomerSearch` form.

In a Continuous Form, each record is displayed as a mini form which is repeated as many times as necessary. When you click on a record, that specific "form" becomes the current one and the `OnCurrent` event is raised. The `Form_Current` event handler takes advantage of this and captures the `ItemID` of the selected record.

The `BasicSearch_Click` method is called when the `Search` button is clicked. It builds a filter string based on the input criteria and then calls the `ApplyFilter` method. Unlike the `CustomerSearch` form, it is assumed here that the only partial values are entered so the keyword is automatically prefixed and suffixed with the "*" wildcard character. The `BasicClear_Click` method clears the `txtKeywords` control and restores the default value for the ComboBox controls.

Testing the Search Function

Now you're ready to try it out. Save the code file and the form changes, then switch to the Form View. Enter a keyword or phrase and click the `Search` button. The form should look like Figure 9-22.

www.it-ebooks.info

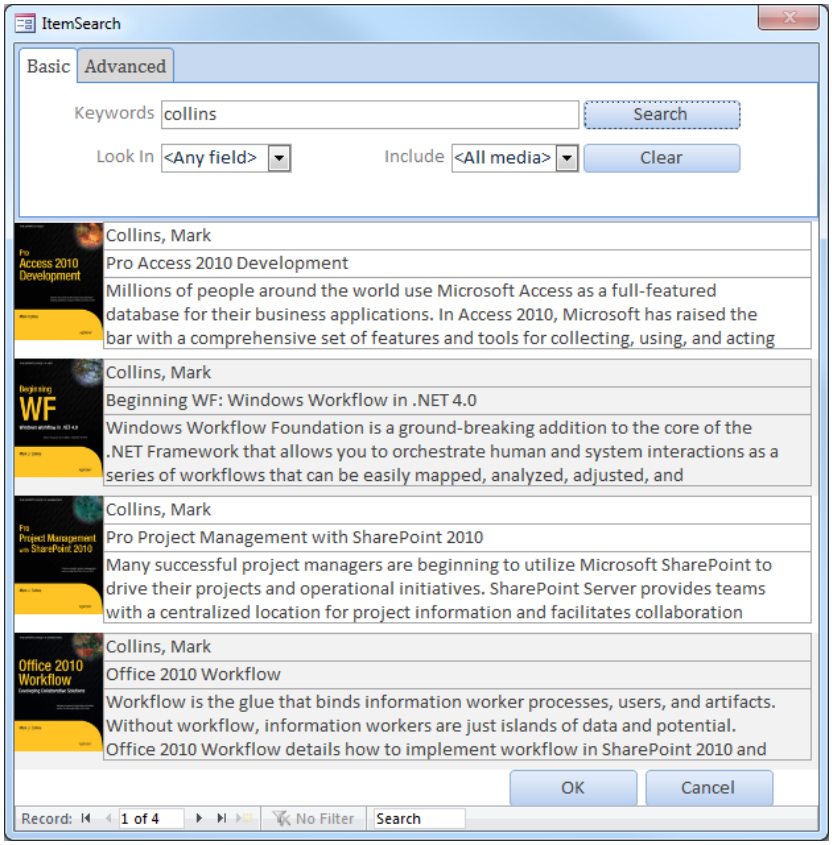


Figure 9-22. *The* `ItemSearch` *dialog*

Using Conditional Formatting

You probably noticed that you can't tell which item is selected. You could turn on the Record Selectors property, which will display a black arrow next to the selected item. However, I'll show you another way to highlight the selected record, using the Conditional formatting feature.

1. Open the `ItemSearch` form in the Design View. In the Detail section, right-click the `txtAuthor` control and click the *Conditional Formatting* link as shown in Figure 9-23.

www.it-ebooks.info

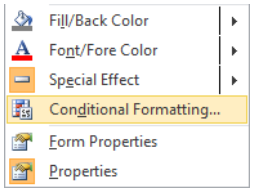


Figure 9-23. *Selecting the Conditional Formatting link*

2. This will display the Condition Formatting Rules Manager, shown in Figure 9-24. The `txtAuthor` control is already selected, and there are no rules currently applied to this control. Click the New Rule button, which will display the New Formatting Rule dialog box.

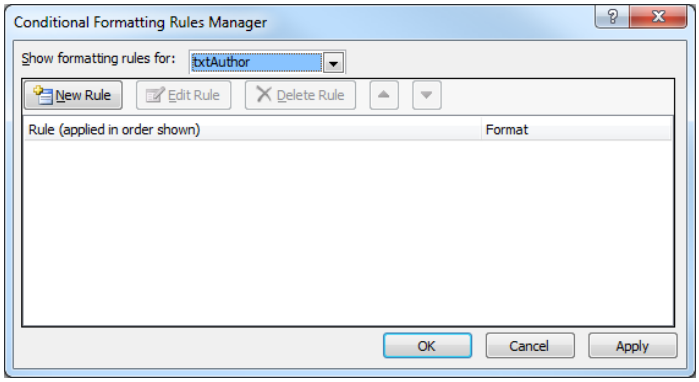


Figure 9-24. *The Conditional Formatting Rules Manager*

3. Select the first option since you'll be using an expression and change the combo box to Expression Is. For the expression enter the following code. The `txtSelectedItemID` control is in the Form Header and the `OnCurrent` event handler updates this to store the ID of the item that is currently selected. The expression uses this to return `True` if the record being displayed has the same ID as the selected record:

```
[txtItemID] = [txtSelectedItemID]
```

4. Finally, you need to specify in the rule what to do when the expression is true. Click the Background Color dropdown and select a background. This will be used when displaying the selected record. The completed rule should look like Figure 9-25.

www.it-ebooks.info

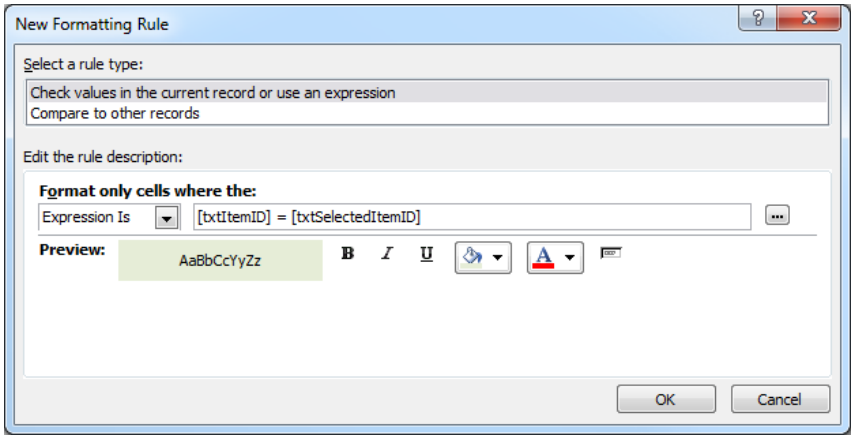


Figure 9-25. *The New Formatting Rule dialog box*

5. Click the OK button to close the dialog box. The Conditional Formatting Rules Manager will now show the rule that you just added.
6. Click the Apply button to save this rule.
7. Select the `txtTitle` control to see the rules defined for it, which should be none. Click the New Rule button and create the exact same rule that you did for the `txtAuthor` field.
8. In the same way, create a rule for the `txtDescription` control. All three controls should have the same rule.
9. Click OK to close the rules manager and then save the form changes.

Open the `ItemSearch` form using the Form View and search for some items. The selected item should be displayed in a different background, as shown in Figure 9-26.

www.it-ebooks.info

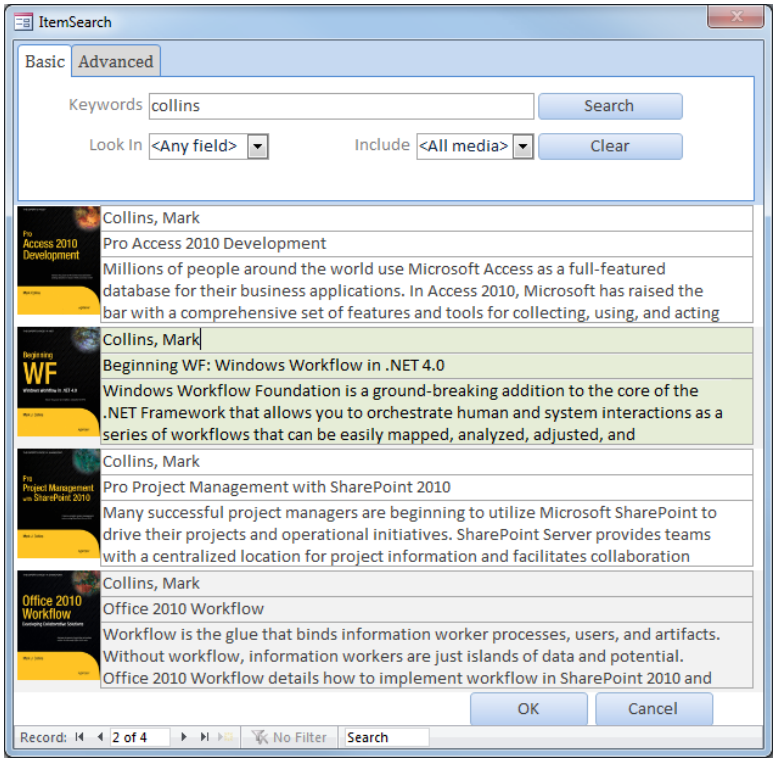


Figure 9-26. *The* `ItemSearch` *form with conditional formatting*

Invoking the ItemSearch Form

Now you'll modify the `Item` form to use the `ItemSearch` form to select an item. This is a fairly simple matter of adding a `Search` button to the Form Header that loads the `ItemSearch` Form. This will work just like the customer search feature you implemented in Chapter 7.

1. In the Property Sheet, select the Form object and the Data tab. For the Filter property, enter **[ItemID] = CLng('0')**. Also set the Filter On Loan property to Yes. This will keep the form from loading the entire list of items. Instead, it will only load the item selected from the `ItemSearch` form.

2. Open the `Item` form using the Layout View. The layout control for the Form Header has two controls; an Image control that displays a logo and a Label that displays the form title. Select the Label control and, from the Arrange tab in the ribbon, click the Split Horizontally button. This will split the cell into two cells, leaving an empty cell on the right.

3. Create a command button and put it in the empty cell. Cancel the control wizard. Enter the Caption property as **Search...** and enter **Search** for the Name property. Resize the cells so the Form Header looks like Figure 9-27.

www.it-ebooks.info



Figure 9-27. *The modified Form Header with a Search button*

4. Right-click the Search button and click the *Build Event* link. In the Choose Builder dialog box select Code Builder. This will create an event handler for the OnClick event and open the VBA Editor. Enter the code shown in Listing 9-4.

Listing 9-4. *Implementation of the Search_Click Method*

```
Private Sub Search_Click()
    Dim sForm As String
    Dim ID As Integer

    sForm = "ItemSearch"

    ' Open the search form
    DoCmd.OpenForm sForm, acNormal, , , , acDialog

    ' If the form is not loaded, the user clicked the Cancel button
    If (IsLoaded(sForm)) Then
        ID = Forms(sForm)!txtSelectedItemID
        DoCmd.Close acForm, sForm

        DoCmd.ApplyFilter "", "[ItemID] = CLng(" & ID & ")", ""
    End If
End Sub

Private Sub Form_Load()
    DoCmd.ApplyFilter "", "[ItemID] = CLng(0)", ""
End Sub
```

This code first opens the ItemSearch form. Because it is a modal dialog, the call does not return until the form is closed or hidden. It then checks to see if the form is still loaded. If it is, it uses the txtSelectedItemID control to get the ID of the selected item. It then applies a filter that returns only the selected item.

This code also provides the implementation for the Form_Load method. Because the search feature is manipulating the Filter property, Access tries to remember the last filter that was applied. This code ensures that the initial filter is used when the form is started.

Open the Item form using the Form View. You should notice that no record is displayed. Click the *Search* button, enter a search, select one of the items, and click the OK button. The ItemSearch form should disappear and the selected item displayed in the Item form, as demonstrated in Figure 9-28.

www.it-ebooks.info

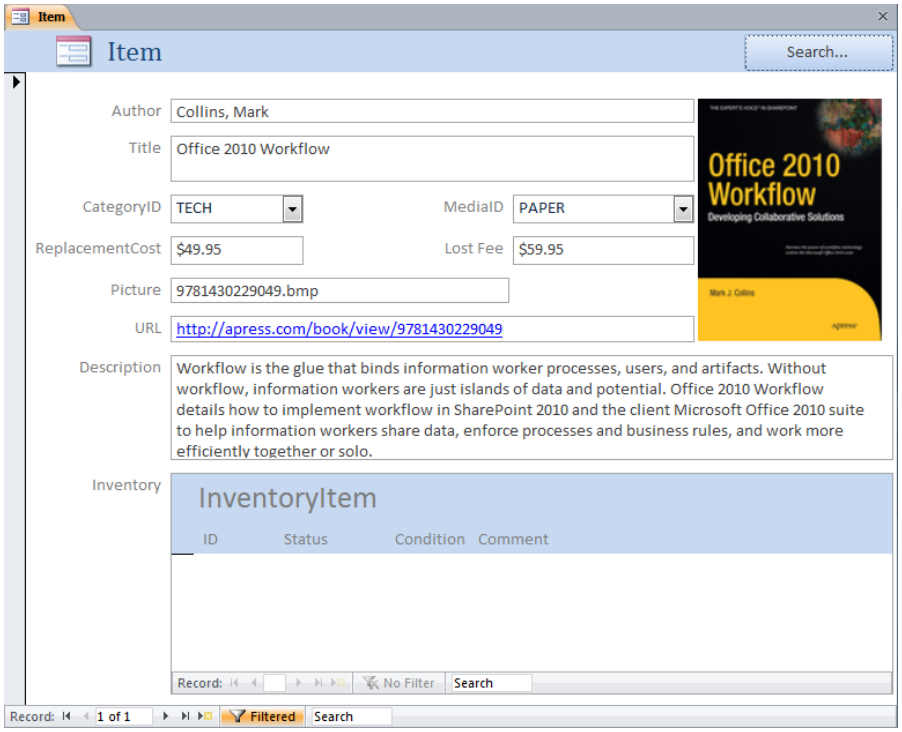


Figure 9-28. *The* `Item` *form showing the selected item*

Enhancing the Item Form

There are a few more enhancements that I want to show you. You may have noticed that there are no inventory items for the item selected in Figure 9-28. The item has been set up in the database and the author, title, and other details have been defined. But you don't actually have a copy of it in your inventory to lend out. You'll add a facility to this form to receive a copy of the item and add it to the inventory.

There is also a URL set up for this item where additional information can be obtained. You'll add an embedded web browser to this form and bind it to this URL. This will cause the browser to automatically display that web page.

Finally, before we finish the chapter, I'll show you how to add the item image to the Item On Loan tab of the `CustomerAdmin` form. You implemented this in the previous chapter and left a place on the form to display an image. You'll add that to finish this page.

Adding an Inventory Item

Adding an inventory item is actually really easy to do. In Chapter 4 you created a query called `AddInventoryItem`, which inserts a record into the `InventoryItem` table. It requires an `ItemID` parameter so it knows which item to add. You'll now add a command button to the `Item` form to run that query. For a little variety, you'll implement the logic behind the button with a macro.

www.it-ebooks.info

1. Open the `Item` form using the Layout View. Add a command button to empty cell just to the right of the `txtPicture` control. Cancel the wizard when it starts. Set the Name properties to **AddInventory** and the Caption property to **Add Inventory**.
2. Right-click this control and click the *Build Event* link. In the Choose Builder dialog, select Macro Builder, which will start the Macro Designer. This macro will perform three actions:
   - Suppress the normal Access warnings about records being updated
   - Run the query
   - Re-query the subform that lists the inventory items so it will refresh its contents
3. The "Add New Action" dropdown list shows the actions that can be added to the macro. Some actions are restricted if the document is not trusted. The `SetWarnings` action is one of these. To be able to select this action, you must first click the Show All Actions button in the ribbon, as shown in Figure 9-29.

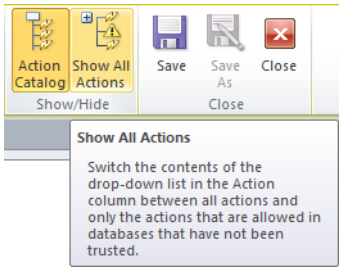


Figure 9-29. *The "Show All Actions" button*

4. In the "Add New Action" dropdown list, select the `SetWarnings` action. Make sure the Warnings On parameter is set to No.
5. For the next action, select the `OpenQuery` action. For the Query Name parameter, select the `AddInventoryItem` query. This will display the `itemID` parameter. Enter [**ItemID**] for its value.
6. For the last action, select `Requery` and enter `InventoryItem` for the Control Name parameter.

The final macro design should look like Figure 9-30.

www.it-ebooks.info

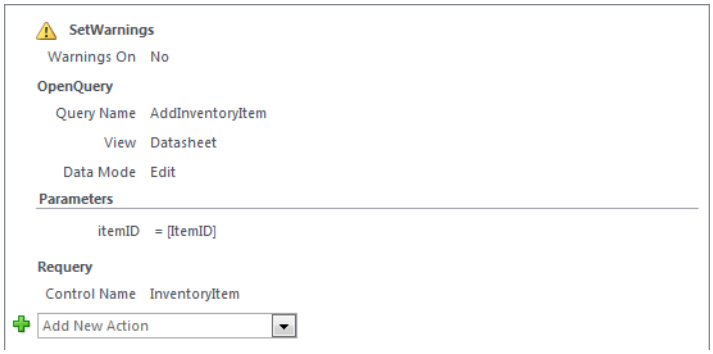


Figure 9-30. *The* `AddInventory` *macro design*

Save the macro and close the Macro Designer. To test this, open the `Item` form using the Form View, select an item and then click the Add Inventory button. A new record should be added to the subform, as demonstrated in Figure 9-31.

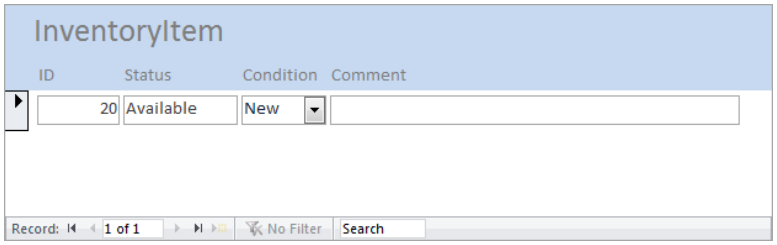


Figure 9-31. *A new inventory item added to the subform*

Adding a Web Browser

Now you'll add a WebBrowser control to the `Item` form. You'll use the `URL` control to automatically navigate to the associated web page.

1. Open the `Item` form in the Layout View. You'll need to first create a cell to drop the browser in. Select the `InventoryItem` subform, because it's on the last row. In the Arrange tab of the ribbon, click the Insert Below button add a new row.

■ **Caution** If the Insert Below button is not enabled, it's probably because you clicked inside the subform, selecting one of its cells. Try clicking on the border of the subform. The Property Sheet will indicate if you have selected the `InventoryItem` subform.

---

www.it-ebooks.info

2. The bottom row will have two cells, merge these together into one. Expand the height of the cell to be about 5 inches.

3. In the Design tab of the ribbon, click the Web Browser Control button, as shown in Figure 9-32.

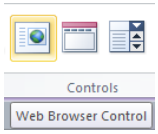


Figure 9-32. *Clicking the Web Browser Control button*

4. Then select the large empty cell that you just created. Cancel the control wizard when it starts. In the Property Sheet set the name of this control to **webBrowser**.

5. Now you'll add a couple of event handlers. Select the `webBrowser` control. In the Event tab of the Property Sheet, for the On Navigate Error property, select Event Procedure and click on the ellipses. Enter the following code for the implementation. This simply navigates the web browser to a home page should the specified URL be invalid.

```
webBrowser.Object.Navigate http://apress.com
```

6. While in the VBA Editor, add the following code to the `Form_Current` method:

```
If (Len(URL) > 0) Then
    webBrowser.Object.Navigate URL
    webBrowser.Visible = True
Else
    webBrowser.Visible = False
End If
```

This causes the browser to navigate to the address specified by the `URL` field. If a `URL` is not specified, the browser is hidden.

There is no place to type a URL so the user can't navigate somewhere else. However, links on the page could take them to other sites. If you want to control where they are navigating to, implement an event handler for the `OnBeforeNavigate` event. To do so, in the Property Sheet, select the On Before Navigate property, select Event Procedure and then click the ellipses. Add the following code for this event's implementation.

```
If (Left(CStr(URL), 17) <> "http://apress.com") Then
    Cancel = True
End If
```

This code will prevent navigating to any URL that doesn't start with "apress.com."

www.it-ebooks.info

■ **Note** You might want to comment out this code and not leave it in your application. It is here for demonstration purposes only.

Open the `Item` form in the Form View and search to one of the Apress titles. The embedded web browser should display the associated page on the Access site, as demonstrated in Figure 9-33.



Figure 9-33. *The `Item` form with the web browser enabled*

Using Page Breaks

Now that you've added the web browser, the `Item` form has become too large to display all of it at once. Of course you can scroll down to see the rest of the form. As an alternative, you can insert page breaks in the form and then navigate between pages by using buttons.

1. Open the `Item` form using the Design View. From the Design tab of the ribbon, click on the Page Break button, as shown in Figure 9-34.

www.it-ebooks.info

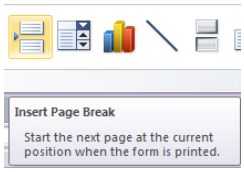


Figure 9-34. *Clicking the Page Break button*

2. Then click on the `Item` form, just above the `webBrowser` control. You should see a series of dots, shown in Figure 9-35, which indicates a page break.

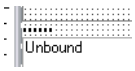


Figure 9-35.*The page break indicator*

3. Now you'll add buttons to the Form Footer that the user can use to navigate to each page. Click the Button button in the ribbon and then click on the Form Footer. Cancel the control wizard. Set the Name property to **Page1** and the Caption property to **Details**. Right-click the control and click the *Build Event* link. In the Choose Builder dialog box, select the Macro Builder.

4. In the Macro Designer, select the `GoToPage` action and enter the Page Number parameter as **1**. The macro should look like Figure 9-36. Save and close the Macro Designer.



Figure 9-36. *Implementing the `Page1` button*

5. In the same way, add another button named `Page2` and set the Caption property to **Browser**. Implement the button with a macro, setting the Page Number parameter to **2**.

6. Now you'll need to arrange the buttons in a layout control. The Form Footer should look like Figure 9-37.



Figure 9-37. *The layout of the Form Footer*

Save the form changes and switch to the Form View. You will have buttons at the bottom of the form that you can use to easily switch between pages as shown in Figure 9-38.

www.it-ebooks.info

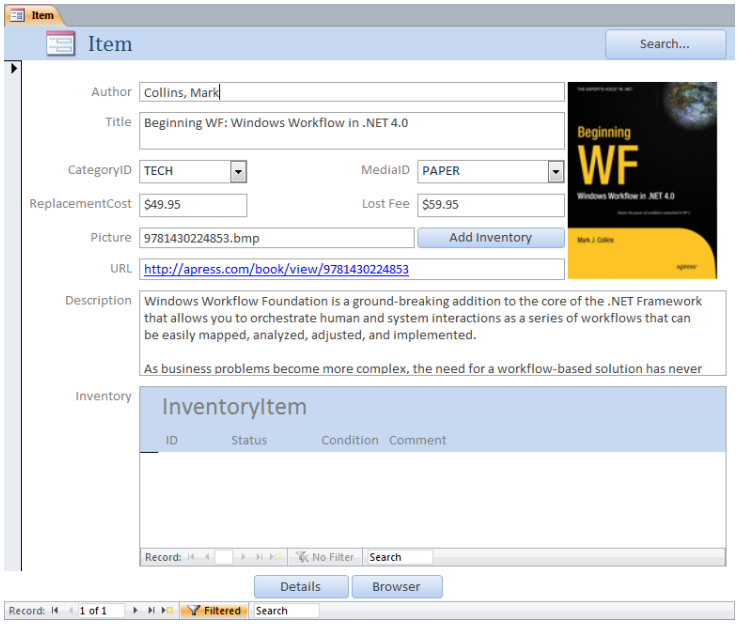


Figure 9-38. *The* `Item` *form with page buttons*

Modifying the CustomerLoan Form

The last thing you'll need to do is modify the `CustomerLoan` form and add an image control.

1. Open the `LoanDetail` query in the Design View. Double-click the `Picture` field in the `Item` table to add this column to the query. Save the query and then close it.
2. Open the `CustomerLoans` form in the Layout View. Click the Add Existing Fields button in the ribbon.
3. Drag the `Picture` field to the empty cell underneath the `Renew` button. This will insert a new column for the associated label. Right-click the label and click the *Delete Column* link.
4. Change the Name property of the Picture control to **txtPicture**. Set the Visible property to No.
5. In the Design tab of the ribbon, click the Image button and then click in the Detail section. Cancel the control wizard. Drag this control to the empty cell on the right side of the form. Set the Name property to **imgPicture**.
6. In the Data tab of the Property Sheet, set the Control Source property to **=GetFullImagePath(txtPicture)**.
7. Save the form.

www.it-ebooks.info

To test the change, open the `CustomerAdmin` form using the Form View, select a customer and select the Item On Loan tab. The form should look like Figure 9-39.

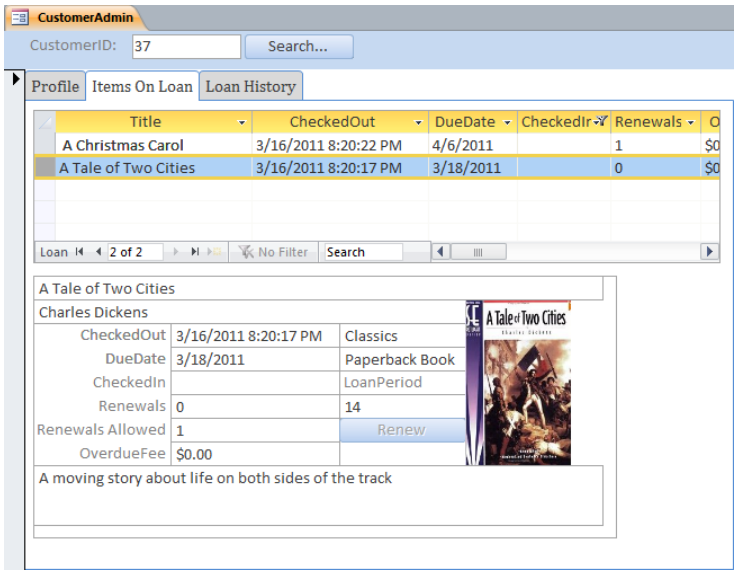


Figure 9-39. *The modified* `CustomerAdmin` *form*

Summary

In this chapter you made some significant enhancements to the `Item` form, including:

- Adding images to the item data
- Providing a search facility to look for items in the database
- Embedding a web browser for additional item information

Some of the other Access techniques that you learned include:

- Importing tables from another Access database
- Using conditional formatting
- Calling an action query from a command button
- Using page breaks

In the next chapter, you will make changes to this application to prepare it for the end users. This will include creating a navigation form and locking down the design and development features.

www.it-ebooks.info

CHAPTER 10

Enhancing the User Experience

At this point, you have developed a well-functioned application and you're ready to turn it over to the end users… well, not quite yet. When you open the Access file, there are lots of neat things like tables and queries, macros, and VBA – all the sorts of items that we developers like to work with. Most end users are not going to want to see this. More important, if you have to support it, you don't want them seeing it either, much less have the ability to change any of it.

In this chapter, I'll show you some ways to package the application with the end user in mind. The first step is to provide a way for them to navigate to the forms that they will be using. I'll demonstrate two ways to accomplish this using a custom navigation form and a custom ribbon. With that in place, you'll then need to remove the standard navigation and lock down the application to prevent unwanted alterations.

Form Navigation

When the application is first loaded, you will need some sort of "welcome" form that will present to the user their choices of things they can do such as check out a customer, look for an item, and so on. Access 2010 provides a really nice facility for creating a navigation form by simply dragging the forms and reports to the appropriate navigation structure. However, the complex forms you have developed so far will not work with this technique. Essentially, the navigation form becomes the main form and all your other forms are added as subforms. As I explained in Chapter 8, there are limitations with subforms. For example, you cannot use a Split Form as a subform.

■ **Note** I will demonstrate the built-in Navigation Form template in Chapter 15. You will use this to organize the web forms.

So you'll need to create your own custom navigation form. Fortunately, this is simple to do – almost surprisingly so. The hardest part is usually deciding what options should be allowed and how to best organize them. Although there is no *right* way to organize these, I suggest the following structure:

- Operations (tasks that you do all day long)
  - Checkout a customer
  - Set up a new customer or view/modify an existing one
  - Lookup an item

www.it-ebooks.info

- Administration (updating configurable objects)
  - Items
  - Categories
  - Media types
- Maintenance (routine tasks for ongoing support and maintenance)
  - Calculate late fees
  - Cancel old requests

Creating the Menu Form

You'll create a blank form and place command buttons on it. You can take advantage of the control wizards that will set up a macro to open the associated form when the user clicks a command button. You will use a TabControl to organize the buttons into the three top level items (Operations, Administration, and Maintenance).

1. Click the Blank Form button in the Create tab of the ribbon.
2. In the Design tab of the ribbon, click the Tab Control button, and then draw a rectangle on the form. This will create a TabControl with two pages. Right-click this control and click the *Insert Page* link. Select each page and set the Name property of each to **Operations**, **Administration**, and **Maintenance**. The form should look like Figure 10-1.

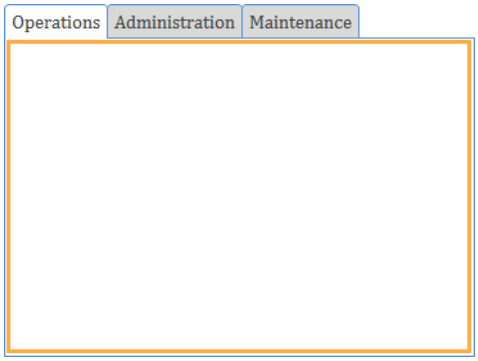


Figure 10-1*. The initial* `Menu` *form layout*

3. In the Property Sheet, select the Form object and set the Record Selectors and Navigation Buttons properties to No. These are not applicable for this form, because it does not access a table.

www.it-ebooks.info

■ **Tip** Make sure that the control wizards are turned on. To check, click the dropdown icon near the bottom-right corner of the Controls section of the ribbon. This will show all of the available controls. The icon next to the *Use Control Wizards* icon should be highlighted as shown in Figure 10-2. If not, click the *Use Control Wizards* link to turn them on.

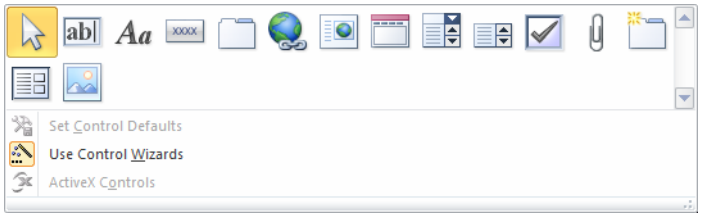


Figure 10-2. *Checking the status of the control wizards*

4. Click the Button button and then click the `Operations` page. This will launch the Command Button Wizard. In the first dialog box, select the `OpenForm` action as shown in Figure 10-3.

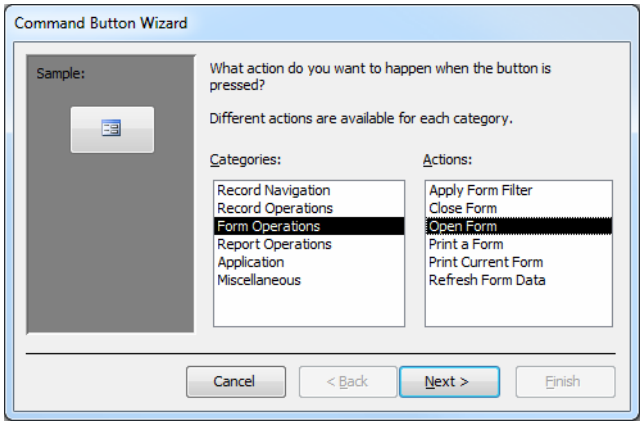


Figure 10-3. *Selecting the* `OpenForm` *action*

5. In the second dialog box, select the `CheckOut` form, as shown in Figure 10-4.

www.it-ebooks.info

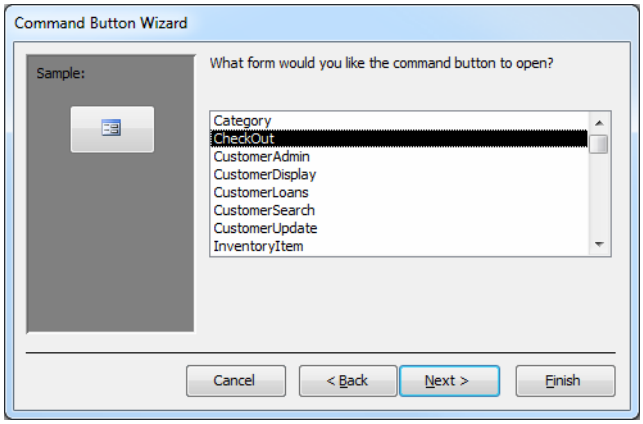


Figure 10-4. *Selecting the* `CheckOut` *form*

6. The third dialog box, shown in Figure 10-5, provides an option to filter the record when opening the form. This doesn't apply in this scenario, so select the second option, which is to show all records.

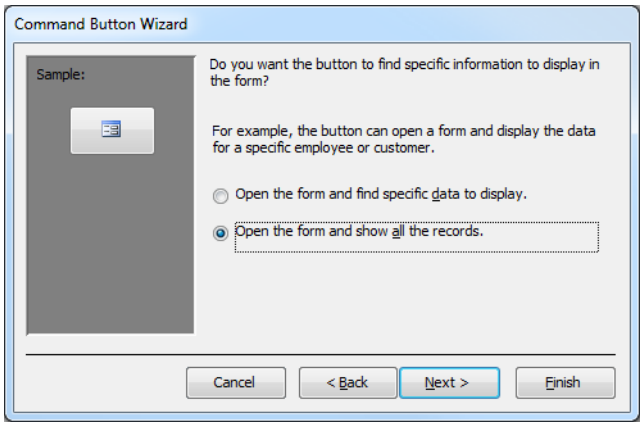


Figure 10-5. *Selecting the option to show all records*

7. The fourth dialog box is used to specify the text or picture that should be displayed on the button. Select the Text option and enter **Check Out**, as shown in Figure 10-6.

www.it-ebooks.info

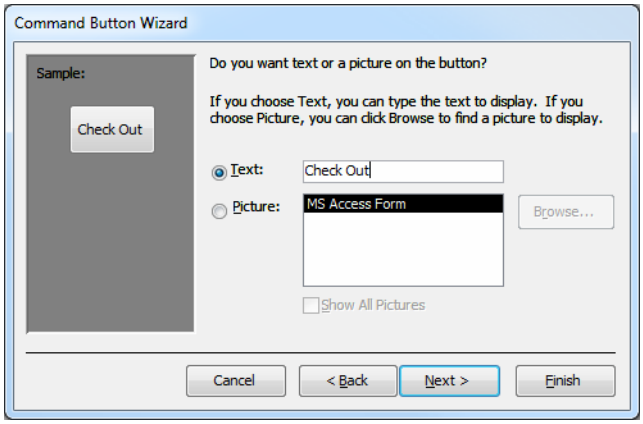


Figure 10-6. *Specifying the button caption*

8. In the final dialog box, enter the name **CheckOut**, as shown in Figure 10-7.

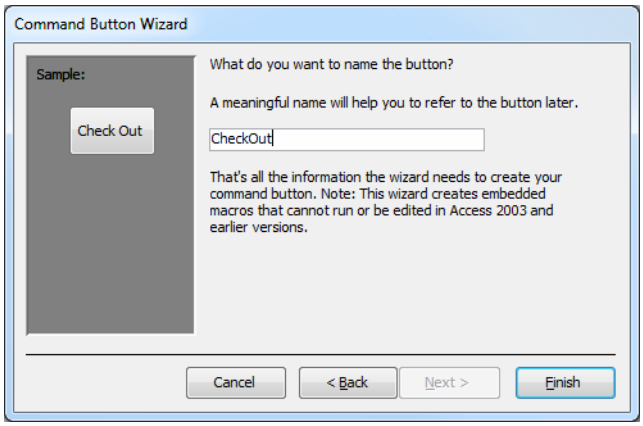


Figure 10-7. *Specifying the name of the* `CheckOut` *button*

9. You should now have a button on the `Operations` page that will open the `CheckOut` form. Repeat this process to add buttons that will open the `CustomerAdmin` and `Item` forms. The page design should look like Figure 10-8.

www.it-ebooks.info

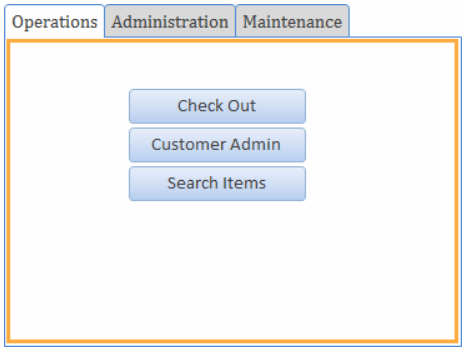


Figure 10-8. The `Operations` page layout

10. Using the same approach, add buttons to the `Administration` page to open the following forms:

    - `Item`
    - `Category`
    - `Media`

On the `Maintenance` page, you'll create buttons that will call a macro. There are two macros included in the Navigation pane: `CalculateLateFees` and `CancelOldRequests`. You created these in Chapters 3 and 4 to demonstrate how to call a data macro and an action query. Now you'll provide buttons in the `Menu` form to allow the user to call these.

11. Add a command button to the `Maintenance` page. In the Command Button Wizard, select the `RunMacro` action, which you'll find in the Miscellaneous category, as shown in Figure 10-9.

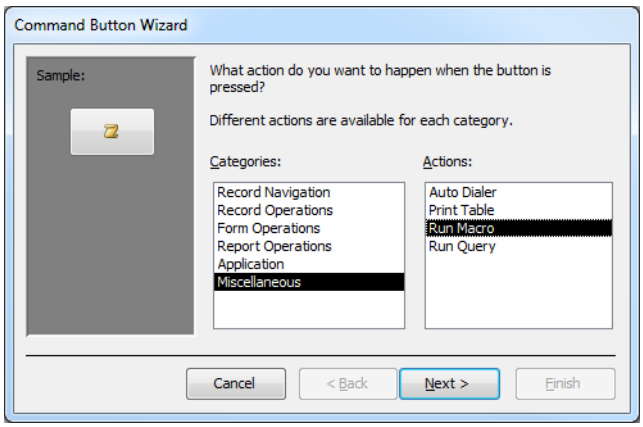


Figure 10-9. Selecting the `RunMacro` action

www.it-ebooks.info

12. In the next dialog box, select the `CalculateLateFees` macro, as shown in Figure 10-10.

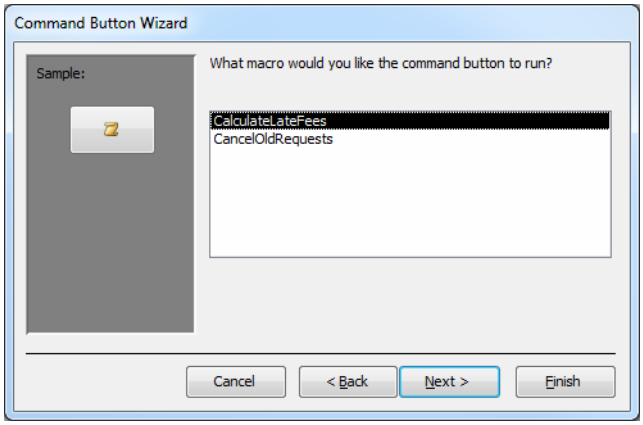


Figure 10-10. *Selecting the* `CalculateLateFees` *maco*

13. Fill out the rest of the dialog boxes as you did for the other command buttons.
14. Repeat this process to create another button that will call the `CancelOldRequests` macro.
15. The `Maintenance` page should look like Figure 10-11.

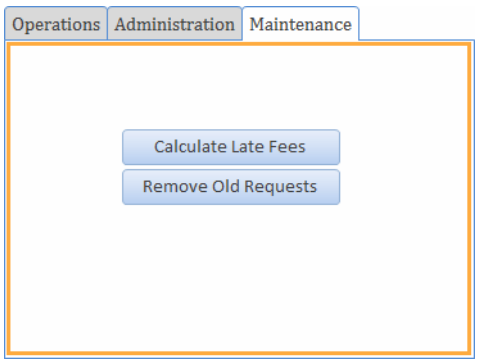


Figure 10-11. *The completed* `Maintenance` *page*

Save the form and enter the name **Menu** when prompted. Switch to the Form view. Try clicking the buttons and verify that the appropriate forms are loaded and that they work as expected.

www.it-ebooks.info

Auto-Loading the Menu Form

The `Menu` form is a handy way for the user to start any of the forms designed for them to access. Now you'll configure Access to load this form automatically when the application is started.

1. Go to the File tab of the ribbon, which displays the Backstage View.
2. Click the Options button to display the Access Options dialog box.
3. Select the Current Database tab and select the `Menu` form in the Display Form combo box as shown in Figure 10-12. This tells Access to load the `Menu` form whenever this file is opened.

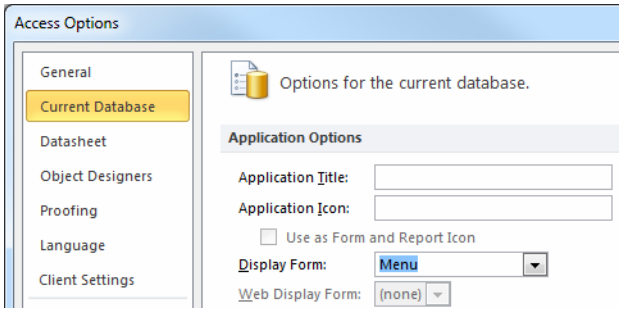


Figure 10-12. *Selecting the* `Menu` *form to be auto-loaded*

When you save the changes, you'll get the pop-up dialog shown in Figure 10-13, which lets you know that this change will take affect the next time the file is loaded.

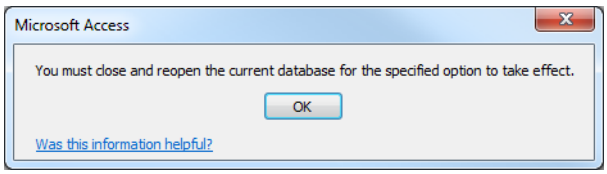


Figure 10-13. *Reminder to close and reopen the database*

Close the Access application and re-open the `Library.accdb` file. You should see the `Menu` form start automatically.

Ribbon Navigation

Another approach to providing navigation to your forms is to customize the ribbon. You can add a tab to the existing ribbon or create your own custom ribbon, with an XML file and a little bit of VBA code. I'll first demonstrate how to add a new tab to the ribbon, and then I will show you how to build your own data-driven menu using a custom ribbon.

www.it-ebooks.info

Implementing a Sample Ribbon Tab

A sample XML file that creates a new ribbon tab is shown in Listing 10-1.

Listing 10-1. *Sample Script to Create a Custom Ribbon Tab*

```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="demo1" label="Demo Tab">
        <group id="group1" label="Sample">
          <button id="button1" size="large" label="Sample1"
                  screentip="Sample 1" supertip="This is a sample button"
                  getImage="GetImage" onAction="OnMenuAction" />
          <button id="button2" size="normal" label="Sample2"
                  screentip="Sample 2" supertip="This is a sample button"
                  getImage="GetImage" onAction="OnMenuAction" />
          <button id="button3" size="normal" label="Sample3"
                  screentip="Sample 3" supertip="This is a sample button"
                  getImage="GetImage" onAction="OnMenuAction" />
       </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

The `ribbon` element has a single attribute called `startFromScratch`. If this is set to `false`, as it is here, the defined tabs are added to the existing ribbon. Use this option if you want to add a custom tab but leave the standard tabs in place. If you set this to `true`, all the existing tabs will be removed and the ribbon will only contain whatever is defined in this XML script.

The `ribbon` element contains a `tabs` collection, which has a single `tab` labeled "Demo Tab." Each `tab` then defines one or more `group` elements. This script has a single group labeled "Sample," which contains three `button` elements.

Each button element specifies the following attributes:

- `id`: A unique identifier for this button
- `size`: The size of the graphic for this button (either `large` or `normal`)
- `label`: The text that is displayed with the button
- `screentip`: Heading for the hover text
- `supertip`: The hover text that is displayed when the mouse is over this control
- `getImage`: The name of a callback function that supplies the image for this button
- `onAction`: The name of a callback function that is called when the button is clicked
- `tag`: Not shown here, but can be used for providing details to the callback functions

www.it-ebooks.info

To add this custom tab to your ribbon, you'll need to implement the two callback functions, `GetImage` and `OnMenuAction`. Then you will also need to install this ribbon in the database.

At the bottom of the navigation pane, you should see the `Main` module. Double-click this to display the VBA editor. Add the code shown in Listing 10-2 to the `Main` code file.

***Listing 10-2**. The Initial Implementation of the Ribbon Callback Functions*

```
Public Sub GetImage(ByVal control As Office.IRibbonControl, ByRef image)

    image = "HappyFace"

End Sub

Public Sub OnMenuAction(ByVal control As Office.IRibbonControl)

    MsgBox "You've clicked the button " & control.ID & " on the Ribbon"

End Sub
```

The `GetImage` callback returns the `HappyFace` icon, which I'll explain later. The `OnMenuAction` callback simply displays the ID of the control that was clicked.

■ **Note** You will need to add a reference to the Office Object Library. In the VBA Editor, click the Tools menu and then the *References* link. In the References dialog box, add the Microsoft Office 14.0 Object Library, as shown in Figure 10-14.

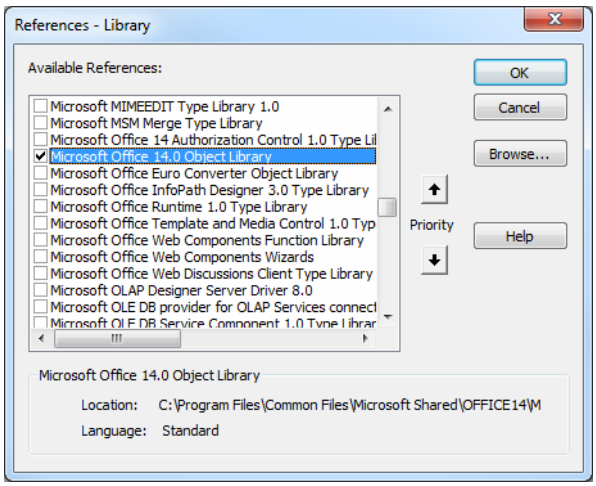


***Figure 10-14**. Adding the object library reference*

www.it-ebooks.info

Now all that's left is to install the XML file. I'll show you one way to do that now, and later in this chapter, I'll show you a second way. In the first approach, you'll create a table named `USysRibbons` and copy the XML into this table. When a database is opened, Access looks for the `USysRibbons` table and will automatically install any ribbons that are defined there.

1. From the Create tab of the ribbon, click the Table Design button.
2. Add the following fields:
   - **RibbonID**: `AutoNumber` (set this as the primary key)
   - **RibbonName**: `Text`
   - **RibbonXML**: `Memo`
3. Save the table and enter the name **USysRibbons** when prompted.
4. Open the table in the Datasheet View. Enter **Demo** in the `RibbonName` field. Enter the XML shown in Listing 10-1 in the `RibbonXML` field and save the record.

The table should look like Figure 10-15.

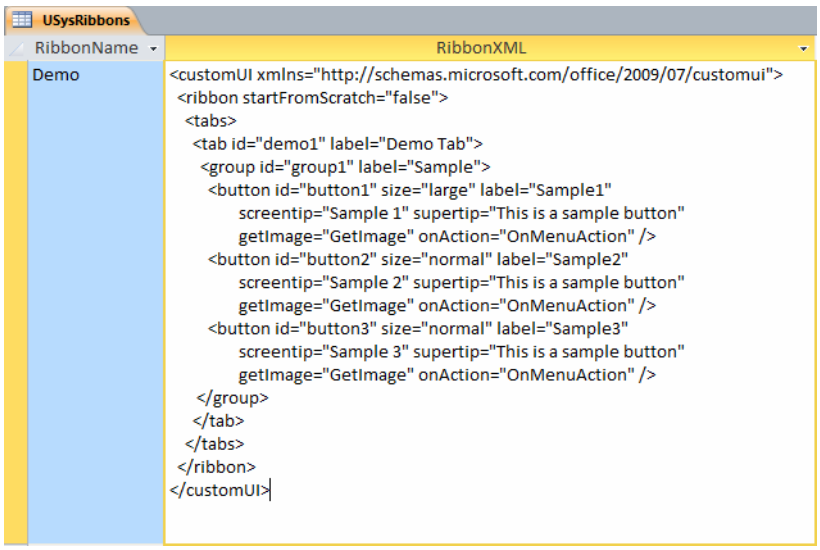


Figure 10-15. *The contents of the* `USysRibbons` *table*

Close the Access database and then re-open it. Access should have loaded your custom ribbon, but now you need to tell Access to use it.

5. Click the File tab to display the Backstage View.
6. Click the Options button and then select the Current Database tab.
7. About halfway down, in the "Ribbon and Toolbar Options" section, select the `Demo` ribbon, as shown in Figure 10-16.

www.it-ebooks.info

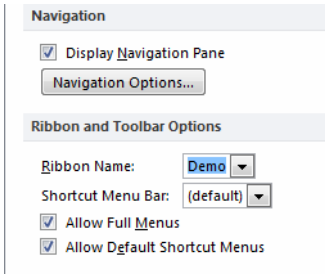


Figure 10-16. *Selecting the* `Demo` *ribbon*

Close the dialog and you will be reminded that you'll need to close the database and re-open it. Do that now and when the database is reloaded you should see a Demo tab like the one shown in Figure 10-17.

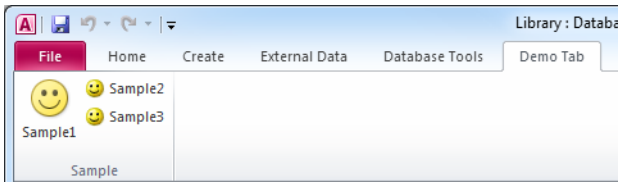


Figure 10-17. *The custom* `Demo` *ribbon tab*

Try clicking the buttons in the custom tab. You should see a pop-up window that tells you the name of the button that was clicked.

■ **Tip** As you probably know, XML is not very forgiving. The slightest syntax error will prevent the file from loading. If Access encounters an error in processing the XML it will simply ignore your custom ribbon. You won't see any errors, but you won't have a custom tab. This can be frustrating when trying to find where the problem is. While developing and testing custom ribbons, you can enable these error messages. Click the Options button on the Backstage View. In the Client Settings tab, scroll down to the General section. Select the "Show add-in user interface errors" check box, as shown in Figure 10-18.

---

www.it-ebooks.info

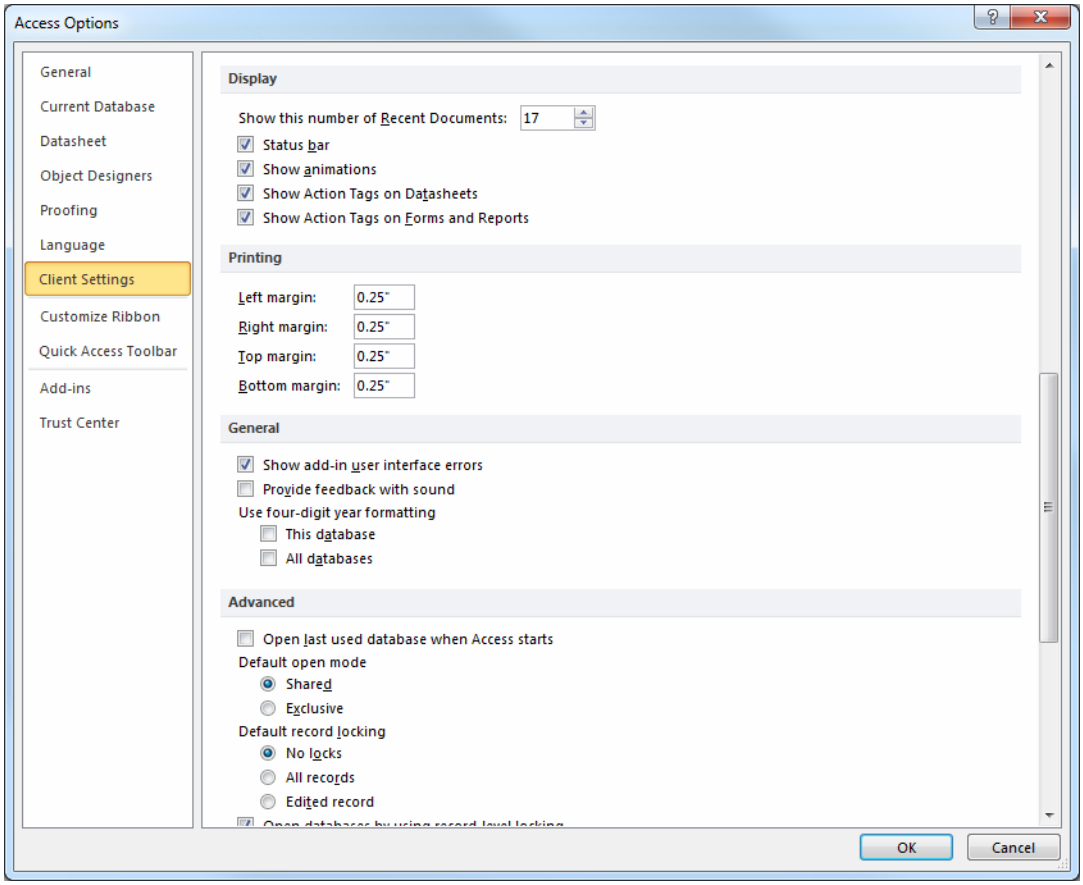


Figure 10-18. *Enabling UI error messages*

Displaying the System Objects

You may have noticed that, when you re-opened the database, the `USysRibbons` table was gone. It's not really gone, but merely hidden from the Navigation pane. Access has several system tables that it uses and these are normally hidden, that is, not included in the Navigation pane.

In addition, any table that you create that starts with "USys" is considered a user-system table and is also hidden. So when you first created the `USysRibbons` table it was available to you. However, when the database was re-opened, this was no longer visible.

To show both system and user-system tables, click the Options button in the Backstage View. Select the Current Database tab and click the Navigation Options button that is shown in Figure 10-19.

www.it-ebooks.info

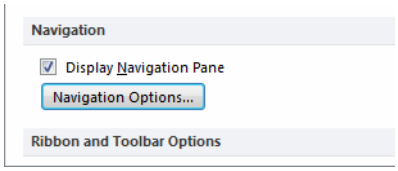


Figure 10-19. *Displaying the Navigation Options*

In the Navigation Options dialog, shown in Figure 10-20, you can control the types of objects that are included and how they are organized.

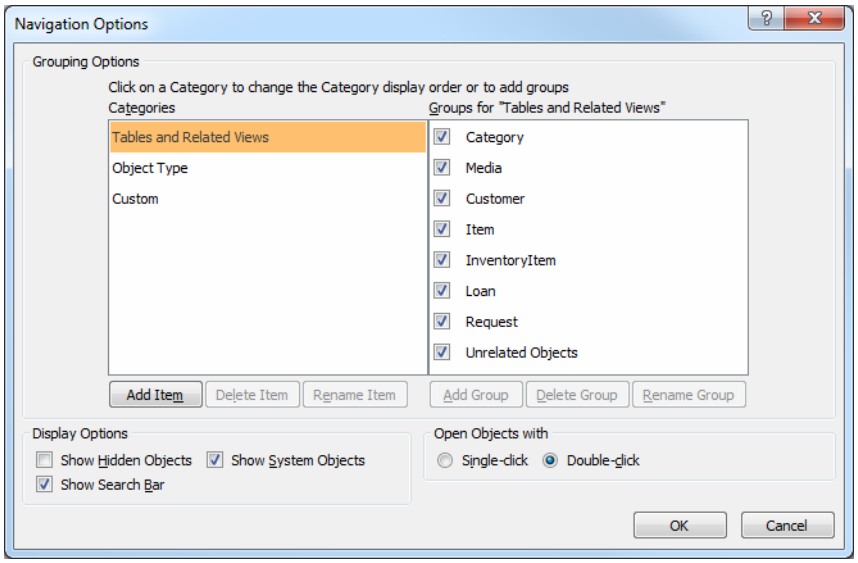


Figure 10-20. *Enabling the system objects*

Select the Show System Objects check box and click the OK button. Then click the OK button to close the Access Option dialog box. You'll see a pop-up saying the changes will not be applied until the database is reloaded. In this case, that's not true. The system tables are added to the Navigation pane immediately.

You should see the `USysRibbons` table in the Navigation pane and you can view and update its contents. You should also notice the `USysApplicationsLog` that you used in Chapter 3 to diagnose data macro errors.

Building a Custom Ribbon

Now that you have the basic concepts for building a custom ribbon, you'll develop a data-driven menu feature. This will allow you to design the menu options using a form and then generate the appropriate XML based on the data. As you add new options, the ribbon will automatically adjust.

As I explained earlier, a ribbon uses the following three-level hierarchy for organizing the controls:

www.it-ebooks.info

- Tabs
- Groups
- Commands

To accommodate this in your design, you'll have a table for each of these levels. You'll then create a form that will allow you to define each of these elements. Once the implementation is done and the data is entered, you'll write VBA code to extract the data and generate the corresponding XML script.

Designing the Tables

You'll start by creating three tables to define the three levels of the ribbon hierarchy. These tables are not really part of the Library data but are internal tables used by the application. To differentiate these, the name will have a "mnu" prefix. The instructions below are abbreviated; you can refer to Chapter 2 if you need more help in designing tables.

1. Create a `mnuTab` table using the following fields:
   - **TabID**: `AutoNumber`, primary key
   - **Label**: `Text` (100)
   - **Sequence**: `Number` (`Integer`)
2. Save the table and close it. Then create the `mnuGroup` table including the following fields:
   - **GroupID**: `AutoNumber`, primary key
   - **TabID**: `Lookup` (use the `mnuTab` table)
   - **Label**: `Text` (100)
   - **Description**: `Text` (255)
   - **Sequence**: `Number` (`Integer`)

■ **Note** When setting up the `Lookup` column, select both the `TabID` and `Label` fields. Use the defaults values for the remaining dialog boxes.

---

3. Save the table and close it. Finally, create the `mnuCommand` table as follows:
   - **CommandID**: `AutoNumber`, primary key
   - **GroupID**: `Lookup` (use the `mnuGroup` table)
   - **Label**: `Text` (100)
   - **Graphic**: `Text` (100)

www.it-ebooks.info

- **GraphicSize**: `Lookup` (enter the allowed choices as **large** and **normal**)
- **TargetType**: `Lookup` (enter the allowed choices as **Form**, **Report**, and **Macro**)
- **Target**: `Text` (100)
- **ScreenTip**: `Text` (100)
- **SuperTip**: `Text` (255)
- **Sequence**: `Number` (`Integer`)

■ **Note** When setting up the `GroupID` lookup column, select both the `GroupID` and `Label` fields. Use the remaining default values.

---

4. Open the `mnuTab` table in the Datasheet view. Add a single record with the `Label` field **Library** and `Sequence` =**1**.

Creating the Menu Forms

You'll start by generating a form for the `mnuCommand` table. Then you'll use this as a subform on the `mnuGroup` form. Because the number of tabs is small, you'll just enter these in the `mnuTab` table directly.

1. Select the `mnuCommand` table in the Navigation pane and click the Form button in the Create tab of the ribbon.
2. Delete the `CommandID` and `GroupID` controls and their associated labels.
3. Set the control padding to None.
4. Set the Text Align property for all of the labels to Right.
5. Remove the Form Header.
6. Change the Default View to Continuous Forms.
7. Re-arrange the controls to look like Figure 10-21.

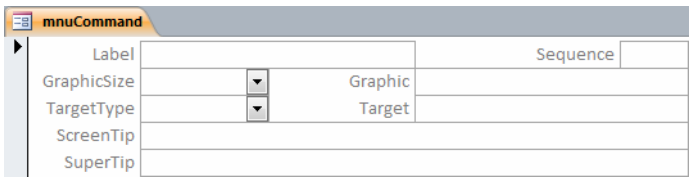


Figure 10-21. *The design of the* `mnuCommand` *form*

8. Save the form and enter the name **mnuCommand** when prompted. Now you'll create the `mnuGroup` form.

www.it-ebooks.info

9. Select the `mnuGroup` table in the Navigation page and click the Form button in the Create tab of the ribbon.

10. This will create Datasheet view to display the associated records of the `mnuCommand` table. Delete this and add a Subform control in its place. The Subform Wizard will find multiple fields in both table with the same name and will prompt you to select the correct field to use for linking the forms. Select the `GroupID` field, as shown in Figure 10-22.

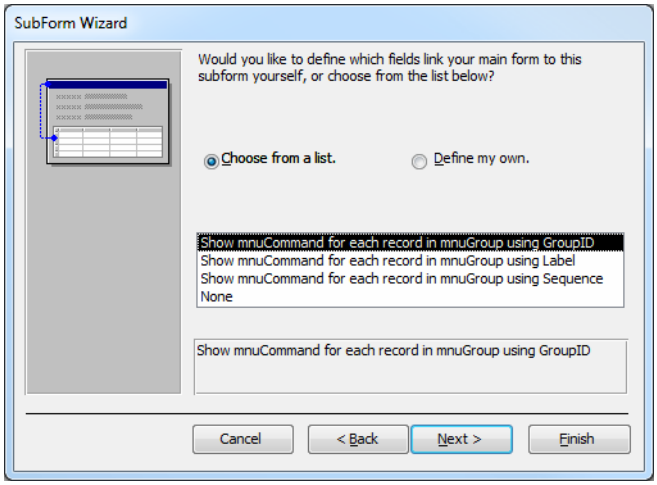


Figure 10-22. *Selecting the* `GroupID` *field for linking the forms*

11. Arrange the controls as shown in Figure 10-23.

www.it-ebooks.info

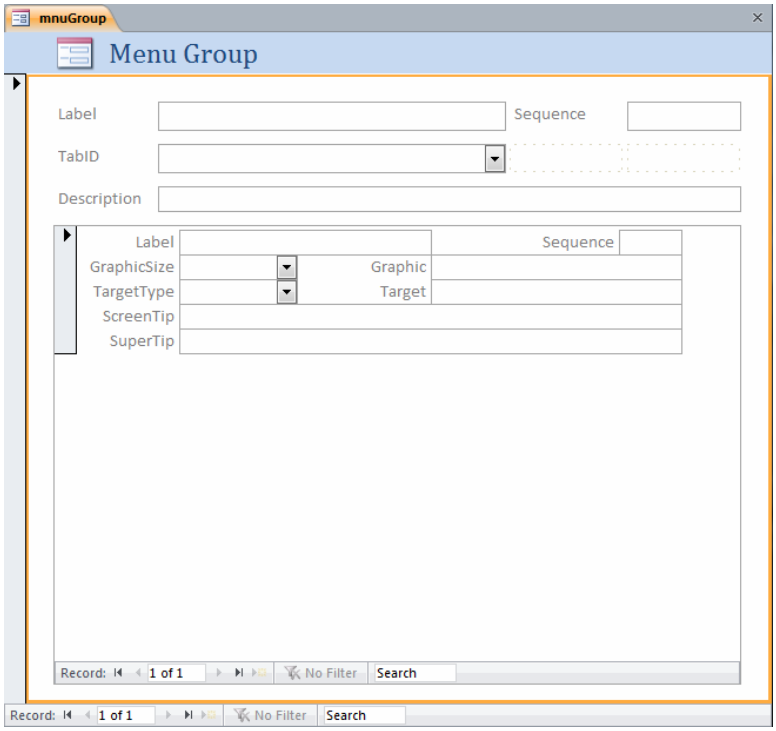


Figure 10-23. *The* `mnuGroup` *form layout*

Save the form and enter the name **mnuGroup** when prompted.

Populating the Menu Tables

Now you'll use the `mnuGroup` form to define the controls that will be placed on the ribbon. You'll follow the same basic structure that you used for the `Menu` form that you created earlier in the chapter. There will be three groups (`Operations`, `Administration`, and `Maintenance`). The first group will look like Figure 10-24.

www.it-ebooks.info

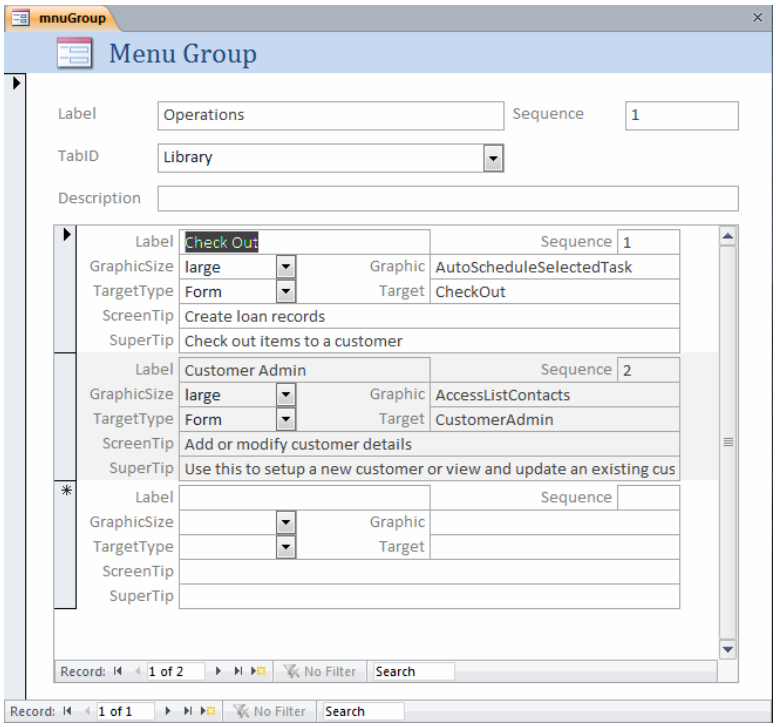


Figure 10-24. The `Operations` group definition

■ **Note** You'll be able to see all of the controls on the ribbon, so there's no need to include a link to the `Item` form on both the `Operations` and `Administration` groups. You'll just include it on the `Administration` group.

The definition of the `Administration` group should look like Figure 10-25.

www.it-ebooks.info

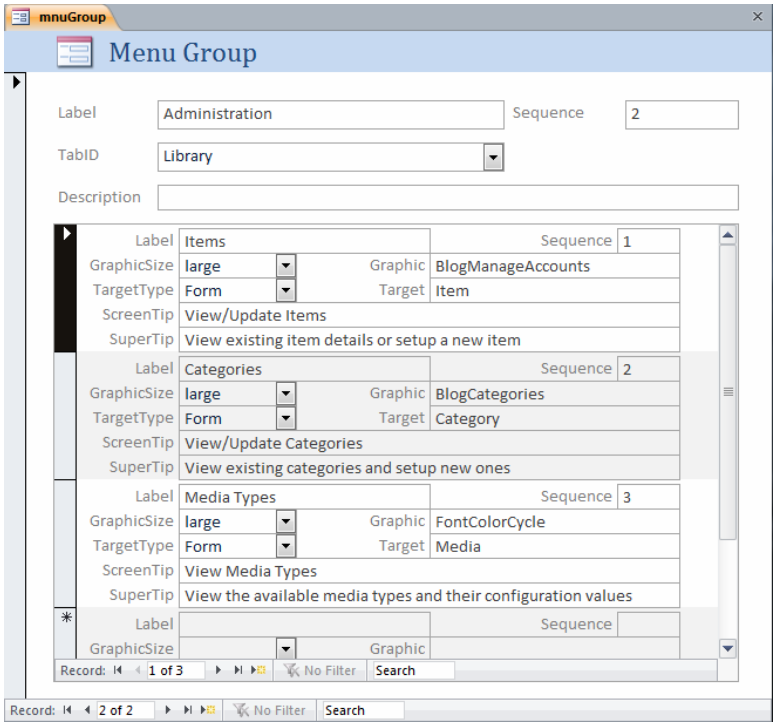


Figure 10-25. *The* `Administration` *group definition*

The `Maintenance` group definition will look like Figure 10-26.

www.it-ebooks.info

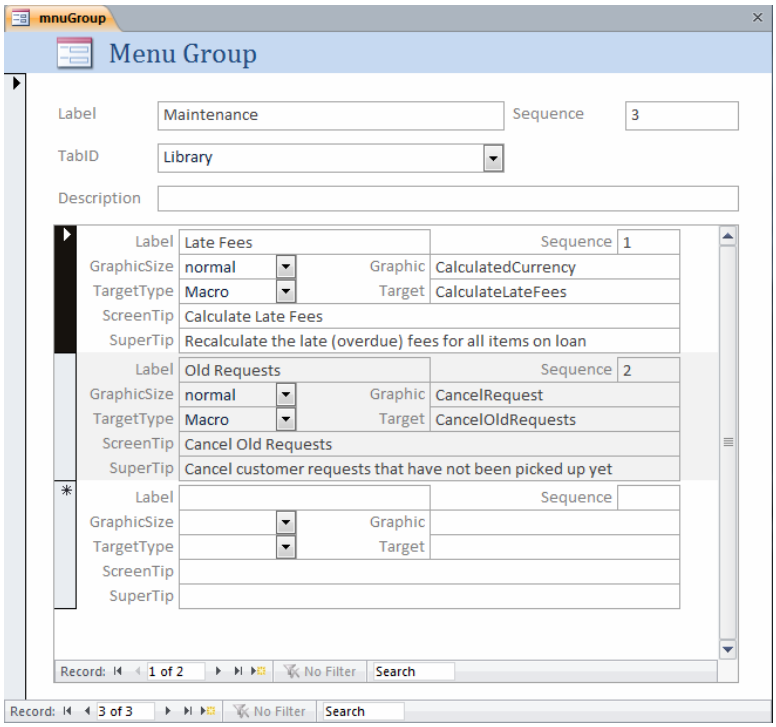


Figure 10-26. *The* `Maintenance` *group definition*

Using MSO Images

You're probably wondering about the unusual values for the `Graphic` fields. I also promised to explain about "HappyFace." Well, I'll take care of both of those now. Microsoft provides literally hundreds of built-in images that are available for you to use in your application. Just about any icon that exists on a ribbon in any of the Office products is included in this image library as well as icons from many other Microsoft products.

To use them all you need to do is specify the image name. The trick is knowing the correct name for each of the images. The best way to do that is to download a Word document from `www.microsoft.com`. Go to the following URL:

```
http://www.microsoft.com/downloads/en/details.aspx?displaylang=en&FamilyID=2d3a18a2-2e75-4e43-
8579-d543c19d0eed
```

Click the Download button and save the file to your local machine. Then open the Word document and go to the Backstage View. There are two tabs that together provide a gallery of all of the built in images. Click the tabs to see the images and their names, as shown in Figure 10-27.

www.it-ebooks.info

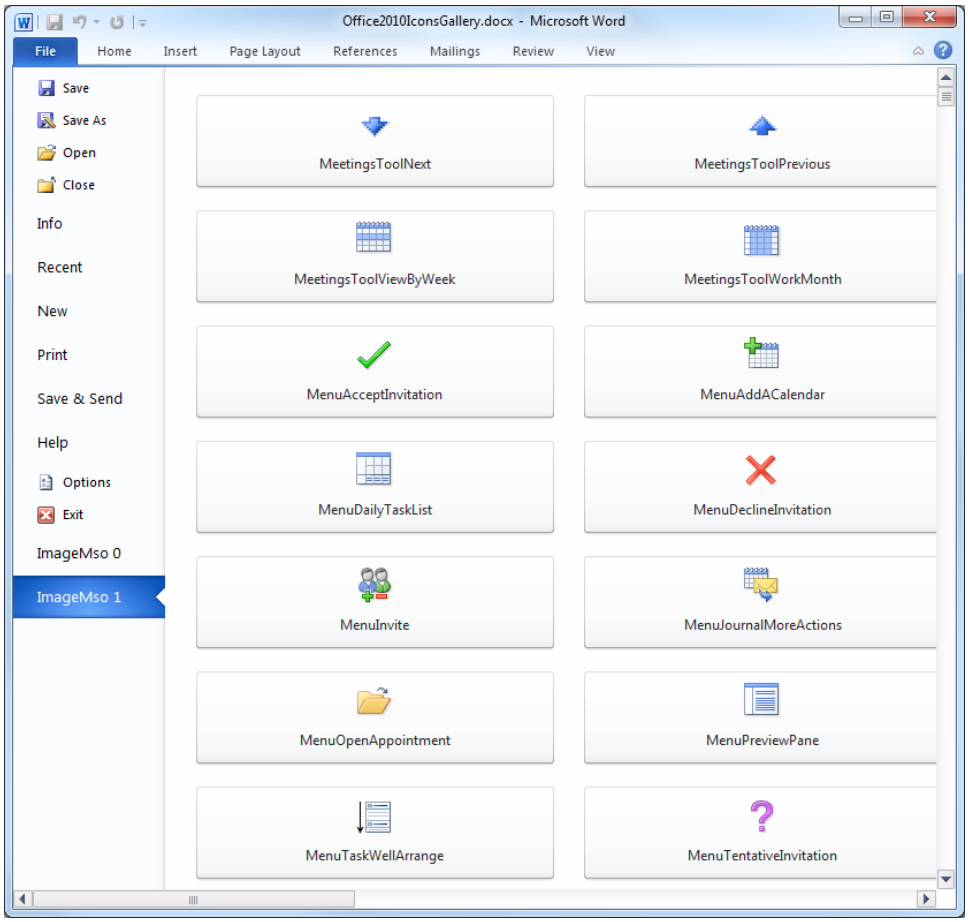


Figure 10-27. *Sample mso image gallery*

I made a quick scan of these images and picked some that seemed appropriate for the `Library` menu controls. Feel free to browse the gallery and choose different images. Just enter the appropriate image name in the `Graphic` field of the `mnuGroup` form.

Implementing the Custom Ribbon

Now that you have all of the ribbon elements defined (tabs, groups, and commands) you're ready to write the logic that reads this data and builds the appropriate ribbon XML. You'll start by designing a query that combines the three tables into a single record set. Then you'll execute this query in VBA code.

Designing the mnuCommands Query

In the Create tab of the ribbon, click the Query Design button. Add the following tables to the query:

www.it-ebooks.info

- mnuTab
- mnuGroup
- mnuCommand

The query should automatically setup the relationships between the tables. Add the following fields to the query:

- mnuTab.TabID
- mnuTab.Label
- mnuTab.Sequence
- mnuGroup.ID
- mnuGroup.Label
- mnuGoup.Description
- mnuGroup.Sequence
- mnuCommand.ID
- mnuCommand.Label
- mnuCommand.Graphic
- mnuCommand.GraphicSize
- mnuCommand.TargetType
- mnuCommand.Target
- mnuCommand.ScreenTip
- mnuCommand.SuperTip
- mnuCommand.Sequence

Uncheck the three Sequence fields so they are not returned in the result set and set the Sort property on these fields to Ascending. The Sequence fields are only included in the query to be used to sort the records.

There is a Label field from all three tables. To avoid ambiguity, you'll give then unique names in the result set. For the mnuTab.Label field, enter the Field name as **TabLabel: Label**. Likewise, for the mnuGroup.Label field, enter **GroupLabel: Label**.

Save the query and enter the name **mnuCommands** when prompted. The query design should look like Figure 10-28.

www.it-ebooks.info

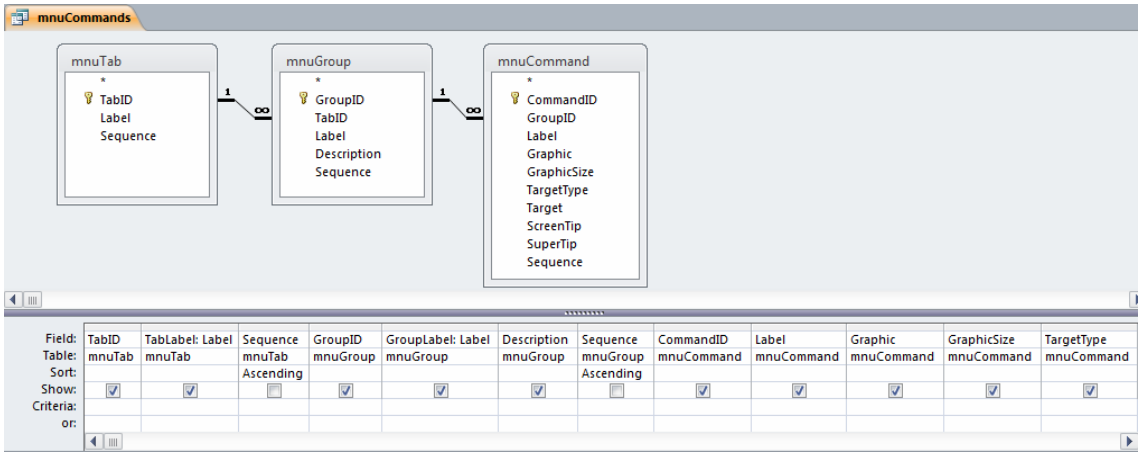


Figure 10-28. *The* `mnuCommands` *query design*

Switch to the Datasheet View and verify that there are seven rows returned.

Generating the Ribbon XML

Now you'll write the VBA code that will generate the appropriate XML based on the configuration data. Open the `Main` code file and enter the code shown in Listing 10-3.

Listing 10-3. *The Implementation of* `GetRibbonDefinition`

```
Function GetRibbonDefinition() As String

Dim txtXML As String
Dim nTabID As Integer
Dim nGroupID As Integer

nTabID = 0
nGroupID = 0

' Add the XML header info
txtXML = "<customUI xmlns=" & Chr(34) & _
"http://schemas.microsoft.com/office/2009/07/customui" & Chr(34) & ">" & vbCrLf & _
"  <ribbon startFromScratch=" & Chr(34) & "false" & Chr(34) & ">" & vbCrLf & _
"    <tabs>" & vbCrLf

    Dim rs As DAO.Recordset
    Set rs = CurrentDb.OpenRecordset("mnuCommands")

    Do Until rs.EOF
        If (rs.Fields("GroupID") <> nGroupID And nGroupID <> 0) Then
            ' Close the previous group
            txtXML = txtXML & "        </group>" & vbCrLf
```


www.it-ebooks.info

```
End If

If (rs.Fields("TabID") <> nTabID And nTabID <> 0) Then
    ' Close the previous tab
    txtXML = txtXML & "      </tab>" & vbCrLf
End If

If (rs.Fields("TabID") <> nTabID) Then
    ' Save the current tab ID
    nTabID = rs.Fields("TabID")

    ' Open the next tab
    txtXML = txtXML & "      <tab id=" & Chr(34) & "tab" & nTabID & _
             Chr(34) & " label=" & Chr(34) & rs.Fields("TabLabel") & _
             Chr(34) & ">" & vbCrLf
End If

If (rs.Fields("GroupID") <> nGroupID) Then
    ' Save the current group ID
    nGroupID = rs.Fields("GroupID")

    ' Open the next group
    txtXML = txtXML & "        <group id=" & Chr(34) & "group" & _
             nGroupID & Chr(34) & " label=" & Chr(34) & _
             rs.Fields("GroupLabel") & Chr(34) & ">" & vbCrLf
    If (Len(rs.Fields("Description")) > 1) Then
        txtXML = txtXML & "          <labelControl id=" & Chr(34) & _
                 "label" & nGroupID & Chr(34) & " label=" & Chr(34) & _
                 rs.Fields("Description") & Chr(34) & " />" & vbCrLf
    End If

End If

' Add a command button
txtXML = txtXML & _
   "          <button " & _
                  "id=" & Chr(34) & "button" & _
                      rs.Fields("CommandID") & Chr(34) & vbCrLf & _
   "                  size=" & Chr(34) & rs.Fields("GraphicSize") & _
                      Chr(34) & vbCrLf & _
   "                  label=" & Chr(34) & rs.Fields("Label") & _
                      Chr(34) & vbCrLf & _
   "                  getImage=" & Chr(34) & "GetImage" & _
                      Chr(34) & vbCrLf & _
   "                  screentip=" & Chr(34) & rs.Fields("ScreenTip") & _
                      Chr(34) & vbCrLf & _
   "                  supertip=" & Chr(34) & rs.Fields("SuperTip") & _
                      Chr(34) & vbCrLf & _
   "                  tag=" & Chr(34) & rs.Fields("Graphic") & _
                      Chr(34) & vbCrLf & _
   "                  onAction=" & Chr(34) & "OnMenuAction" & _
                      Chr(34) & " />" & vbCrLf
```


www.it-ebooks.info

```
        rs.MoveNext
    Loop

    txtXML = txtXML & "        </group>" & vbCrLf
    txtXML = txtXML & "      </tab>" & vbCrLf
    txtXML = txtXML & "    </tabs>" & vbCrLf
    txtXML = txtXML & "  </ribbon>" & vbCrLf
    txtXML = txtXML & "</customUI>"

    GetRibbonDefinition = txtXML

    rs.Close
    Set rs = Nothing

End Function
```

This code may look complicated, but it just executes the `mnuCommands` query and does a lot of string manipulation to construct a properly formed XML file. The structure of the XML is identical to the one I presented at the beginning of the chapter. Instead of hard-coded data, it is read from the query results and it allows for multiple tabs and groups.

The next step is to write code to load the ribbon. To do that, add another method to the `Main` module using the following code:

```
Public Function LoadRibbon() As Integer
    On Error Resume Next
    Application.LoadCustomUI "Library", GetRibbonDefinition
End Function
```

This code calls the `LoadCustomUI` method passing in a hard-coded ribbon name of **Library** and calling the `GetRibbonDefinition` method, which supplies the XML. At the beginning of the chapter you populated the `USysRibbons` table with a `RibbonName` and `RibbonXML` fields. Access reads this table on startup and does the same thing as the `LoadCustomUI` call. The `LoadCustomUI` method will fail if a ribbon with the same name already exists. The `On Error Resume Next` statement is used to ignore this error.

Implementing the Callback Methods

You already have an implementation for the two callback methods, `GetImage` and `OnMenuAction`. Now you'll need to replace that demo code with a real implementation. Replace these with the code shown in Listing 10-4.

Listing 10-4*. The Implementation of the Callback Methods*

```
Public Sub GetImage(ByVal control As Office.IRibbonControl, ByRef image)

    image = control.Tag

End Sub
```


www.it-ebooks.info

```
Public Sub OnMenuAction(ByVal control As Office.IRibbonControl)

    Dim rs As DAO.Recordset
    Set rs = CurrentDb.OpenRecordset("mnuCommand", dbOpenDynaset)

    rs.FindFirst "[CommandID]=" & Mid(control.ID, 7)
    If rs.NoMatch Then
        MsgBox "You've clicked the button " & control.ID & " on the Ribbon"
    Else
        Select Case rs.Fields("TargetType")
            Case "Form"
                DoCmd.OpenForm rs.Fields("Target"), acNormal
            Case "Report"
                DoCmd.OpenReport rs.Fields("Target"), acViewPreview
            Case "Macro"
                DoCmd.RunMacro rs.Fields("Target")
            Case Else
                MsgBox "You've clicked the button " & control.ID & " on the Ribbon"
        End Select
    End If
End Sub
```

For the `GetImage` callback, instead of supplying a hard-coded image name, it uses the Tag property. If you look at the implementation of the `GetRibbonDefinition` method, you'll notice the tag attribute is set using the `Graphic` field from the `mnuCommands` query.

The implementation of `OnMenuAction` uses the `ID` property to determine which button was clicked. The `id` attribute was generated using the `CommandID` field prefixed with the "button" text. This prefix is stripped off and then the `mnuCommand` table is queried using the `CommandID`. This method then calls either `OpenForm`, `OpenReport`, or `RunMacro` depending on the `TargetType` column. The `Target` column contains the name of the form, report or macro that should be used. If the `mnuCommand` record was not found or is not one of the supported types, this method just displays the button ID.

Using the Autoexec Macro

Now you'll need to call the `LoadRibbon` method when the database is opened. Access provides an easy way to do that by using the `autoexec` macro. If you create a macro with that exact name, **autoexec**, Access will run that for you when the database is opened. You can then put whatever startup logic you need inside this macro.

From the Create tab of the ribbon, click the Macro button, which will display the Macro Designer. Select the `RunCode` action. Enter **LoadRibbon**() for the Function Name parameter.

■ **Caution** Make sure that you include the parentheses. These are needed even if there are no parameters being passed to the function.

---

www.it-ebooks.info

Save the macro and enter the name **autoexec** when prompted. The macro should look like Figure 10-29.

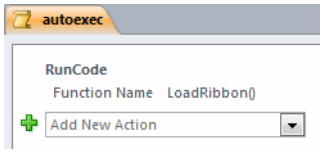


Figure 10-29. *The* `autoexec` *macro*

Now try running this macro. You can click the Run button in the Design tab of the ribbon or double-click it in the Navigation pane. You won't notice it doing anything. You can add a breakpoint to the `GetRibbonDefinition` method if you want to step through this code.

Selecting the Library Ribbon

At this point the `Library` ribbon has been installed as well as the `Demo` ribbon that you added to the `USysRibbons` table. As before, you have to tell access to use this ribbon before it is visible.

Click the Options button in the Backstage View. On the Current Database tab, expand the Ribbon Name combo box and select the `Library` ribbon as shown in Figure 10-30.

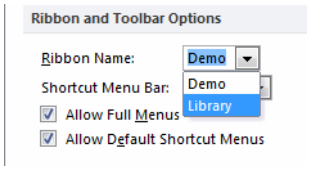


Figure 10-30. *Selecting the* `Library` *ribbon*

The database is now configured to use the new `Library` ribbon when it is restarted.

■ **Note** You should remove the existing `Demo` ribbon that you loaded earlier. Open the `USysRibbons` table and delete the row that is in this table.

Close the Access database and then re-open it. You should have a new `Library` tab that looks like Figure 10-31.

www.it-ebooks.info

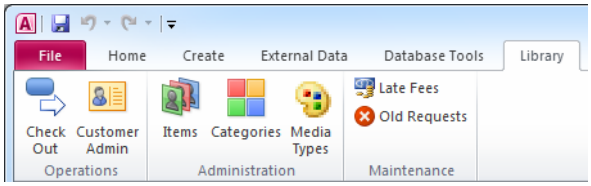


Figure 10-31. *The new* `Library` *ribbon tab*

Try clicking some of the buttons in the `Library` ribbon. This should open the associated form. If you hover the mouse over one of the buttons, the text you supplied for the `ScreenTip` and `SuperTip` fields should be displayed as demonstrated in Figure 10-32.

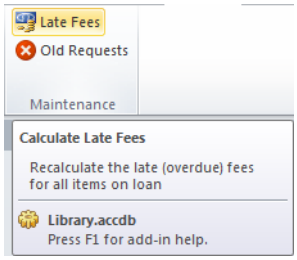


Figure 10-32. *The custom hover text*

■ **Tip** I have shown you two ways to provide custom navigation; a custom form with buttons that use macros to launch a form and a custom ribbon. I'll show you a third way in Chapter 15, which is to use a Navigation Form. As I mentioned, it doesn't work for many of your existing forms. There is also a fourth approach, which is to use a *Switchboard*. In Access 2010 this feature is still available but they didn't make it easy to find. This is provided for legacy purpose so I didn't cover it but wanted you to know about, for completeness.

Locking Down the Database

You have provided two different ways for the user to navigate to the various forms that are available to them. Now that this is in place you can lock down the database by removing access to the developer-oriented functions. You'll do this in two phases: first by removing much of the menu and navigation. The user will then only see the items that they should be using. Second, because a knowledgeable user can easily circumvent this, you'll compile a executable-only version of the database that has most of the developer features removed.

www.it-ebooks.info

■ **Tip** I will show you how to lock the database and you should try these options to see what the effect is. However, be sure you undo the changes when you're done as there is still more development to do.

Removing Navigation

By unchecking a few check boxes, you can drastically limit the features available to the end users. This will make Access look more like an application and less like a development platform.

Let's start with the Windows title bar, which probably reads something like "Library : Database (Access 2007) – Microsoft Access." This gives the impression that you have opened an Office document, which, of course, you have. However, you can change this title so it looks like you have launched an application.

1. Open the Backstage View, click the Options button, and select the Current Database tab.
2. At the top of the dialog, enter **Library Management System (LMS)** for the Application Title.
3. While you're here, you should fix the icon as well. You've probably noticed that the icon in the task bar is the standard Microsoft Access 2010 icon. Click the Browse button and select an icon file. (If you've downloaded my `Images\Static` folder, you can use the `Library.ico` file. Otherwise just search the local drive for `*.ico` files.)
4. Select the Use As Form And Report Icon option. The top portion of the Application Options section should look like Figure 10-33.

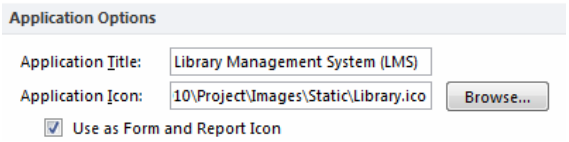


Figure 10-33. *Setting the application options*

5. Click the OK button to save the changes. Notice the icon in the top-left corner of the application and the task bar has changed as well as the title bar.

Now you'll remove the standard navigation options.

6. Go back to the Backstage View, click the Options button and select the Current Database tab.
7. Unselect the following options, as highlighted in Figure 10-34:
   - Use Access Special Keys
   - Enable Layout View

www.it-ebooks.info

- Enable design changes for tables in Datasheet view
- Display Navigation Pane
- Allow Full Menus
- Allow Default Shortcut Menus

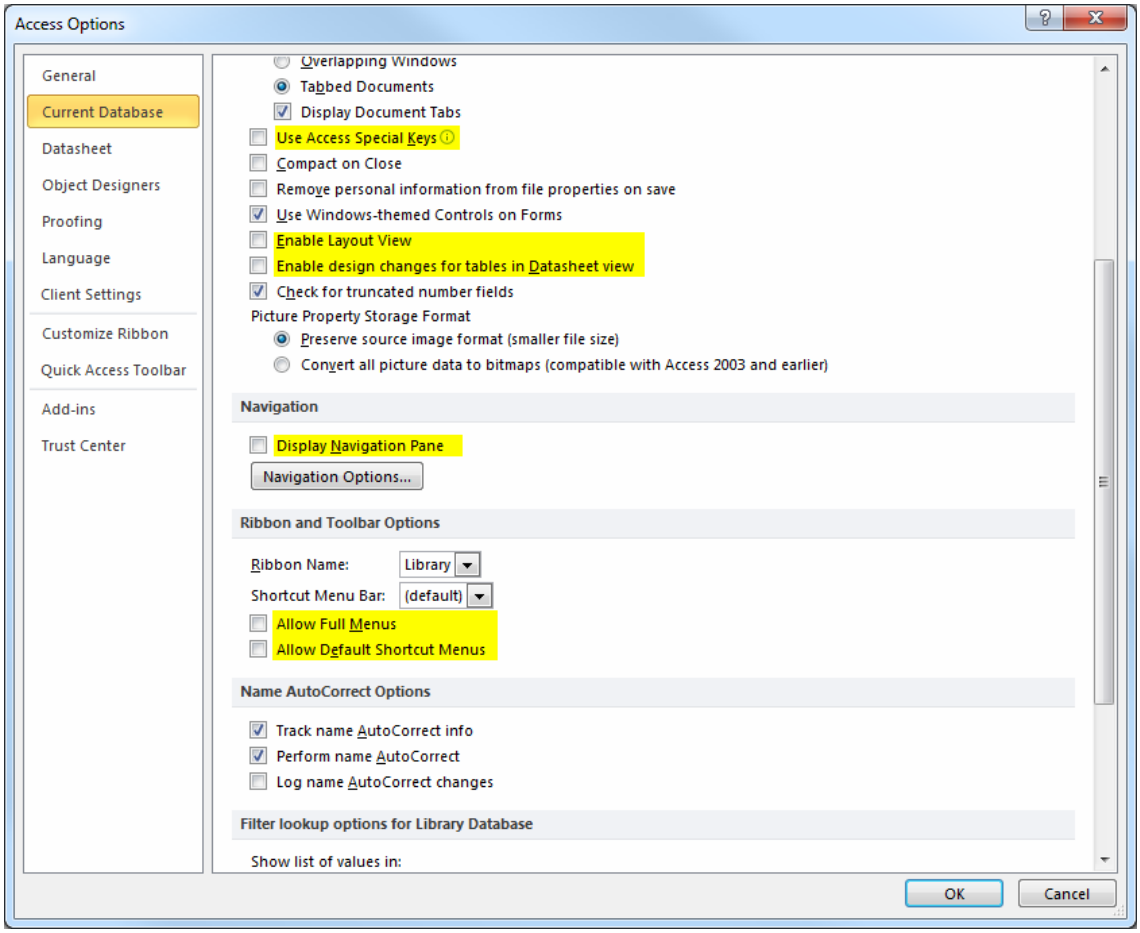


***Figure 10-34**. Removing the navigation options*

www.it-ebooks.info

■ **Note** Some of these options are redundant. For example, the "Enable Design Changes for Tables in Datasheet View" doesn't really need to be turned off, because the user can't get to the tables anyway. But it's still a good idea to disable this feature. If you decide later to allow navigation to the tables, you'll want to prevent the user from changing them.

8. Click the OK button to close the form. You will need to close the application and re-open it for these changes to take effect. Your database should look like Figure 10-35.

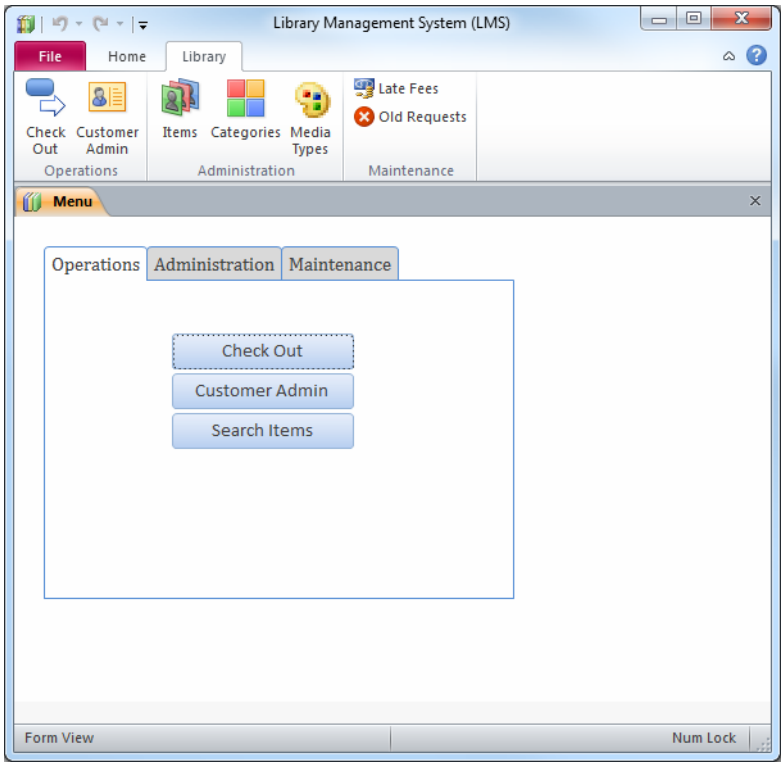


***Figure 10-35**. The database with navigation removed*

Click the File tab to see the Backstage View. You'll notice that is has been changed to pretty much bare bones as well. You may be wondering how you are going to turn these options back on so you can keep developing. Well, don't panic. Click the Privacy Options button as shown in Figure 10-36. This displays the Access Options dialog box that you can use to enable the standard navigation.

www.it-ebooks.info

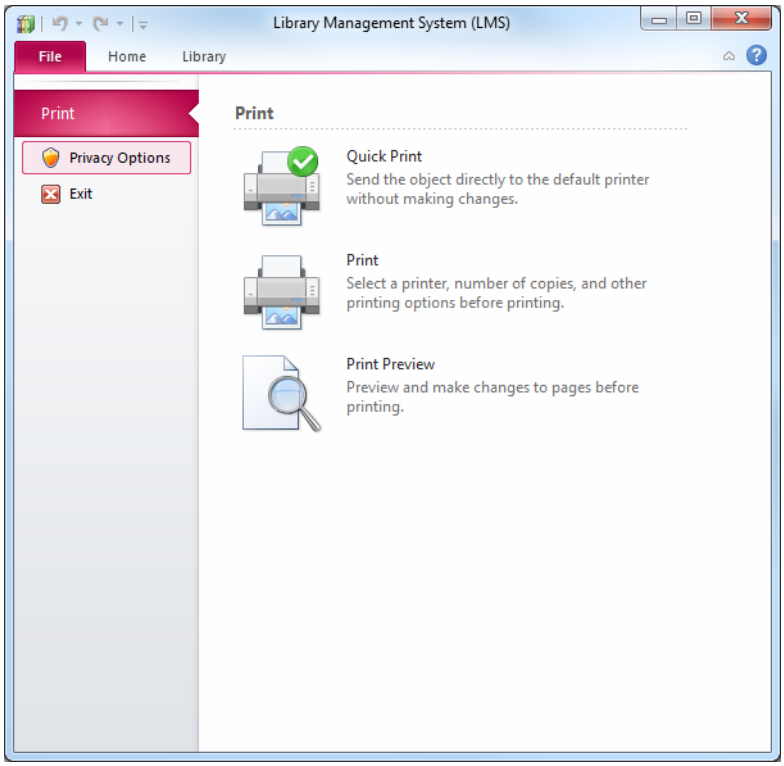


Figure 10-36. *Using the Privacy Options button*

Compiling the Database

As you can see, even though you have removed access for most users, there's nothing to prevent someone from bypassing this and turning the navigation and menus back on. However, Access gives you a way to create a version of the database that does not contain any of the code and that does not allow the Design Views. First, turn on the "Allow Full Menus" option, if they're not currently enabled.

1. Go to the Backstage View and click the "Save & Publish" tab.
2. Select the "Save Database As" option and then select the Make ACCDE format in the Advanced section as shown in Figure 10-37.

www.it-ebooks.info

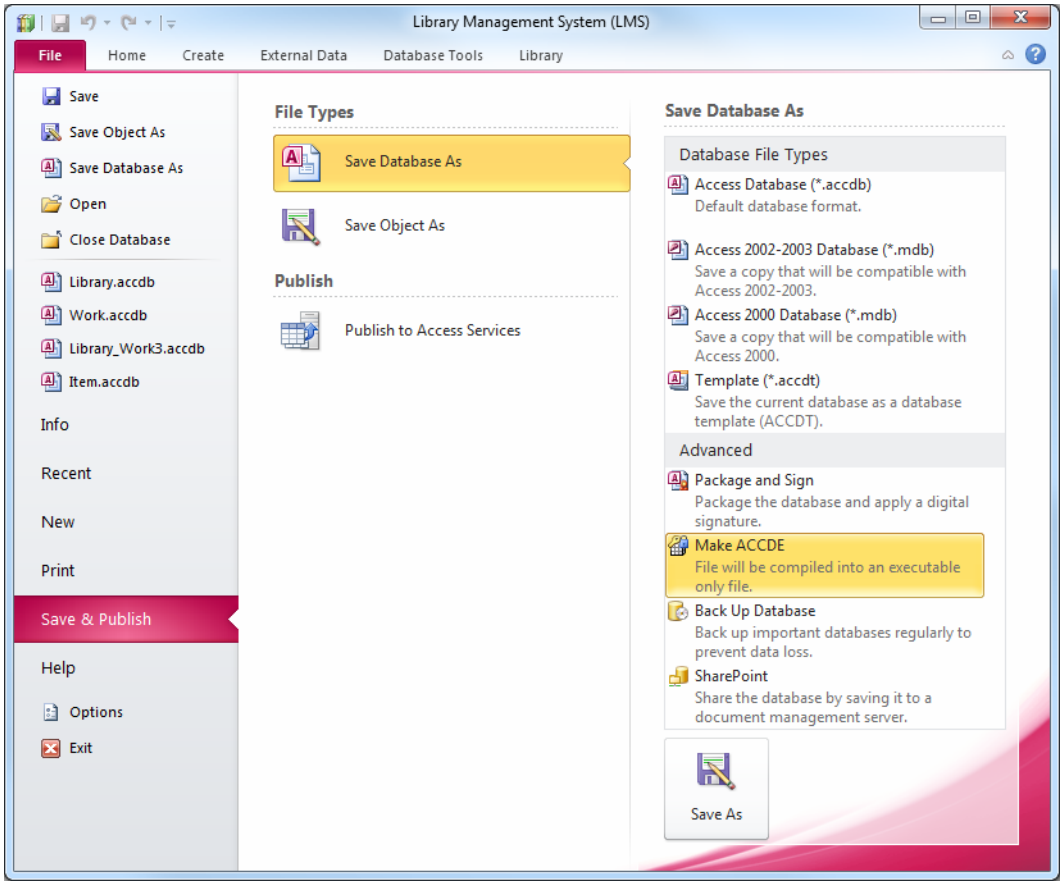


Figure 10-37. *Saving as an ACCDE file*

3. Click the Save As button, which will display a Save As dialog box. You can use the default file name, which should be `Library.accde`. This will create a new file with the `.accde` extension. When this has finished, close the existing database. Then open the new `Library.accde` file. You'll need to turn off the Allow Full Menus options.

The `Library.accde` is now ready for the end users. If you turn on the Display Navigation Pane option, for example, you'll be able to browse the various database objects but you cannot modify them. You will still need to have Access installed to be able to open this file. In Chapter 14, I will show you how to distribute this file and use the Access Runtime so you can deploy this without needing to purchase a copy of Access for every end user.

www.it-ebooks.info

Summary

In this chapter you packaged your application so it is "user-ready." This implemented two methods for navigating to the various forms: a simple navigation form and a custom ribbon. You also developed a data-driven menu system so the ribbon controls are generated automatically based on the menu configuration. As you add new features to the database, you'll just need to add them to the menu configuration and when the database is reloaded, the associated control will be added to the ribbon.

You also locked down the database to remove the developer features. The first step was to remove the navigation to these features. You then compiled a .accde file that has the development features removed.

The key concepts that were introduced in the chapter include:

- Using the Command Button Wizard to create macros to launch a form or run a macro
- Creating a custom UI ribbon
- Adding custom ribbons to the `USysRibbons` table
- Using mso images
- Creating an `autoexec` macro
- Configuring the Access navigation options
- Compiling a locked-down version of your database

In the next chapter you will look at some more of the visual aspects of your Access database such as colors, fonts and images. This will enable you to quickly give your application a whole new look.

www.it-ebooks.info

CHAPTER 11

Branding with Themes and Styles

Branding is a marketing concept that that focuses on product identification or familiarity. It goes as far back as the days of the large cattle-drives, where a cow was branded with a symbol so everyone would know whom it belonged to. The idea is that people recognize colors and shapes that they are familiar with. You can go to a store and pick out a product without even reading the label, just because you recognize the color or shape of the package.

People expect that same familiarity when it comes to software applications. If you go to `pizzahut.com` to order lunch and don't see the red roof logo, you will probably consider the site suspect and order somewhere else. There is comfort in familiarity. When you develop your Access applications, you will likely have to deal with the necessity for branding.

In this chapter, you will brand your `Library` application to look like the `apress.com` website. It's fairly easy to do with just a few simple techniques. I will first explain how to use themes, as this will go a long way to mirror the colors and fonts. Then I'll show you how to use the image gallery to place a few graphics on the forms. The end result will be a metamorphosis of the user experience.

Using Office Themes

Each Office product allows you to create, view, and edit a type of document, such as a text document (Word), spreadsheet (Excel), or presentation (PowerPoint). There is a fairly clear distinction between the content (the data supplied by and maintained by the users) and the application (the menus, ribbons, and forms) provided by the Office application. Office themes allow users to customize the colors and fonts used in the application areas but have no effect on the content. Thus, two users can open up the exact same document and have a completely different look and feel.

Access is sort of the odd man out when it comes to Microsoft Office products, because your Access database includes forms. So the users will see both built-in Access UI elements as well as UI elements that are included in the content, such as forms. In the last chapter, I showed you how to remove as much of the Access UI as possible. However, you can't remove everything (and you don't want to). So you'll have your own custom UI elements, but they will still run under the Office UI framework.

I will first explain how to customize the standard Office UI, and then we'll look at how that spills over into your form design.

Understanding Office Themes

I think the best way to fully understand themes is to create one for yourself. I'll show you how to define a custom theme, and then we'll look into how it works. If you open any form in the Design View or the Layout View, in the Design tab of the ribbon you'll see the Themes group shown in Figure 11-1.

www.it-ebooks.info

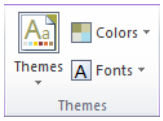


Figure 11-1*. The Themes group of the Design tab*

Click the Themes button and you'll see a list of themes, shown in Figure 11-2, that you can select from. As you hover the mouse over each of the choices, you can see your form redrawn based on the selected theme.

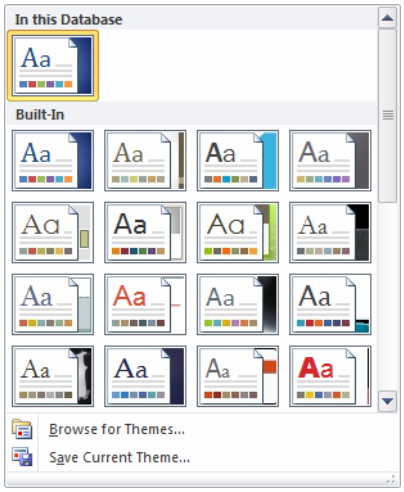


Figure 11-2*. The available themes*

This dialog is divided into several sections. The first shows the theme that is currently defined for this database. If you have any custom themes, they will be displayed next. Finally, the built-in themes are listed. Just click off this dialog box to close it; you won't change the theme at this time.

Creating a Custom Color Theme

A theme is comprised of a color scheme and a font scheme. You define these separately and then later you can save them as a theme. Let's start with the color scheme.

1. Click the Colors button in the ribbon. You should see a fairly long list of existing color schemes, as shown in Figure 11-3.

www.it-ebooks.info

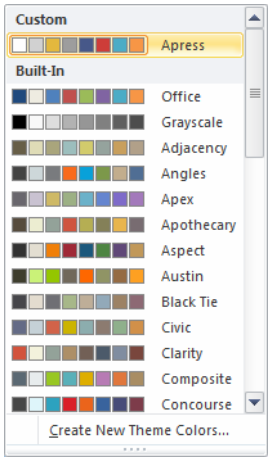


Figure 11-3. *Viewing the available color schemes*

2. As you can see, the custom and built-in color schemes are listed separately. I have already created a color scheme named `Apress`. Click the *Create New Theme Colors* link at the bottom of this list, which will display the Create New Theme Colors dialog box shown in Figure 11-4.

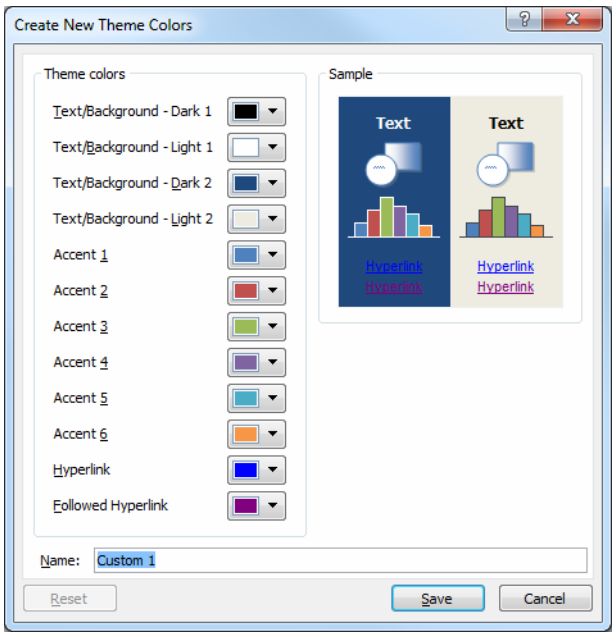


Figure 11-4. *The Create New Theme Colors dialog box*

www.it-ebooks.info

■ **Note** A color scheme is simply a collection of twelve colors. There are two pairs of background/foreground colors. Normally you will use the light foreground with the dark background and vice versa. This allows you to visually toggle between selected items and non-selected items. There are also six accent colors and two hyperlink colors.

■ **Tip** In my opinion, the names given to the first four colors are difficult to follow. The first two are the primary foreground and background colors. The next two are the alternate foreground and background colors. When you need to select a color from either the Property Sheet or the color picker, different names are used. Fortunately, those names actually make sense (Text 1, Background 1, Text 2, Background 2). Also, note that the preview picture is wrong and doesn't display the colors correctly. Just ignore this.

---

3. The first two colors are just standard black and white, and you can leave this as it is. The fourth color, which is called Text/Background – Light 2, should be changed. Click the dropdown icon next to it, which will reveal the standard color picker, shown in Figure 11-5.

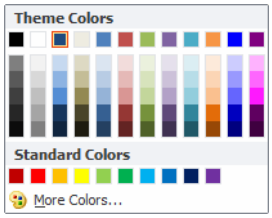


Figure 11-5. *The standard color picker*

4. The top portion of the color picker shows the theme colors. There are a row of twelve colors, each one representing the current theme colors. Below each are five shades of that color. The idea is that once you have picked the colors you will use, from then on you normally pick from those colors or possibly a lighter or darker shade of one of these colors. However, at this point you are still defining the colors, so this is not helpful. Instead, click the *More Colors* link to display the Colors dialog box shown in Figure 11-6.

www.it-ebooks.info

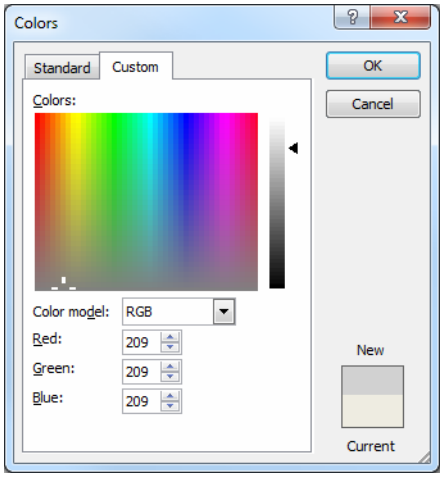


Figure 11-6. *Defining the theme color*

5. Go to the Custom tab to select the appropriate color. What I did to eyeball the correct color is drag this dialog on top of the color I was trying to match. Then I moved the crosshairs to get the correct color and moved the slider to adjust the brightness. I did that until the preview of the New color matched. Normally, however, your client should be able to tell you the correct settings for each color. For this color enter the following values:
   - Red: **209**
   - Green: **209**
   - Blue: **209**
6. Click OK to update this color.
7. Repeat this procedure to set the color for the remaining colors using the values in Table 11-1. You can leave the hyperlink colors as they are.

Table 11-1. *Accent Color Definitions*

Theme Color	Red	Green	Blue
Text1	0	0	0
Background 1	255	255	255
Text 2	255	255	255
Background 2	209	209	209
Accent 1	227	184	61

www.it-ebooks.info

Theme Color	Red	Green	Blue
Accent 2	157	157	157
Accent 3	71	88	137
Accent 4	205	61	57
Accent 5	75	175	198
Accent 6	61	58	68

8. When you have finished, enter the Name **Apress**, and click the Save button.

Defining the Font Scheme

Setting up a font scheme is pretty simple: there are only two font families that you'll need to define. At first glance, that may seem too simplistic. However, a good UI design uses a consistent font throughout the application. Too many different fonts can make the design seem cluttered and disorganized. It is a really good design practice to limit the design to only two fonts. You can always use fonts that are not included in the theme, but in most cases you should stay with these two font families.

9. Click the Fonts button on the ribbon, which will list the existing font schemes as shown in Figure 11-7.

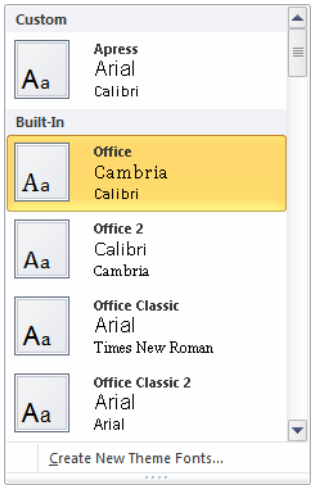


Figure 11-7. *The existing font schemes*

10. Again, the custom and built-in schemes are separated. Click the *Create New Theme Fonts* link, which will display the Create New Theme Fonts dialog box.

11. Select Arial for the heading font and Calibri for the body font.

www.it-ebooks.info

12. Enter **Apress** for the Name, as shown in Figure 11-8. Click the Save button to update the font scheme.

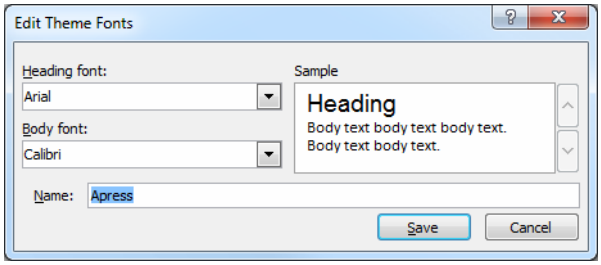


Figure 11-8. *Defining a new font scheme*

Creating an Office Theme

At this point, you have created a color scheme and a font scheme. Access has already selected these as the schemes defined for the `Library` database. You've probably already noticed that your forms have changed appearance because they are now using the new color and font schemes that you defined. Now you'll save the current configuration as a custom theme.

Click the Themes button in the ribbon, which will display the list of existing themes, as shown in Figure 11-2. At the bottom of this list, click the *Save Current Theme* link. In the Save Current Theme dialog box, enter the name `Apress.thmx`, as shown in Figure 11-9.

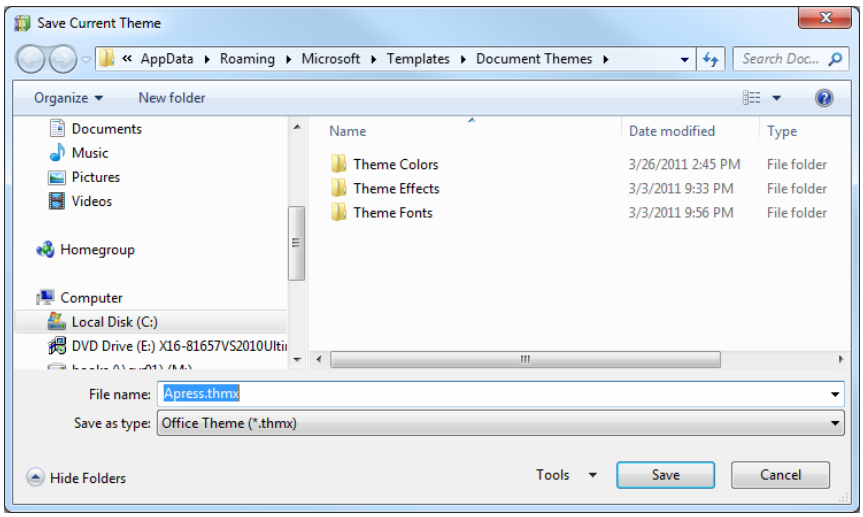


Figure 11-9. *Saving the* `Apress.thmx` *file*

www.it-ebooks.info

The theme definition is saved in a file with a `.thmx` extension. Like all Office files ending in "x" this is actually a compressed folder that contains several files and subfolders. It includes a `.jpg` file that is used for the icon that is displayed in the list of themes. There is also an `.xml` file that provides the theme definition. A portion of that is shown in Listing 11-1.

Listing 11-1. *Partial Listing of* `theme.xml`

```
<a:themeElements>
  <a:clrScheme name="Apress">
    <a:dk1><a:sysClr val="windowText" lastClr="000000"/></a:dk1>
    <a:lt1><a:sysClr val="window" lastClr="FFFFFF"/></a:lt1>
    <a:dk2><a:srgbClr val="FFFFFF"/></a:dk2>
    <a:lt2><a:srgbClr val="D1D1D1"/></a:lt2>
    <a:accent1><a:srgbClr val="E3B83D"/></a:accent1>
    <a:accent2><a:srgbClr val="9D9D9D"/></a:accent2>
    <a:accent3><a:srgbClr val="475889"/></a:accent3>
    <a:accent4><a:srgbClr val="CD3D39"/></a:accent4>
    <a:accent5><a:srgbClr val="4BACC6"/></a:accent5>
    <a:accent6><a:srgbClr val="3D3A44"/></a:accent6>
    <a:hlink><a:srgbClr val="0000FF"/></a:hlink>
    <a:folHlink><a:srgbClr val="800080"/></a:folHlink>
  </a:clrScheme>
  <a:fontScheme name="Apress">
    <a:majorFont>
      <a:latin typeface="Arial"/><a:ea typeface=""/><a:cs typeface=""/>
    </a:majorFont>
    <a:minorFont>
      <a:latin typeface="Calibri"/><a:ea typeface=""/><a:cs typeface=""/>
    </a:minorFont>
  </a:fontScheme>
  <a:fmtScheme name="Composite">
    <a:fillStyleLst>
      <a:solidFill><a:schemeClr val="phClr"/></a:solidFill>
      <a:gradFill rotWithShape="1">
```

This theme definition specifies the twelve colors and the two fonts as you would expect. It also defines some formatting schemes including fills and gradients. These are inherited from the standard Office effects.

■ **Note** In Word and Excel, you can also define effects, which control how lines and fills are drawn. This is not possible in Access. Your custom theme will include the effects defined by the default theme.

Notice the location of the `Apress.thmx` file. It is in the following folder:

```
%USERPROFILE%\AppData\Roaming\Microsoft\Templates\Document Themes
```


www.it-ebooks.info

If you want to share this theme with other users, you'll need to send a copy for them to install on their local machine. As an alternative, you can place this in a shared location. Any user who wants to use this theme can add it by browsing to that location and selecting this file.

The extra step of creating the `Apress.thmx` file is not necessary for the `Library` application. In your Access database, all you need to do is define the color and font scheme. The color and font schemes are stored in the database and will be used when other users open the database file. The theme file is used to share these schemes with other Office applications. For example, open Word or Excel and notice the `Apress` theme is now available, as demonstrated in Figure 11-10.

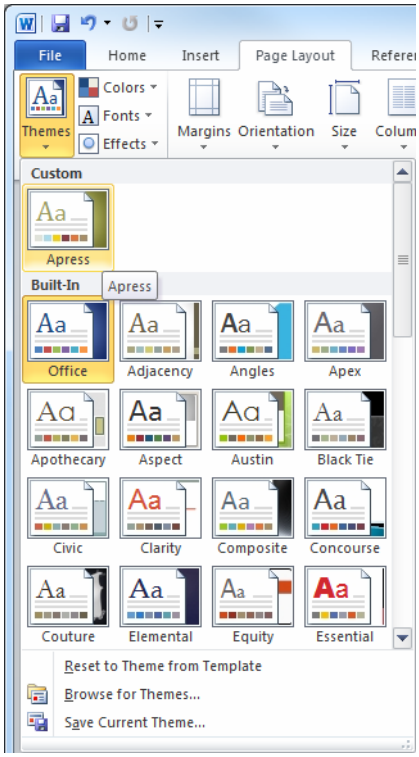


Figure 11-10*. The theme is available in all Office applications.*

The reverse of this also true; any theme that was created in any Office application can be used in Access as well. Before designing a custom theme, you should check to see if an appropriate theme has already been defined. This could save you some time by simply selecting that theme.

You can apply a theme to only a single form, leaving the others as they are. Perhaps you'll want to use a different theme for the main navigation form. Of maybe the data entry forms should have a different look and feel from the administrative forms.

To do that, right-click the theme that you want to apply as shown in Figure 11-11. Then click the *Apply Theme to This Object Only* link.

www.it-ebooks.info

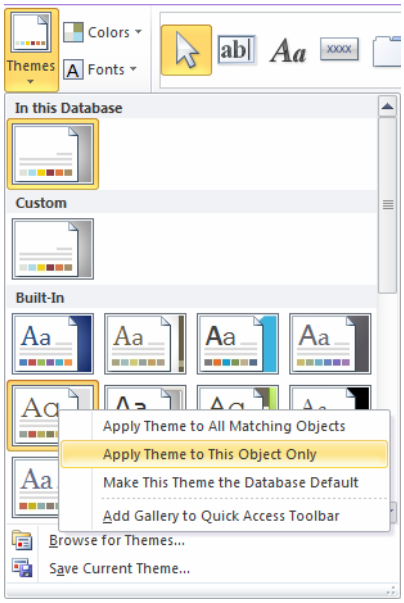


Figure 11-11. *Applying a theme to a single form*

The first link, *Apply Theme to All Matching Objects*, can be useful if you have already applied a different theme to a subset of forms. If you want to change all of these at once, this option will update the current form and all others that also use the same theme.

Applying an Office Theme

So far, you have defined the color and font schemes, which Access will use in its UI elements. Now you'll need to adjust your forms to use the appropriate colors. You have 10 colors to work with (ignoring the two that are used for hyperlinks). The forms created by Access will use the theme colors but may not use the ones you intended. For example, it may use Accent 1 and you wanted Accent 3 to be used instead.

Open the `Menu` form in the Layout View and look in the Property Sheet for the Tab Control object. The default color and font definitions will look like Figure 11-12.

Use Theme	Yes
Back Color	Background 1, Darker 15%
Border Style	Solid
Border Color	Text 2, Lighter 40%
Hover Color	Background 1
Pressed Color	Background 1
Hover Fore Color	Text 1, Lighter 25%
Pressed Fore Color	Text 1, Lighter 25%
Fore Color	Text 1, Lighter 25%
Font Name	Arial (Header)
Font Size	11
Font Weight	Normal
Font Underline	No
Font Italic	No

Figure 11-12. *The default color and font selections*

www.it-ebooks.info

Notice all the foreground colors use Text 1 and the background colors use Background 1. To achieve the desired effect, the Lighter or Darker attributes are used. However, the look you're trying to match uses a different approach. The main navigation area for the `www.apress.com` site is shown in Figure 11-13.

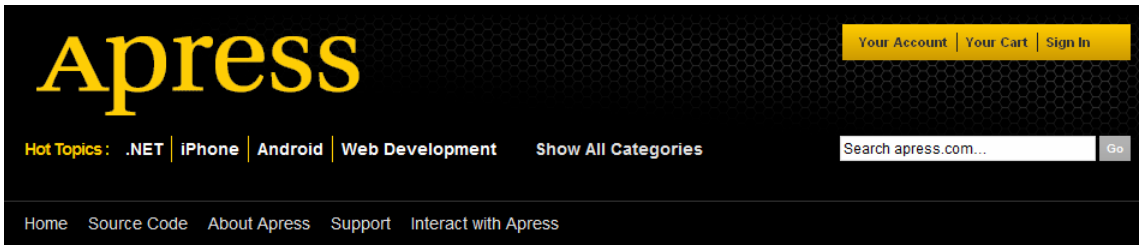


Figure 11-13. *Sample Apress menu*

The background is black and the menu options are in white. Actually, the menu text is a light gray but they turn white when the mouse is hovered over that option. For the following properties, click the ellipses to display the color picker and then select the standard Black color:

- Back Color
- Hover Color
- Pressed Color

■ **Tip** There are several ways that you can select the Black color. There is a Theme color that is Black, which is the primary text color. There is also a Standard color that you can use. This is probably just a matter of personal preference, but to use a Theme color that is intended as foreground text, as a background color seems to be violating the purpose of a Theme. However, by using the Standard color, the background will not change if someone modifies the Theme.

For the corresponding foreground colors (Hover Fore Color, Pressed Fore Color, and Fore Color) use the Text 2 theme color. For the Fore Color property, add the Darker 15% modifier.

The final Property Sheet will look like Figure 11-14.

Back Color	#000000
Border Style	Solid
Border Color	Text 2, Lighter 40%
Hover Color	#000000
Pressed Color	#000000
Hover Fore Color	Text 2
Pressed Fore Color	Text 2
Fore Color	Text 2, Darker 15%

Figure 11-14. *The updated color properties*

www.it-ebooks.info

■ **Tip** If you're not sure which color to use, click the ellipses next to the particular property that you're setting, which will display the color picker. If you hover the mouse over one of the theme colors, it will tell you the color name, as shown in Figure 11-15. Also, if you hover over one of the various shades that are displayed, it will indicate the corresponding Lighter or Darker attribute. You can click on one of these colors and the property will be updated with the corresponding value.

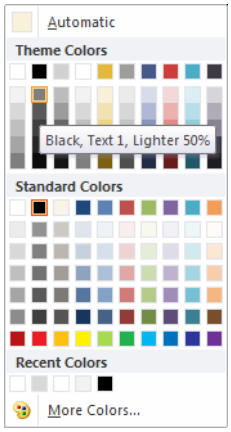


Figure 11-15. *Using the color picker*

The buttons on the form use the Accent 1 color, which is what you'll want but they use a lighter shade. By default, the background colors for buttons are:

- Back Color: Accent 1, Lighter 40%
- Hover Color: Accent 1, Lighter 60%
- Pressed Color: Accent 1, Darker 25%

Remove the Lighter 40% modifier on the Back Color property. And change the Hover Color property to use Lighter 25%. You'll need to make these changes to all eight buttons on this form.

Making Other Visual Adjustments

If you compare the Access form with the `apress.com` site, you'll notice a few visual differences. You'll make some minor adjustments to closer match the UI. The menu on the website uses links instead of tabs. However, by squaring off the tabs and removing the border, you can approximate the same effect.

1. In Access, select the Tab Control and set the Border Style property to Transparent.
2. From the Format tab of the ribbon, click the Change Shape button and click the Rectangle shape, as shown in Figure 11-16.

www.it-ebooks.info

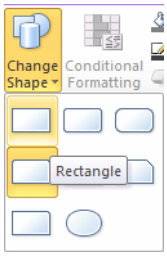


Figure 11-16. *Change the TabControl to use square corners.*

The final `Menu` from should look like Figure 11-17. If you compare this to the website sample shown in Figure 11-13, you can see that that it's a pretty close match.

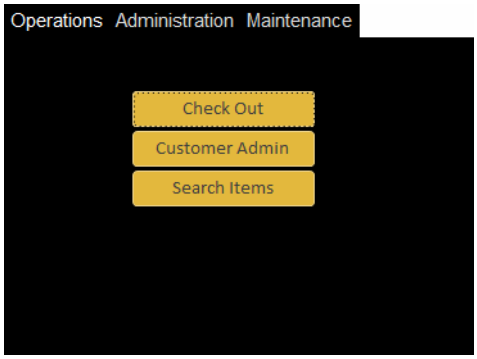


Figure 11-17. *The final* `Menu` *form*

Using Graphics

The other principle factor in branding an application is using familiar graphics. I mentioned the Pizza Hut logo at the beginning of the chapter. Most organizations have at least one widely known symbol that will give instant brand recognition. For Apress, the yellow and black banner is a key element.

Adding static images in Access 2010 is really easy. When you first use a graphic file it is added to the image gallery. Reusing it is as simple as selecting it from a dropdown list. I have created a few graphic files that you'll use for this project. If you haven't already, download my `Images` folder. The `Static` subfolder contains the graphics you'll need.

Adding a Banner Graphic

You'll add this black and yellow banner to the `Menu` form. You'll first need to create an empty cell to put the graphic in. Then you'll use the image gallery to add the images to the form.

1. Open the `Menu` form in the Layout View and select the TabControl. From the Arrange tab of the ribbon, click the Insert Above button. This will create an empty cell; expand its height to be about 1 inch.

www.it-ebooks.info

2. From the Design tab of the ribbon, click the Insert Image button, as shown in Figure 11-18. This will list the images that are currently in your image gallery; but since there are none, it will just have a *Browse* link.

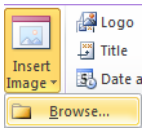


Figure 11-18. *Selecting the Browse link*

3. Click the *Browse* link, navigate to the `Images\Static` folder, and select the `AppressLogo.jpg` file.
4. Next click the empty cell. This will add the image to this cell. The Size Mode property defaults to Zoom. Change this to Clip. This graphic was made quite long to accommodate some of the wider forms. Rather than zooming out to fit the entire graphic, you just want to clip off whatever doesn't fit on the form.
5. Now you'll eliminate the white space between the banner and the Tab Control. In the Property Sheet, select the Detail object and the Format tab. For the Back Color property, select the standard Black color.

Open the `Menu` form using the Form View. The final design should look like Figure 11-19.



Figure 11-19. *The final* `Menu` *form design*

No go back to the Layout View and click the Insert Image button again, this time you'll see the logo that you just used as demonstrated in Figure 11-20. When you need to add this to other form, you can simply select it from this list.

www.it-ebooks.info

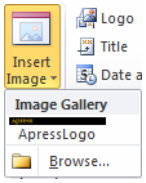


Figure 11-20. *The logo included in the image gallery*

If you need to modify an image that has been loaded in the gallery, right-click the image in the gallery and click the *Update* link, as shown in Figure 11-21.

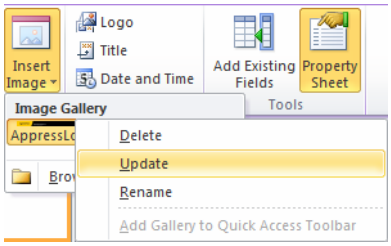


Figure 11-21. *Updating an existing gallery image*

Then browse to and select the modified file. The new version will be stored in the gallery, replacing the existing one. Also, any place that this image has been used will be updated automatically.

Using a Background Image

Another design element that you'll want to mimic is the use of a header graphic that has a curved top-right corner as shown in Figure 11-22. You'll add this effect to the `Category` form.



Figure 11-22. *Sample Apress heading*

1. Open the `Category` form in the Design View. In the Property Sheet, select the Form object.

www.it-ebooks.info

2. The Picture property is used to specify the background image for the form. Click the dropdown icon next to this property. Notice that the `ApressLogo` image is listed here, as demonstrated in Figure 11-23.

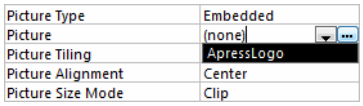


Figure 11-23*. Selecting the image gallery from the Property Sheet*

■ **Note** The Property Sheet is integrated into the image gallery. Any items that are already in the gallery are available in the dropdown list. As I'll show you, when you select a new image, it will be loaded into the gallery as well.

---

3. Change the Picture Type property to Shared. This will indicate that you want to use the image gallery.
4. Select the Picture property and click the ellipses and browse to and select the `ApressArc.jpg` file.
5. The image will be displayed in the middle of the form. Change the Picture Alignment property to Top Right.
6. Shrink the height of the Form Header so the background picture just fits in the header section. You'll probably have to first shrink the height of the Label control.
7. Select the Label control and set the Fore Color property to Text 2.

Notice that when you click the Insert Image button in the ribbon, the new file is now included in the list. Open the `Category` form using the Form View. The form should look like Figure 11-24.



Figure 11-24*. The modified* `Category` *form*

www.it-ebooks.info

The other forms will need similar modifications; I'll leave that for you to do on your own.

Summary

In this chapter you "branded" your `Library` application to mimic the `www.apress.com` website. Anyone familiar with that site will easily recognize your application as being from the same organization. While this is a somewhat fictional exercise, the concept is important. End users tend to respond first to how a system looks before they notice how well it works. Developers need to concentrate effort on the UI as well as the functionality of the systems they build.

You created a custom theme, following the same color and fonts schemes found on the website. You then modified the forms to use the appropriate colors in the right places. You added a few graphics to the forms making use of the image gallery. With some simple techniques, the effective use of color, fonts, and graphics turned an average looking application into one with some "wow" factor. More important, the instant brand recognition will go a long way in providing acceptance and comfort.

In the next chapter you'll create reports for the `Library` application. As you'll see, reports are essentially read-only forms that are designed for print output.

www.it-ebooks.info

CHAPTER 12

Reports

Most database solutions will need to provide reports, which can be difficult and often tedious. With Access, however, even complex reports are easy to build. There are several ways to create a report that provides different levels of customizations, from creating a report with a single click to starting with a blank sheet and designing from scratch. The report designer includes a powerful facility for grouping and sorting data. Best of all, reports are essentially read-only forms, so once you have mastered building forms, you can easily create great reports as well.

In this chapter, I'll explain some of the basic concepts and then show you how to create several reports using the different options available. I'll demonstrate how to the use the wizards to quickly create useful reports. You'll also build a report yourself by starting with a blank design. At the end of the chapter, I'll show you some advanced features, such as generating a PDF file and outputting a report automatically each day at a specified time.

Exploring Access Reports

Reports in Access are implemented much like forms. Each report uses a single data source, which can be a table, query, or SQL statement. You use data-bound controls to extract a field from the data source, format it, and display it in an appropriate location on the report. The Property Sheet will display many of the same properties that you used for forms. You can also add unbound controls and use macros or VBA code to manipulate their content.

One thing to keep in mind is that reports are read-only. You can set the Enable and Locked properties, but when opened in the Report View they cannot be edited. You can modify them in code, however.

Understanding Report Sections

When creating forms, you had multiple sections to design, including Detail and Form Header. Reports work the same way, except that there are more sections to work with. In addition to a Report Header and Report Footer sections, you'll also have page header footer section as well as group headers and footers as illustrated in Figure 12-1.

www.it-ebooks.info

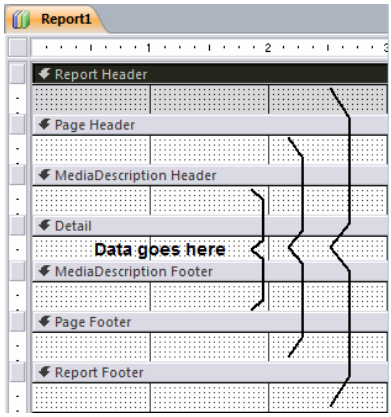


Figure 12-1. *The report sections*

To help you visualize how these sections will be used, Figure 12-2 shows a portion of a report that you will create later in the chapter. I have annotated the various header and footer sections.

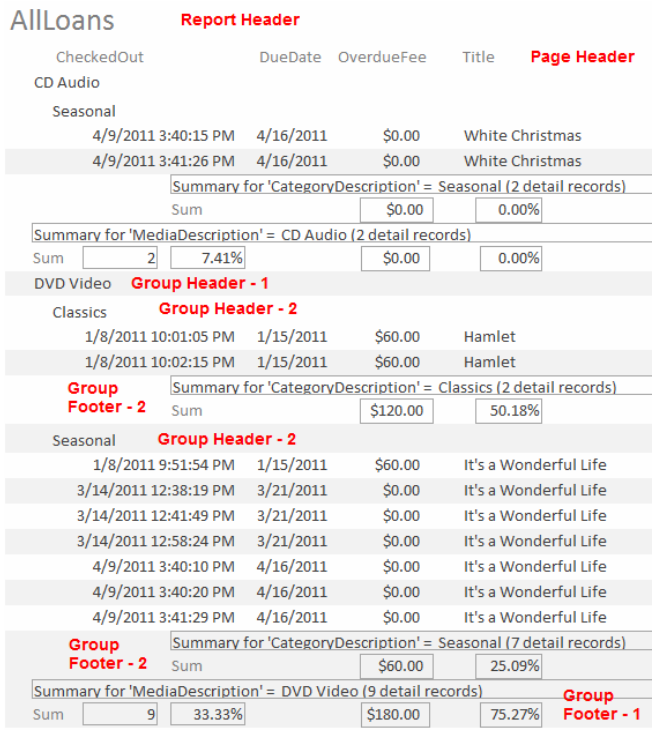


Figure 12-2. *A sample report with two grouping levels*

www.it-ebooks.info

■ **Note** Forms also support Page Header and Page Footer sections, although these are seldom used. Group header and footer sections, however, are only found on reports.

As with forms, the Detail section will be repeated for each record returned by the data source.

Using Page Sections

In addition to the Report Header and Report Footer section, you will also use the Page Header and Page Footer section. By default, the first page of a report will display the Report Header followed by the Page Header and the Page Footer will be included at the bottom of the page. The last page will start with the Page Header and end with the Page Footer, followed by the Report Footer. All intermediate pages will have only the Page Header and Page Footer sections. Figure 12-3 demonstrates how these sections are arranged.

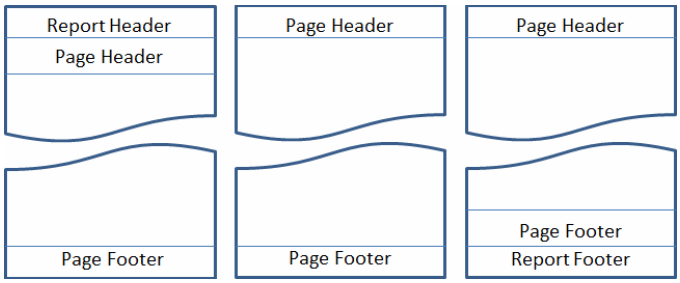


Figure 12-3*. Placement of Report and Page sections*

You should keep these rules in mind when designing your report. On a multiple page report, you'll need to decide what you want on the first page only (or last page only) and what you want to display on every page. It might help to image what the report will look like when printed on paper. Column headings, for example, should probably be on every page, so they should be in the Page Header rather than the Report Header.

Sometimes having both the Report Header and Page header on the same page can be awkward. For these situations, Access allows you to suppress the Page Header (or Page Footer) on the first or last page. If you select this option, there will be a Report Header on the first page (without a Page Header) and a Page Header on subsequent pages. You'll need to be sure to include the appropriate controls, such as column headings, on both sections.

Using Group Sections

One of the really great features of Access reports is the ability to group records based on a field in the data source. For example, if you choose to group by `MediaDescription` as I did in Figure 12-1, all records with the same media type are grouped together. So all the CDs will be listed, and then all the DVDs, and so on. With each group, header and footer sections can be used.

www.it-ebooks.info

A group header precedes the detail records that fall into that group. You can use this to display a group label to introduce the group. Likewise, a group footer will follow the last detail record of that group. This is a great place to add a subtotal or other group summary information. These sections are optional; if no controls are placed in the section it will be ignored.

You can nest groups up to ten levels deep. You could, for example, further divide each media type by the category. So within the CD group, you would have subgroups for Children's, Classics, and so on. Each subgroup can also have its own header and footer section.

Options for Creating Reports

Access provides several ways to create a new report. In the Create tab of the ribbon, the Reports group includes a command button for each option as shown in Figure 12-4. I will explain each of these methods throughout this chapter.

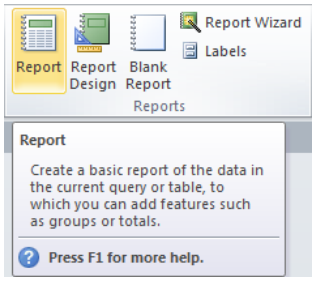


Figure 12-4. *Options for creating a new report*

The easiest way to create a report is to select the table or query that should be used as the data source and then click the Report button. On the opposite end of the spectrum, the Report Design button will create a blank report using the Design View where you can build a report from scratch and have complete control of all aspects of the report.

The Blank Report button works the same way except the report is opened using the Layout View. The Layout View for reports is very similar to the Layout View you used for designing forms. You can add fields from the data source and rearrange them. One of the nice things about the Layout View is that it displays actual data so you can get a better idea of how the report will look.

Access provides two wizards. The Report Wizard enables you to design a complex report, including multiple grouping levels, sorting, and layout options. This will generate a good starting point for most of your reporting needs. The Label Wizard allows you to define a report that will print on label stock. It comes with a large selection of built-in templates from many popular label vendors; but you can easily define your own as well.

Creating a Simple Report

For your first report, you'll generate a report of media types. Select the `Media` table in the Navigation pane. Then click the Report button in the Create tab of the ribbon. After a few seconds it will open the `Media` report in the Layout View as demonstrated in Figure 12-5.

www.it-ebooks.info

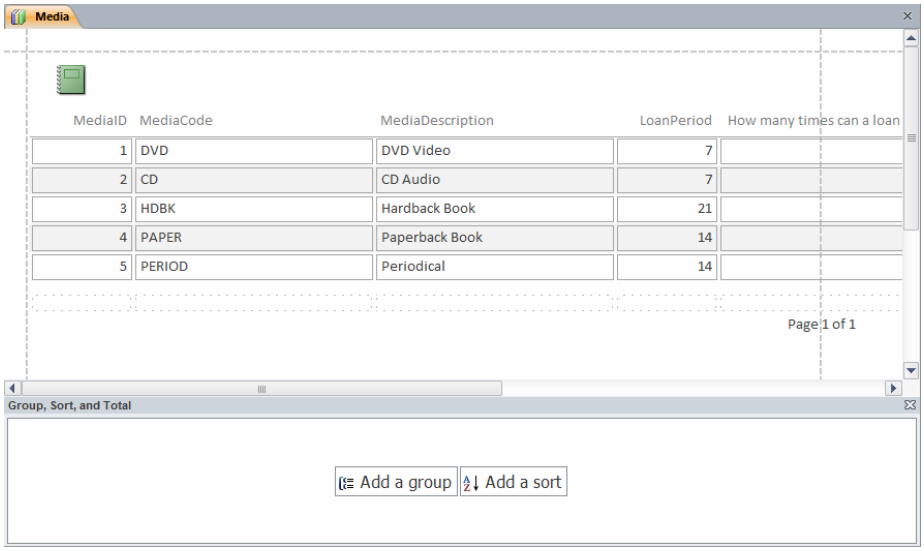


***Figure 12-5**. The initial `Media` report in the Layout View*

The template generates an `Auto_Header0` control in the Report Header that displays the title of the report. By default, this uses the Text 2 color. In our custom theme, this is White, just like Background 1, which is the background color for the report. This combination makes the title invisible. To fix this, select the `Auto_Header0` control and change the Fore Color property to Text 1. Make the same change to the `Auto_Date` and `Auto_Time` controls, which are also in the Report Header.

Modifying the Page Setup

Notice the dotted lines, which show you where the page breaks are. These are calculated based on the current page configuration. Select the Page Setup tab of the ribbon. This tab provides several configuration options, as shown in Figure 12-6.

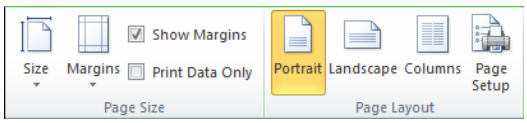


***Figure 12-6**. The Page Setup ribbon tab*

The printable area is defined by three factors:

- Paper size
- Margins
- Orientation

www.it-ebooks.info

The Size button allows you to select from a list of standard paper sizes, such as Letter, Legal, and A4.

■ **Tip** If you need to define a custom paper size, the process is a little more convoluted than you might expect. Paper sizes are actually defined by the operating system's print server. If you're using Windows 7, go to the Devices and Printers option from the Start Menu. Then click the *Print server properties* link at the top of the dialog box. This will display the Print Server Properties dialog box shown in Figure 12-7. Select the "Create a new form" check box, fill in the form details, and then click the Save Form button. Your custom paper size should now be included in the available options when you click the Size button in Access.

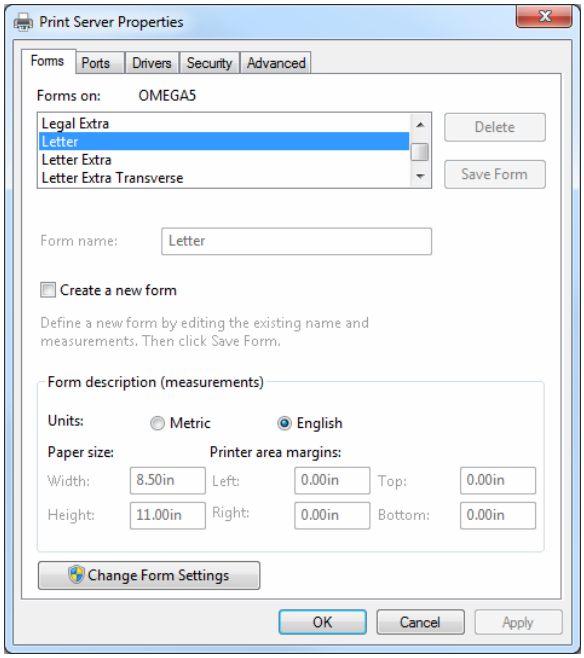


Figure 12-7. *The Print Server Properties dialog box*

You can use the Margins button to select a standard margin configuration or to define custom margin, as shown in Figure 12-8.

www.it-ebooks.info

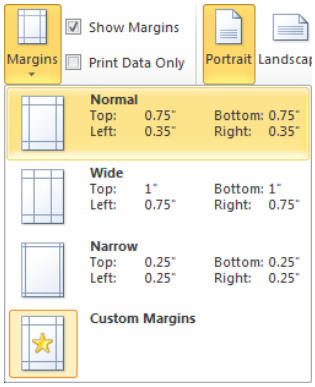


Figure 12-8. *Selecting the margin options*

The orientation is specified by selecting either the Portrait or Landscape buttons in the ribbon.

■ **Tip** You could also click the Page Setup button, which will display the Page setup dialog box. You can configure all the page options (size, margins, and orientation) from here.

Modifying the Report Layout

The column widths need to be adjusted so the report will fit on a single page. While using the Layout View, you can resize columns on a report just like you did with forms.

1. First remove the `MediaID` column by right-clicking the label or any of the `MediaID` values and selecting the *Delete Column* link.
2. Change the column heading for the `RenewalsAllowed` column to **AllowedRenewals**.
3. Resize each of the columns so the data just fits inside them.
4. Drag the page number control to the first column of the last row (the same row the grand total is on).
5. Switch to the Design View and then drag the right edge of the report as far to the left as it will go to remove all unused space.

The design should look like Figure 12-9.

www.it-ebooks.info

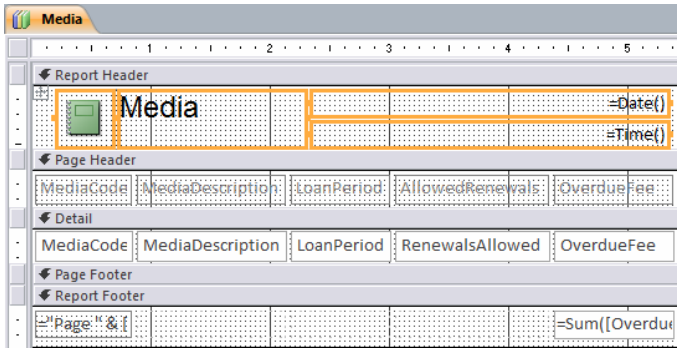


Figure 12-9*. The Design View of the* `Media` *report*

6. It doesn't make sense in this context to sum the `OverdueFee` columns. Select this control in the Report Footer and delete it.

Switch to the Report View to see the final report. The report should look like Figure 12-10.

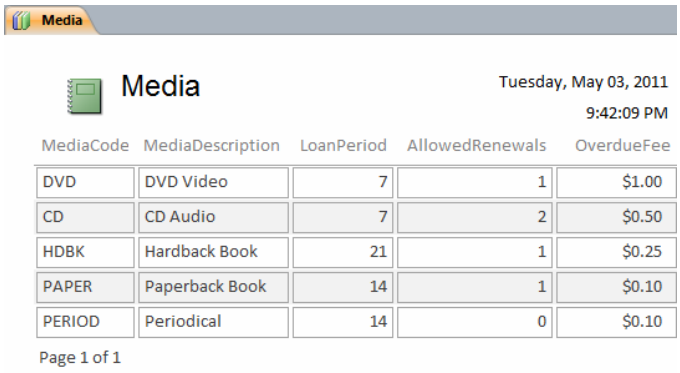


Figure 12-10*. The final Media report design*

The Report View is used to display the report on the screen. To see how it will look when printed, select the Print Preview, as shown in Figure 12-11.

www.it-ebooks.info

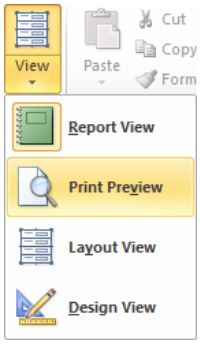


Figure 12-11. *Selecting the Print Preview view*

Save the report and enter the name **Media** when prompted.

Creating the AllLoans Report

You'll now create a more complex report and use the Report Wizard to set up the initial design. This report will use the `AllLoans` query that you created in Chapter 4, and will use multiple grouping levels to demonstrate how to configure nested groups. I showed you a sample of this report in Figure 12-2.

Using the Report Wizard

From the Create tab of the ribbon, click the Report Wizard button. This will launch the Report Wizard that will take you through a series of dialog boxes that you'll use to configure the report.

Selecting the Data Source

The first dialog box allows you to specify the table or query that will be used as the source of data for this report.

1. Select the `AllLoans` query. The list of available fields will be displayed on the left side. Double-click a field to move it to the list of selected fields on the right side.
2. Select the following fields to be included in the report, as shown in Figure 12-12:
   - `CheckedOut`
   - `DueDate`
   - `OverdueFee`
   - `Title`
   - `Author`

www.it-ebooks.info

- CategoryDescription
- MediaDescription
- Overdue

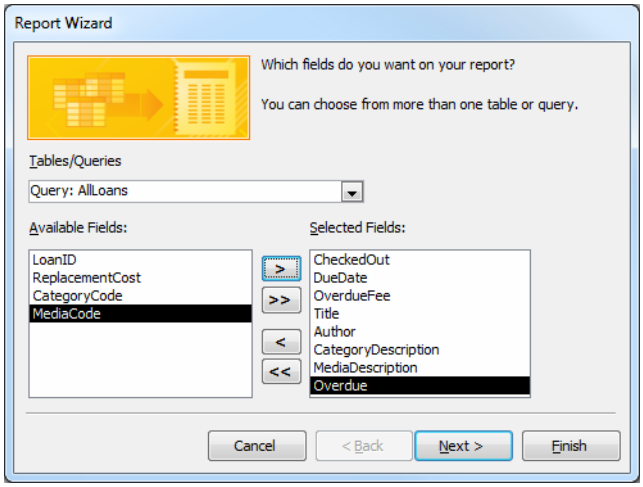


Figure 12-12. *Selecting the table or query and fields*

Grouping Records

Access looks at the table relationships and tries to provide some default grouping schemes. The second dialog box, shown in Figure 12-13, lists the options that are available.

www.it-ebooks.info

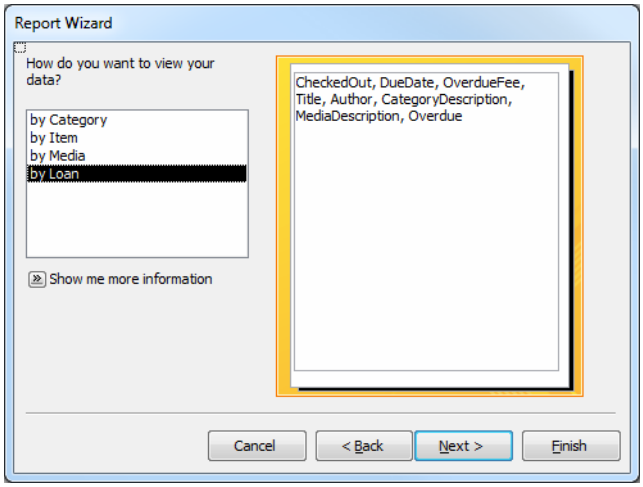


Figure 12-13. *Selecting the initial group option*

The default scheme, which is called "by Loan," does not provide any grouping. The `Loan` table is the main table in the `AllLoans` query. The other tables – `InventoryItem`, `Item`, `Media`, and `Category` – are all joined with a many-to-one relationship. (Refer to Chapter 4 more information on join properties.) In other words, these tables were joined to provide additional columns such as `Author` and `MediaDescription` but will not produce any new rows.

The design of the `AllLoans` query, shown in Figure 12-14, illustrates these relationships. As you can see the `Loan` record is at the lowest level.

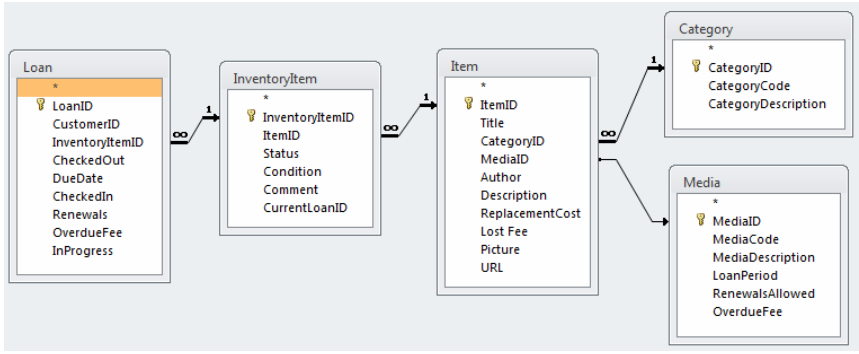


Figure 12-14. *The design of the* `AllLoans` *query*

www.it-ebooks.info

PROPOSED GROUPING OPTIONS

You will set up the grouping manually since none of the default schemes satisfies the requirement of this report. However, I wanted you to see the way these work. Select the "by Media" option and the option should look like Figure 12-15.

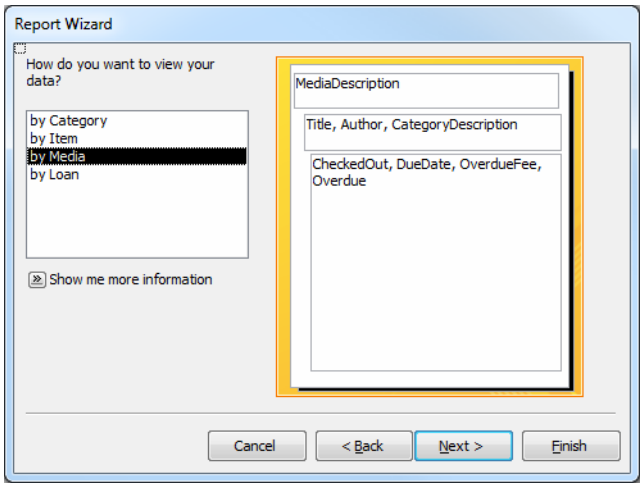


Figure 12-15. *Available grouping options*

If you refer to the query design in Figure 12-14 you can see the logic behind this grouping. If you select `Media` as the top-level group by following the join relationship, the next level in the hierarchy would be the `Item` table. So the top level summarizes each media type and the second level summaries each item within that media type.

The next level would be the `InventoryItem` table. However, there are no fields from this table used in the report so this level is skipped. The final level is the `Loan` table. The Report Wizard places each field in the appropriate grouping level to help you see the details that are available for each group.

Try selecting the other options and see that the grouping schemes follow this same logic. You can probably see why when selecting the `Loan` table there is no way to add additional groups since it's already at the lowest level.

1. Select the "by Loan" option and click the Next button. By selecting this option, no groups were created for you. However, you can easily add your own in the third dialog.
2. Select the `MediaDescription` column and then click the ">" button.
3. Select the `CategoryDescription` column and click the ">" button again.

The dialog box should look like Figure 12-16.

www.it-ebooks.info

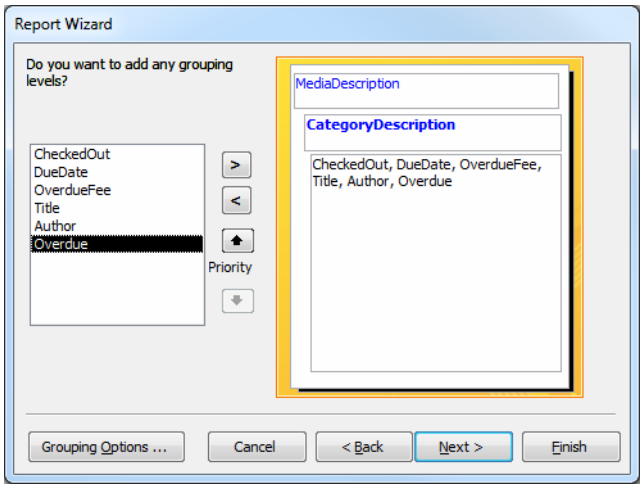


Figure 12-16. *Adding manual grouping levels*

New levels are added by selecting the field to be grouped by and clicking the ">" button. You can remove a group by selecting it on the right side of the dialog box and clicking the "<" button. If you want to change the order the fields are grouped, select the group on the right and use the up and down arrows.

The Grouping Options button will display the dialog box shown in Figure 12-17. This allows you to group by the entire field value or just by the first few characters. A good example of this is when you're grouping by a name. By selecting just the first letter, you'll have a group for A,B, and so on.

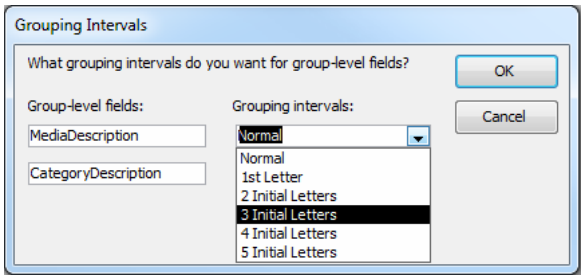


Figure 12-17. *Specifying how to group field values*

■ **Tip** If you want to group by some other portion of the field, such as the third character only, you can create a calculated column to the query. (See Chapter 4 for details on how to do that.) Then group by the calculated column. With this approach you can group by just about any scheme imaginable.

---

www.it-ebooks.info

Sorting and Summarizing the Details

The fourth dialog box allows you to specify how the records in the Detail section should be sorted. This does not specify how the groups themselves are sorted. You can adjust than in the Layout or Design View once the report has been created.

1. Select the `CheckedOut` field, as shown in Figure 12-18.

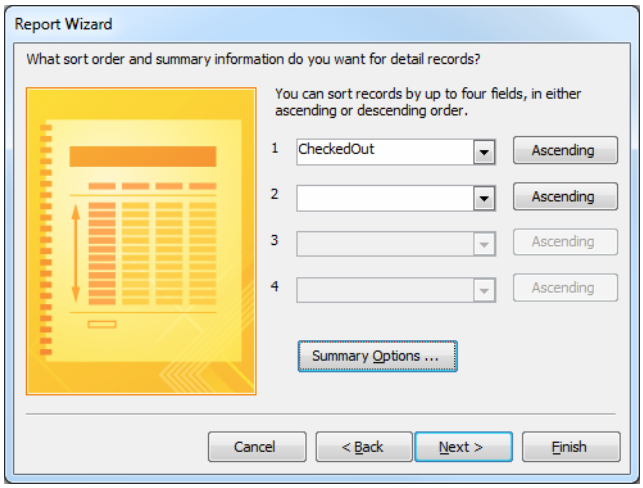


Figure 12-18. *Selecting the sort options*

2. Click the Summary Options button, which will display the Summary Options dialog box. This will list all the numeric fields in the Detail section that you might want to provide a summary of.

3. Select the `Sum` function for the `OverdueFee` field, as shown in Figure 12-19.

www.it-ebooks.info

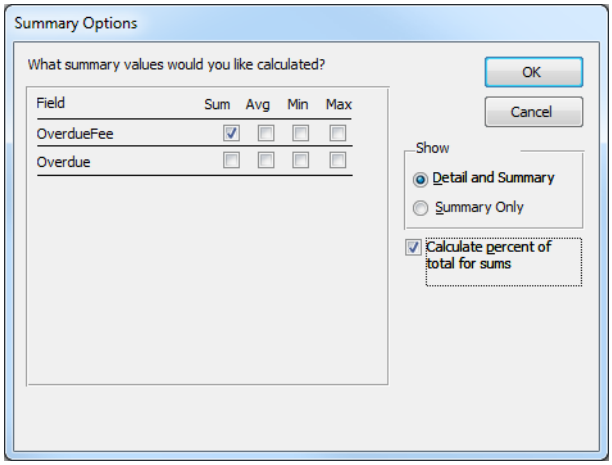


Figure 12-19. Enabling group sumaries

■ **Note** This dialog box also provides the option of including both detail and summary information, or just the summary. If you select the Summary Only option, instead of listing each `Loan` record, it will just provide the summary for each media type and category.

4. Select the "Calculate percent of total for sums" check box. This will include this information in the group summary.

Selecting the Format

In the fifth dialog box you can select from one of three predefined layouts. The options are called Stepped, Block, and Outline. Select each of these options and the image will change to show how that layout is defined. In this dialog box you can also choose the page orientation and force the fields to fit within one page width.

1. Leave all of the default values, as shown in Figure 12-20.

www.it-ebooks.info

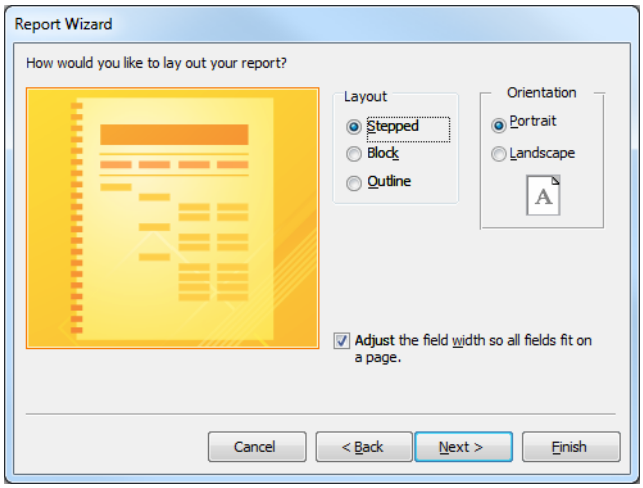


***Figure 12-20**. Selecting the page layout template*

2. In the final dialog box, enter **AllLoans** for the report name, as shown in Figure 12-21, and click the Finish button.

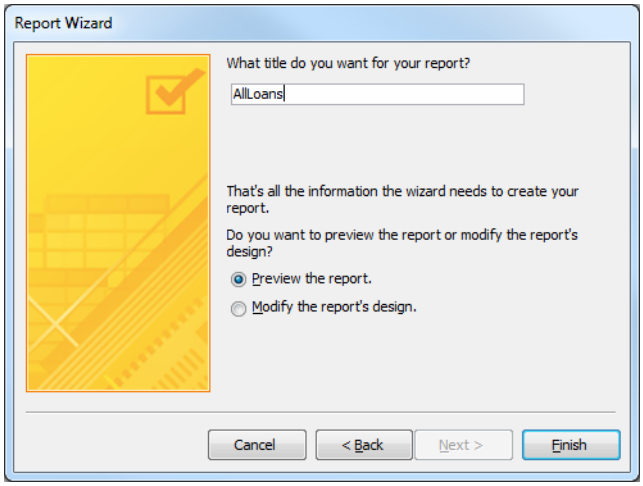


***Figure 12-21**. Entering the report name*

The `AllLoans` report should be opened in Print Preview and should be similar to Figure 12-22.

www.it-ebooks.info

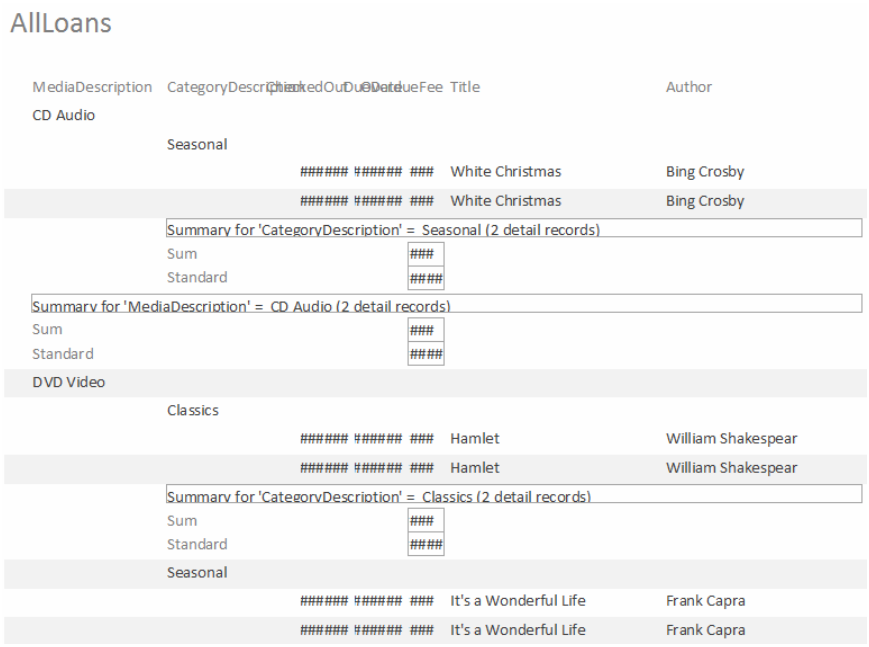


Figure 12-22. *The initial* `AllLoans` *report in Print Preview*

Exploring the Design View

Click the Close Print Preview button in the ribbon. This will close the Print Preview window and open the report using the Design View. The Design View should look similar to Figure 12-23.

www.it-ebooks.info

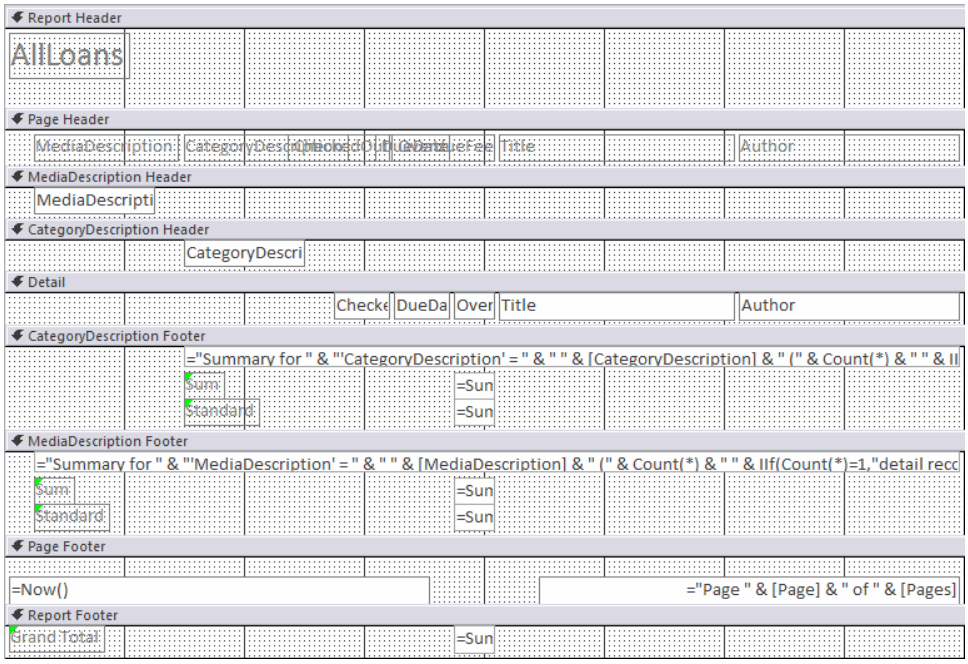


Figure 12-23*. The initial Design View*

The Report Wizard didn't do anything that you couldn't do yourself by adding sections and placing the appropriate controls on the report. While the Design View is open, take a look at the report design. Notice the sections which are organized like I described earlier. The Report Wizard created the following sections:

- Report Header
- Page Header
- MediaDescription Header
- CategoryDescription Header
- Detail
- CategoryDescription Footer
- MediaDescription Footer
- Page Footer
- Report Footer

In the Detail section, there are data-bound controls for displaying the actual report data. The `MediaDescription` and `CategoryDescription` fields are included in their respective header sections. There are `Label` controls in the Page Header section, which provide the column headings. There are also a

www.it-ebooks.info

number of unbound controls that provide details such as current date/time, page number, and various subtotals. I will explain these later in this chapter.

Using the Layout View

The Report Wizard does a good job creating the structure for your report but as you can see, some formatting will be required. The Layout View works well for that purpose.

1. Switch to the Layout View, which is shown in Figure 12-24. You told the Report Wizard to fit all the controls in a single page width. The fields were sized to fit on a page but most of the fields are too small to display the data in them. The hash symbol (#) is displayed inside a control to indicate that there is not enough room to format the data correctly. One easy way to gain some room is to change the indentation of the Category Description and Detail sections to be much smaller than they currently are.

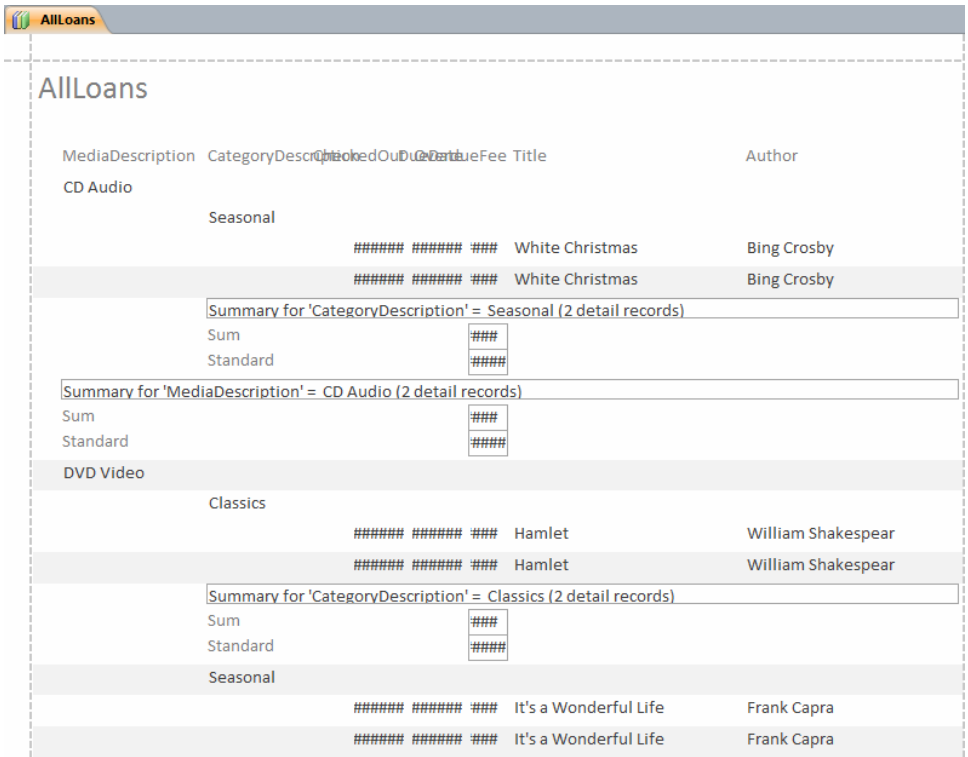


Figure 12-24. *The initial Layout View*

www.it-ebooks.info

■ **Tip** The fields were generated this way so the column heading could be displayed on a single row. Even though the data-bound controls for the MediaDescription Header, CategoryDescription Header, and Detail sections are on different rows, their associated column headings are on a single row in the Page Header. For this report, you can simply delete the `MediaDescription` and `CategoryDescription` column headings as the data should be self-explanatory. However, if you needed to keep all of the column headings, you could put some on a second row in the Page Header.

---

2. Delete the column heading controls for the `MediaDescription` and `CategoryDescription` fields.
3. Select the `CategoryDescription` control and drag it to the left so it is indented from the `MediaDescription` control about a quarter of an inch.
4. Now you'll need to spread out and enlarge the controls in the Detail section. To keep then aligned with the column headings, select the control, hold down the Shift key, and select the associated column heading. Then you can drag them both simultaneously.

■ **Tip** With both the control and its associated label selected, you can use the left and right arrow keys to drag the controls to the appropriate location.

---

5. In the CategoryDescription Footer section there are two subtotal controls called Sum and Standard. The Sum control displays the subtotal of `OverdueFee` for that group. The Standard control displays this as a percentage of the grand total. Move the Standard control to be on the same row as the Sum control. Expand the width of both controls so the data can be displayed.
6. Delete the label for the Standard control.
7. Repeat this for the MediaDescription Footer.

The final format should look like Figure 12-25.

www.it-ebooks.info

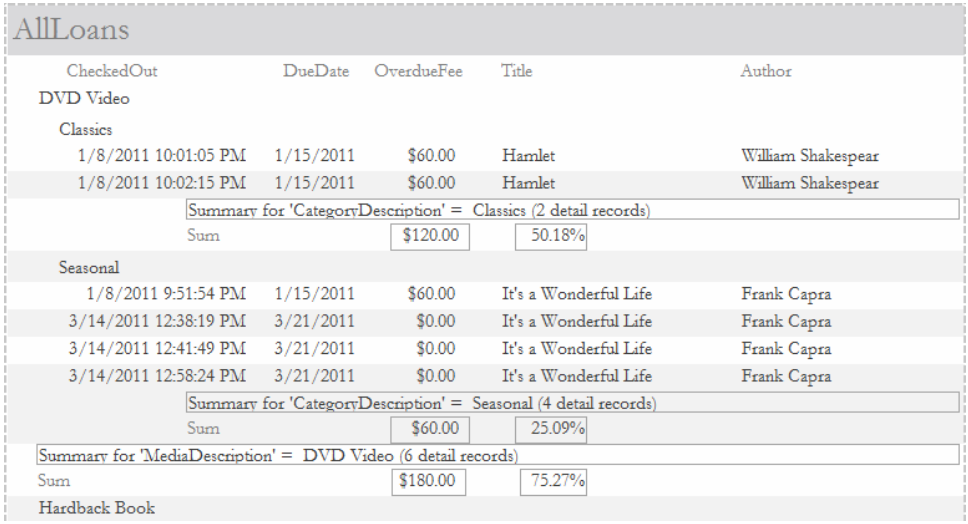


Figure 12-25. *The final layout of the* `AllLoans` *report*

Configuring Grouping and Sorting

The Report Wizard generated the group sections for you but you can create or modify groups yourself from the Design View. This also gives you more options that are not available from the Report Wizard.

1. Switch to the Design View. Right-click the report and select the *Sorting and Grouping* link. This will display the "Group, Sort, and Total" pane at the bottom of the Access application, which is shown in Figure 12-26. You can see the `MediaDescription` group followed by the `CategoryDescription` group and then the Details are sorted by the `CheckedOut` field. These were generated by the Report Wizard.

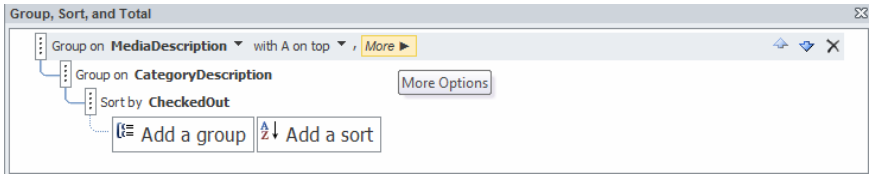


Figure 12-26. *The Group, Sort, and Total pane*

2. Click the *More* link on the first group to display additional options. The group will expand to show more links, as shown in Figure 12-27.

www.it-ebooks.info

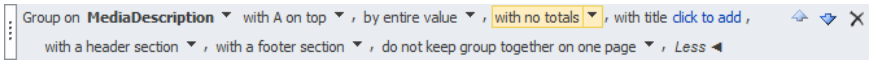


Figure 12-27. *Displaying the additional group options*

The current group configuration is described with a collection of phrases. Each phrase is a link that will open up a dropdown list where the available options are displayed. These phrases include:

- *Group on <field name>*: The dropdown list allows you to select a field that is included in the report.
- *with A on top*: The group instances are sorted by the group field – in this case, `MediaDescription`. This option allows you to choose if you want to sort in ascending or descending order.
- *by entire value*: This option allows you specify if the records are grouped by the entire field name or just the first few characters.
- *with no totals*: This option allows you to control how the records should be summarized.
- *with title click to add*: Allows you to specify a title that will be displayed with this group.
- *with a header section*: Use this to add or remove the header section.
- *with a footer section*: Use this to add or remove the footer section.
- *do not keep group together on one page*: This provide options for how page breaks are used within a group. You can change this to require that the header, details, and footer are all on the same page. You can also specify that just the header and the first details stay on the same page.

3. Click the dropdown icon for the *with no totals* phrase. Select the `LoanID` field and choose Count Records. Select the "Show group totals as % of Grand Total" and "Show subtotal in group footer" check boxes, as shown in Figure 12-28. This will add two unbound controls to the MediaDescription Footer to show the number of loans and the percent of the grand total. Both of these controls are on top of existing controls.

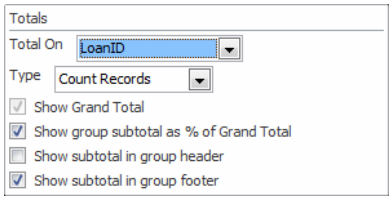


Figure 12-28. *Configuring the group totals*

www.it-ebooks.info

4. Drag the bottom control to an empty space to the right. Drag the top one to the right of the bottom control. The MediaDescription Footer should look like Figure 12-29.

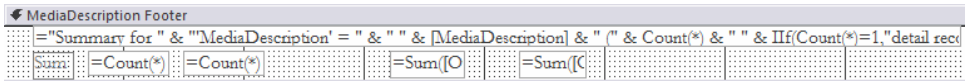


Figure 12-29. *The layout of the MediaDescription Footer*

5. Just like you did with the `Media` report, you'll need to fix some of the Report Header controls. Select the `Auto_Date` and `Auto_Time` controls and change the Fore Color property to use Text 1.

6. Save the report and switch to the Report View.

The final report should look like Figure 12-30.

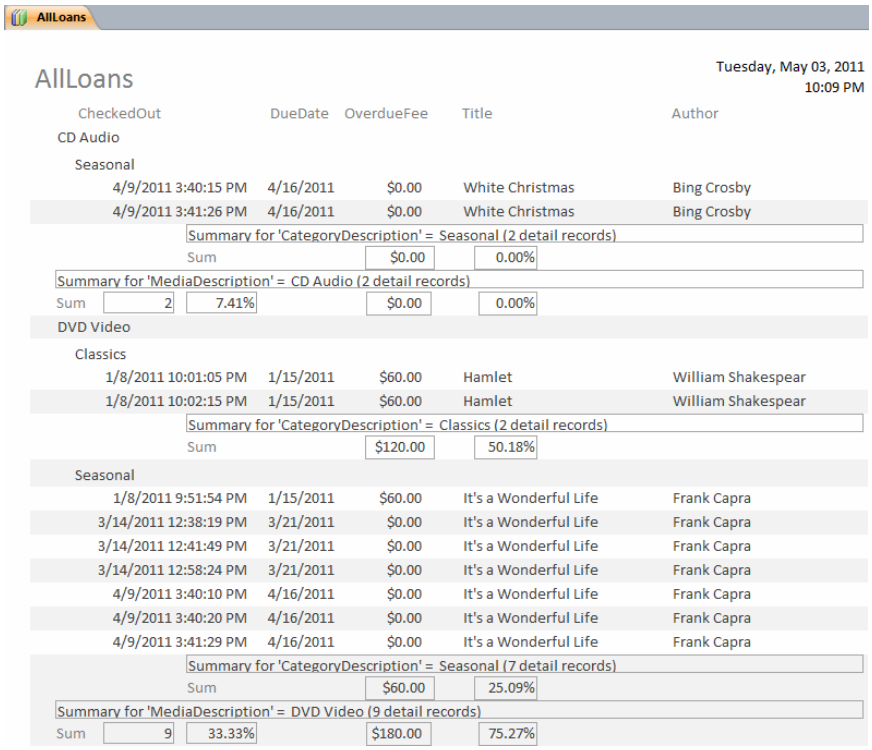


Figure 12-30. *The final design of the* `AllLoans` *report*

www.it-ebooks.info

Using Unbound Controls

As I mentioned earlier, in addition to data-bound controls that display your report data, you can use unbound controls to add other details. You can learn a lot by just looking at the controls that were generated by the report wizard. Switch to the Design View.

Formatting Page Numbers

There is a control on the right-hand side of the Page Footer section. Select this and then, in the Data tab of the Property Sheet, look at the Control Source property. The formula for this control is:

```
="Page " & [Page] & " of " & [Pages]
```

The keywords `Page` and `Pages` are used to build a string in the form of "Page 1 of 2." You can use these keywords to format the page number however you want. You can also move this control to another location in this section or move it to the Page Header.

Access provides a utility for making this even easier.

1. Delete this control from the Page Footer.
2. In the Design tab of the ribbon, click the Page Numbers button, as shown in Figure 12-31.

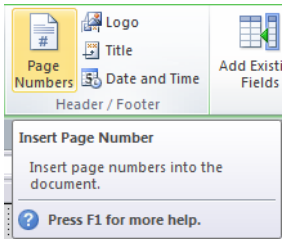


Figure 12-31*. Selecting the Page Numbers utility*

3. In the Page Numbers dialog box, select the "Page N of M" option, select the Footer section, and set the Alignment to Right, as shown in Figure 12-32.

www.it-ebooks.info

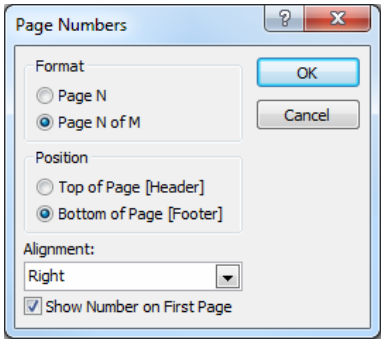


Figure 12-32. *Configuring the page numbers*

This will add the control back like it was.

Displaying the Current Date and Time

Notice that the ribbon also has a "Date and Time" button. This will add a date and/or time control to the Report Header. Click this button and the dialog box shown in Figure 12-33 will be displayed.

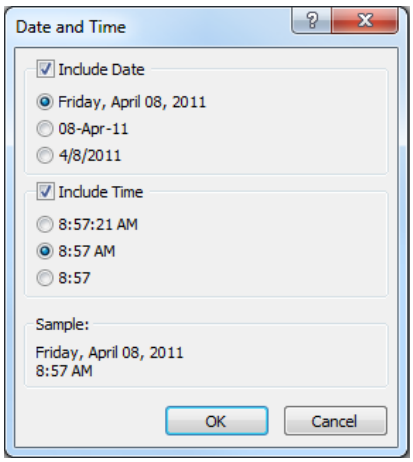


Figure 12-33. *The Date and Time dialog box*

This will create separate controls for the date and time depending on the options you select. Cancel this dialog box, because your report already has these controls.

You can easily add your own controls. Just add a TextBox control and enter one of the following formulas for the Control Source property.

- `=Date()`: Displays the current date (no time)

www.it-ebooks.info

- `=Time()`: Displays the current time (no date)
- `=Now()`: Displays the current date and time

Then select the desired format from the Format tab of the Property Sheet.

Using Totals and Subtotals

You can add unbound controls anywhere on the report and specify a formula for its content in the Control Source property. The page number and date/time controls are specific examples of this. Another use of this is for displaying totals and subtotals.

Access provides several built-in functions for aggregating data. The Expression Builder is a useful tool for editing formulas.

1. Select the "Sum Of OverdueFee" unbound control in the CategoryDescription Footer.
2. In the Data tab of the Property Sheet, select the Control Source property and clear the value of this property. You will now re-create it manually.
3. Click the ellipses, which will display the Expression Builder dialog box.
4. For the Expression Elements, select the Built-In Functions and then select the SQL Aggregate category. The available functions will be listed as shown in Figure 12-34. If you select one of the functions, the text at the bottom of the dialog box will contain a description of the function.

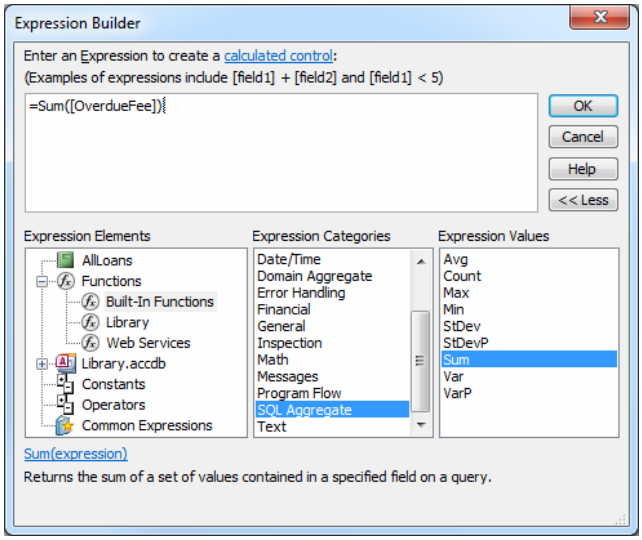


Figure 12-34. *Using the Expression Builder*

5. Double-click the `Sum` function, which will add it to the expression.

www.it-ebooks.info

6. Select the "<<expression>>" place holder and start typing **OverdueFee**. IntelliSense will fill in the remainder of the column name.

7. Click the OK button to save the expression and update the Control Source property.

You may have noticed that the exact same function is used in the CategoryDescription Footer, MediaDescription Footer, and Report Footer sections. Access knows, based on what section the control is in, the appropriate records to be included in the aggregate function.

In the Report Wizard you specified to also show the summary as a percentage of the total. If you select that the control, you'll see its Control Source is specified as:

```
=Sum([OverdueFee])/([OverdueFee Grand Total Sum])
```

This simply takes the subtotal and divides it by the grand total. The Format property specifies that the value should be displayed as a percent. To get the grand total, the formula references the name of the control that contains the grand total. The control in the Report Footer is named "OverdueFee Grand Total Sum."

■ **Tip** The subtotal is expressed as a percentage of the grand total. You could just as easily present this as the percentage of the total for the media type. To do that, just divide the subtotal by `[Sum Of OverdueFee1]`, which is the name of the control in the MediaDescription Footer.

Both group footers contain an unbound control with a formula like this:

```
="Summary for " & "'CategoryDescription' = " & " " & [CategoryDescription] &
" (" & Count(*) & " " & IIf(Count(*)=1,"detail record","detail records") & ")"
```

This formats a summary line in each footer like this:

```
Summary for 'CategoryDescription' =  Seasonal (4 detail records)
```

This is a good example of the kind of logic you can add to unbound controls. You can combine field values, summary calculations, and conditional logic. Notice this code uses the `IIf` function to conditionally include the singular or plural versions.

Fixing the Percentage Formula

The formula generated by the Report Wizard does not handle the case where the grand total is 0. When that happens, the current formula gets a divide-by-zero error. You'll fix that now by using the `IIf` method to only perform the division if the grand total is not zero.

1. Select the "Standard Of OverdueFee" control in the CategoryDescription Footer. In the Property Sheet select the Control Source property and click the ellipses to open the Expression Builder.

2. Enter the following code for the formula:

www.it-ebooks.info

```
=IIf([OverdueFee Grand Total Sum] = 0, 0,
      Sum([OverdueFee])/([OverdueFee Grand Total Sum]))
```

3. Select the "Standard Of OverdueFee1" control in the MediaDescription Footer and enter the same formula for the Control Source property.

Creating a CheckOut Report

In Chapter 7, you created a `CheckOut` form that is used to record when items are loaned to a customer. Now you'll modify this form to printout a receipt that lists the items that were checked out. You'll start with a blank report and create this report from scratch. This will also demonstrate using a subform on a report.

1. In the Create tab of the ribbon, click the Report Design button. This will create a blank report and open it using the Design view. By default the report has a large detail section as well as a Page Header and Page Footer.
2. This report is designed to be printed like a cash register receipt so there will not be pages per se. Right-click the report and select the *Page Header/Footer* link to remove these sections.
3. Right-click the report again and select the *Report Header/Footer* link to add these sections.
4. Resize the Report Header to be about one and a half inch and resize the Detail section to be about half an inch.

Adding the Data-Bound Controls

This report will use the `LoanDetail` query, which is the same one used by the `CheckOut` form. When the `CheckOut` form runs this report it will supply a filter to limit the details to the items being checked out. There will be no filter while designing and testing the report so all `Loan` records will be included.

1. In the Property Sheet, select the Report object and the Data tab. For the Record Source property select the `LoanDetail` query.

■ **Tip** If there is too much data in the `Loan` table, you can define a filter such as `[CustomerID] = CLng(37)` to restrict the data to a single customer. Set this in the Filter property of the report. When the report is run from the `CheckOut` form you will add code to dynamically supply a filter so this default filter will be overridden.

---

2. Click the Add Existing Fields button in the Design tab of the ribbon.
3. Drag the `CustomerID` field to the Report Header. Set the Visible property to No. This control will be used for linking a subreport and does not need to be displayed.

www.it-ebooks.info

4. Drag the following fields to the Detail section:
    - `DueDate`
    - `Title`
    - `MediaDescription`
    - `Author`

Because this report will be designed to fit on a cash register receipt you'll keep the data as concise as possible.

5. Delete all of the associated label controls, including the label for `CustomerID`.
6. Change the Text Align property of the `DueDate` field to Left. This is initially set to General and date fields are normally right-justified. In this case, left-justification will make the form look better.
7. Add a TextBox control to the Report Footer and enter the following formula for the Control Source property:

```
="Total items checked out: " & Count(*)
```

Formatting the Report Header

Since this report is designed for a receipt-type printer that probably does not support color, you should use only Black and White on this form.

1. To set a White background, select the following properties, click the ellipses and select White from the list of standard colors:
    - Report Header: Back Color
    - Detail: Back Color
    - Detail: Alternate Back Color
    - Report Footer: Back Color
2. Select all of the Text Box controls and change the Back Style and Border Style properties to Transparent.
3. While all the controls are still selected, select Black for the Fore Color.

■ **Tip** If you will be using a lot of reports for black and white printers, you should create a theme that uses only black and white and then you can simply apply that theme to the appropriate reports. See Chapter 11 for instructions on creating a custom theme.

---

www.it-ebooks.info

4. Add a Label control to the Report Header. In the Format tab of the Property Sheet, set the Font Size property to 18 and enter the Caption as **Checkout Receipt**.

Adding a Subreport

When you design a form, you can add a subform that will display details about the currently selected record. If you have set up a link between the master form and the subform, as you navigate through the records in the master form, the data in the subform will dynamically change. Reports do not work this way; they are designed to be printed on paper. You can't have content on paper change based on the record you're looking at. If you link a subreport based on fields in the data source, it will use the data from the first record and ignore the rest.

Subreports are still useful, but they serve a different purpose than subforms. Subreports are often used to include otherwise unrelated data. For example, suppose you needed a daily summary report that includes a summary of items loaned, a summary of item checked back in, and a list of overdue items. You could build these as three separate reports and then include them as subreports on a master report.

Another use of a subreport is for providing details that apply to all records in the report. The `CheckOut` report is a good example of this. All the records on the report are being loaned out to the same customer. So a subreport can be used to display the customer details.

For the `CheckOut` report, the receipt will need to display the customer name and address. The easiest way to do that is to add the `CustomerDisplay` subform to the Report Header. This is already designed to format the name, address, and contact information into a compact space, much like a mailing label.

1. In the Design tab of the ribbon, click the Subform button and then draw a rectangle in the Report Header. The will launch the SubReport Wizard. Select the `CustomerDisplay` form, as shown in Figure 12-35.

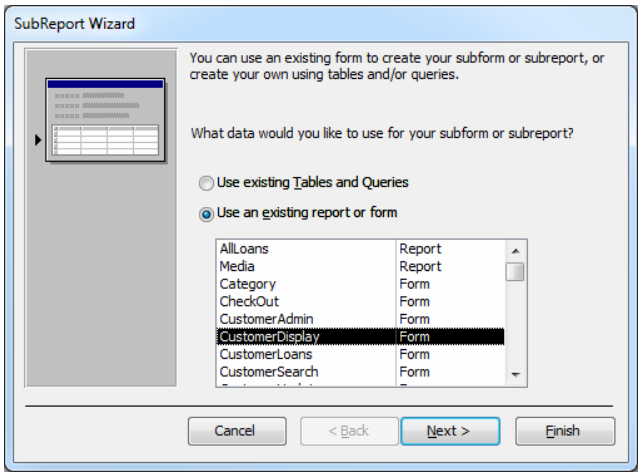


Figure 12-35. *Selecting the* `CustomerDisplay` *form*

www.it-ebooks.info

USING A TABLE OR QUERY FOR A SUBREPORT

There is also an option to use a Table or Query. You won't use this feature for this report but I want to show you how it works, because it is a convenient way to add details from another table. If you select the "Use existing Tables and Queries" option and click the Next button, the dialog box shown in Figure 12-36 will appear.

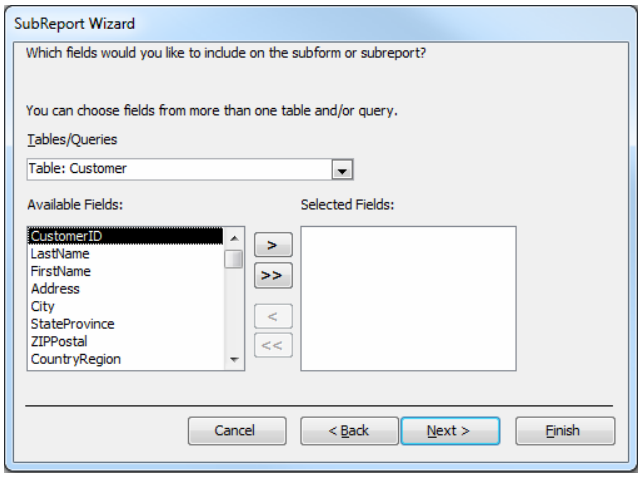


Figure 12-36. *Selecting a table for the subreport*

You select the table or query that you want to use as the data source and then select the fields that should be included. The wizard will then create a subreport for you with the specified fields.

2. As you can see, subreports can use either a form or a report. You should keep in mind, however, that when using a form, the subreport will be read-only, even if the form it is based on allows edits. In this case, the `CustomerDisplay` form is read-only anyway so this is not an issue. The second dialog in the wizard allows you to link the subreport to the master report. Since it found a control named `CustomerID` in both objects, this field is suggested for linking the subreport, as shown in Figure 12-37. This is the field that you'll need to link on so leave this selected and click the Next button.

www.it-ebooks.info

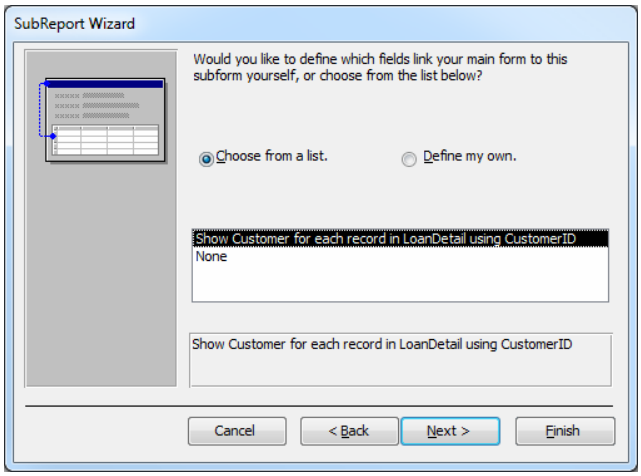


Figure 12-37. Selecting the link details

3. In the final dialog box, shown in Figure 12-38, enter the name **CustomerDisplay** and click the Finish button. The `CustomerDisplay` form will be added to the Report Header.

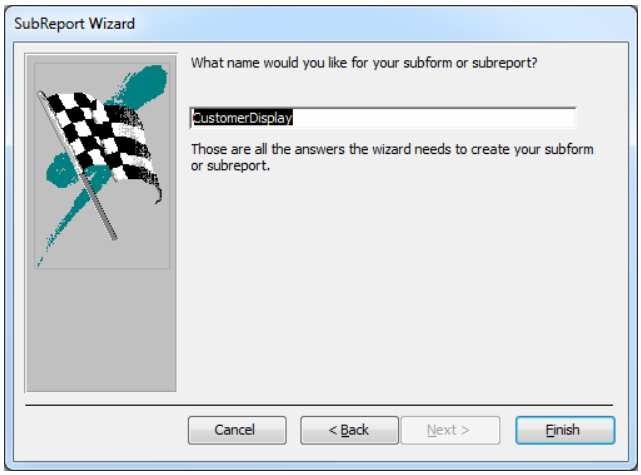


Figure 12-38. Specifying the name of the subreport

4. Delete the associated label control and drag the subreport underneath the Checkout Receipt label.
5. With the subreport still selected, go to the Property Sheet and set the Border Style to Transparent.

www.it-ebooks.info

6. On the `CheckOut` report, add another label underneath the subreport and enter the Caption property as **The following items were checked out:**.
7. Shrink the width of the report by dragging the right edge of the report as far left as it will go.
8. Save the report and entered the name **CheckOut** when prompted.

The design of the report should look like Figure 12-39.

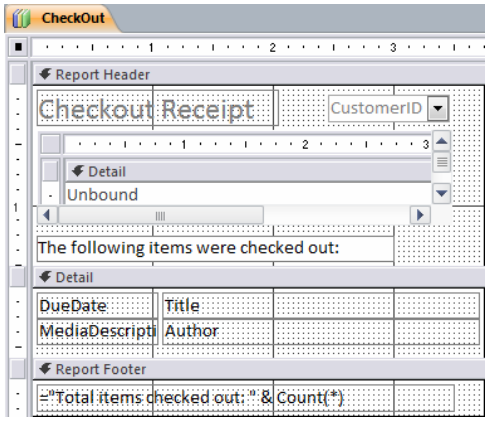


***Figure 12-39**. The final design of the `CheckOut` report*

You can test the report by switching to the Report View. Keep in mind that you haven't setup the correct filter yet so the data will not be correct but you can verify the format looks correct.

Modifying the CheckOut Form

Now you'll need to modify the `CheckOut` form to call the `CheckOut` report when the checkout process is complete. This will be done by adding a few lines of code to the event handler of the `Complete` button.

1. Close the `CheckOut` report and open the `CheckOut` form in the Design view.
2. Right-click the `Complete` button in the Form Footer and click the *Build Event* link. This will display the VBA editor and open the `Form_CheckOut` code file.
3. The complete implementation of the `Complete_Click` method is shown in Listing 12-1. Add the lines in bold to your code file.

***Listing 12-1**. The Implementaton of `Complete_Click`*

```
Private Sub Complete_Click()
    Dim sSQL As String
    Dim sFilter As String
```


www.it-ebooks.info

```
    sFilter = "[CustomerID] = CLng(" & txtCustomerID & ") AND [InProgress] = True"

    DoCmd.OpenReport "CheckOut", acViewReport, , sFilter, acWindowNormal
    DoCmd.PrintOut acPrintAll
    DoCmd.Close acReport, "CheckOut", acSaveNo

    If (Len(txtCustomerID) > 0) Then

        sSQL = "UPDATE Loan SET Loan.InProgress = False " & _
               "WHERE Loan.InProgress=True " & _
               "AND Loan.CustomerID=" & txtCustomerID & ";"
        Application.CurrentDb.Execute sSQL, dbFailOnError
    End If

    txtCustomerID = ""
    txtInventoryItemID = ""

    Me.Requery

    CustomerDisplay.Visible = False
    InventoryItemLookup.Visible = False
    CheckOut.Visible = False

    txtCustomerID.SetFocus
End Sub
```

This code builds a filter that returns the items currently being checked out. It then opens the report passing in the appropriate filter. This will override whatever default filter you entered for testing purposes. The `PrintOut` command causes the current report to be sent to the printer. Finally the report is closed.

Switch the `CheckOut` form to the Form View and check out a few items to a customer. You should have a receipt automatically print that looks like Figure 12-40.


Checkout Receipt

Leonard Boswell 42
1026 Longworth House Office Building
Washington, DC 20515-1503
(202) 225-3806

The following items were checked out:

4/16/2011	White Christmas
CD Audio	Bing Crosby
4/16/2011	It's a Wonderful Life
DVD Video	Frank Capra
4/30/2011	A Christmas Carol
Hardback Book	Charles Dickens

Total items checked out: 3

Figure 12-40. *A Sample* `Checkout` *Receipt*

www.it-ebooks.info

■ **Tip** If you don't have a printer connected or don't want to print out the report while testing, just comment out the last two lines. This will open the report in a separate tab but not print it and close it. You can go to that tab to view the report.

Generating InventoryItem Labels

The items that you have available to be checked out will need some sort of inventory identification. This will indicate that the item belongs to your library and include an ID number that uniquely identifies that copy. This is used when checking out an item. You might want to include the title and perhaps a shelf location for re-shelving purposes. You can easily generate these labels in Access using the Label Wizard. A sample label report is shown in Figure 12-41.

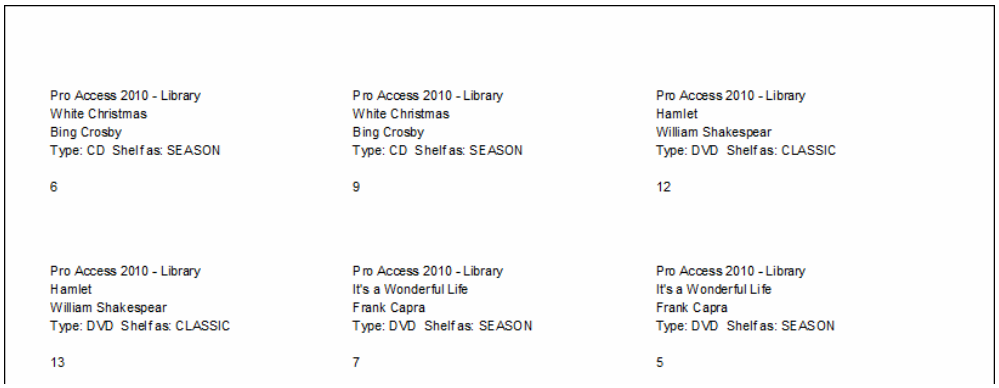


***Figure 12-41**. A sample label report*

Creating an InventoryItemDetail Query

The Label Wizard uses an existing table or query for the data source. The `InventoryItem` table does not include details such as the `Title`; these are stored in the `Item` table. So you must first create a query that contains all the fields you'll need for the labels.

1. From the Create tab of the ribbon, click the Query Design button.
2. Add the following tables to the query:
   - `InventoryItem`
   - `Item`
   - `Category`
   - `Media`

www.it-ebooks.info

■ **Note** The joins between the tables should be set up automatically based on the existing table relationships.

3. Double-click the following fields to add them to the query:
   - `InventoryItem.InventoryItemID`
   - `Item.ItemID`
   - `Item.Title`
   - `Item.Author`
   - `Category.CategoryCode`
   - `Media.MediaCode`
4. Save the query and enter the name **InventoryItemDetail** when prompted.

The query design should look like Figure 12-42.

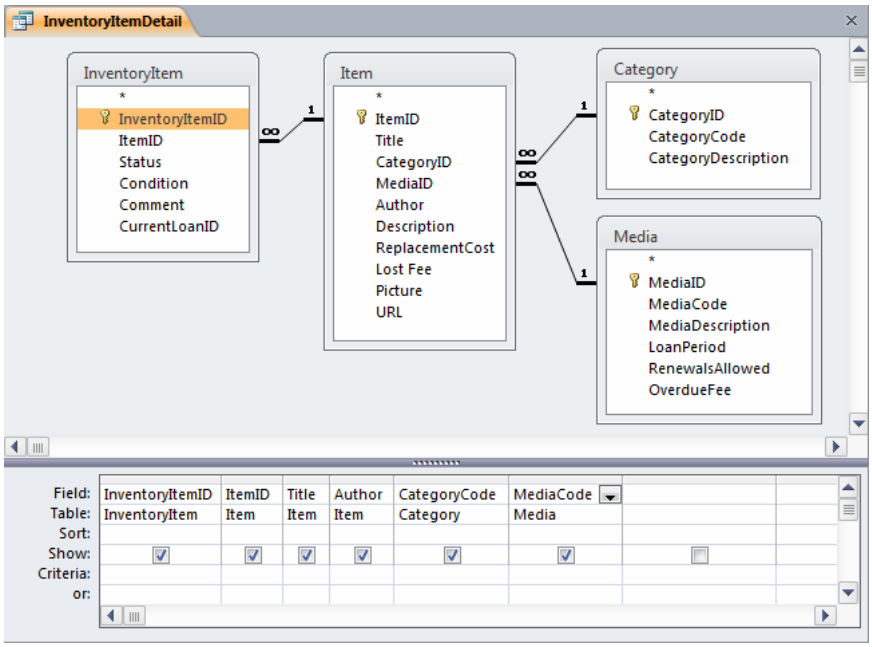


Figure 12-42. *The* `InventoryItemDetail` *query design*

www.it-ebooks.info

Using the Label Wizard

Now you'll use the Label Wizard to build a report that will generate the inventory labels. The wizard has to first identify the label stock that you're using so it can align the data properly. Then you'll format the label by specifying the text to be printed on each line. To launch the Label Wizard, from the Create tab of the ribbon, click the Label button.

Selecting the Label Stock Template

Access ships with a large selection of pre-formatted templates. Just choose the vendor and then select from the list of product numbers. You can also define your own template, if necessary. Select the Avery 2160 label stock as shown in Figure 12-43.

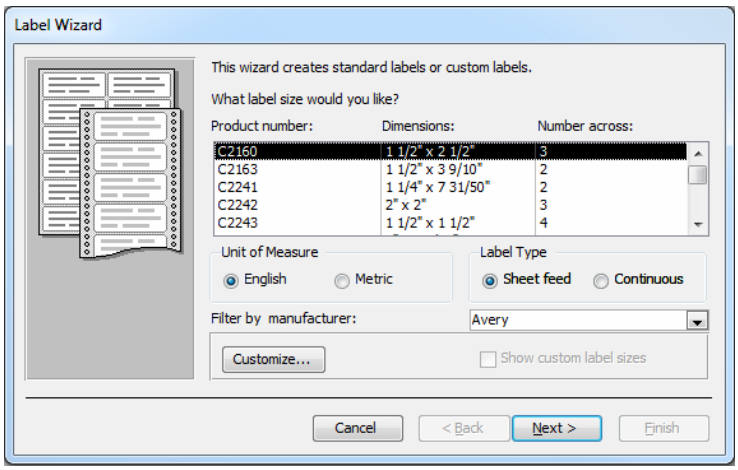


Figure 12-43. *Selecting the label stock*

DEFINING CUSTOM LABEL TEMPLATES

If you don't find the stock you're using, you can set up a custom template. Click the Customize button on the first dialog box, which will list the custom templates. This list will be empty, initially. Click the New button to define a new template. The New Label dialog box shown in Figure 12-44.

www.it-ebooks.info

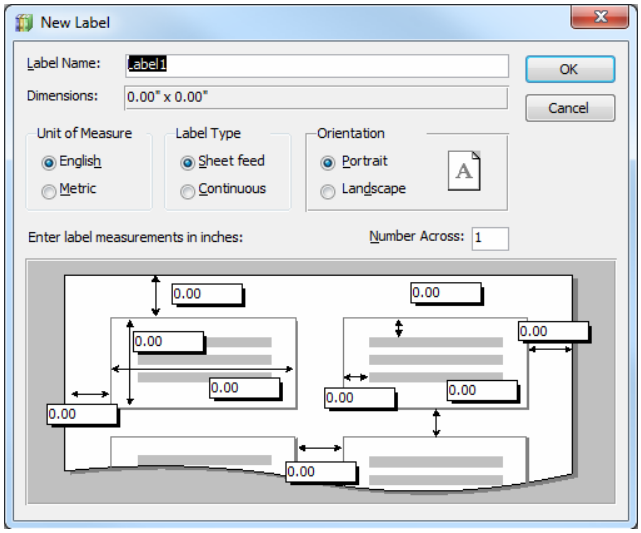


Figure 12-44. Defining a new label stock template

There are nine measurements that you'll need to enter to define the label stock. Select the units, whether English or Metric, and then type the values directly on the form. The Label Name and Dimensions fields at the top of the dialog box are for reference purposes, so you can identify this template when you want to use it later. The Dimensions field is filled in automatically as you supply those measurements.

Once the template has been set up, you can use it again later. In the first dialog box of the Label Wizard, select the "Show custom label sizes" check box and your custom templates will be listed in the dialog box.

Formatting the Labels

Once you have selected the label template you're ready to format the text that will appear on the label. You will first define the font that should be used. Then you'll specify the text for each line and finally define the sort criteria.

1. In the second dialog box, shown in Figure 12-45, use all the default values.

www.it-ebooks.info

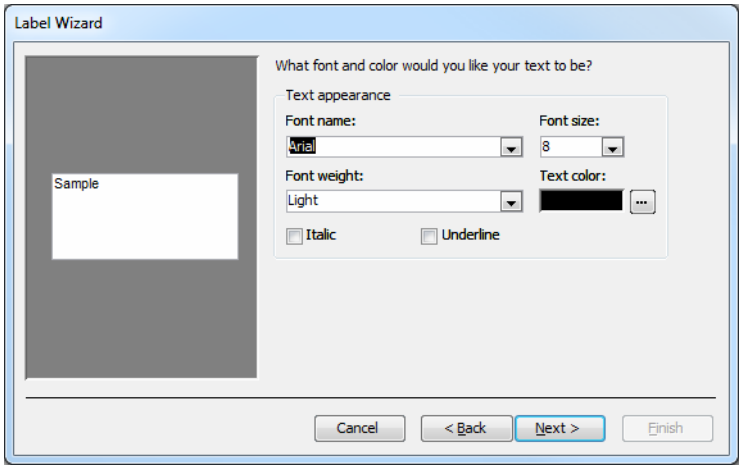


Figure 12-45. *Specifying the font attributes*

2. In the third dialog box, you'll specify what text should go on each line. The right side of the dialog box contains a prototype of the label. Access calculates the number of lines that will fit based on the label dimensions and the font that was chosen. The Label Wizard supports a maximum of eight lines, but it could be fewer than that. In this case only six lines can be used. You format each line separately. When you select one of the lines on the right side it will be highlighted. On each line you can type static text and/or add one or more fields from the list of available fields. For the first line, enter the static text **Pro Access 2010 – Library**.

3. For the second and third lines, double-click the `Title` and `Author` fields, respectively.

4. The fourth line will contain both static text and data-bound fields. Enter **Type:** and then double-click the `MediaCode` field. Then type **Shelf as:** and select the `CategoryCode` field.

5. Leave the fifth blank.

6. For the sixth line, add the `InventoryItemID`. The format should look like Figure 12-46.

www.it-ebooks.info

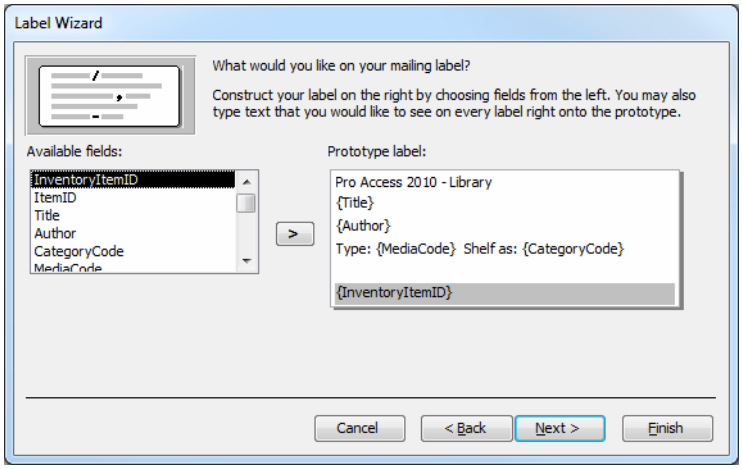


Figure 12-46. *Formatting the label*

7. In the fourth dialog box you'll specify the sort order. Enter the sort fields as `MediaCode`, `CategoryCode`, and `Author` as shown in Figure 12-47. This probably matches how they are grouped on the self.

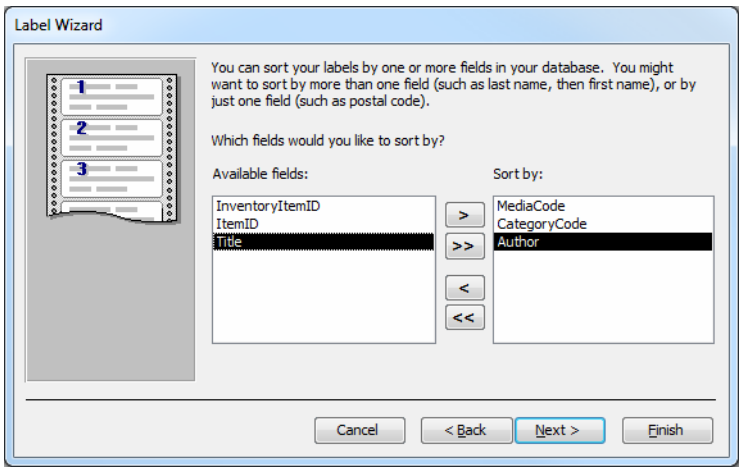


Figure 12-47. *Specifying the sort order*

8. In the final dialog box, enter the name as **InventoryLabels**, as shown in Figure 12-48.

www.it-ebooks.info

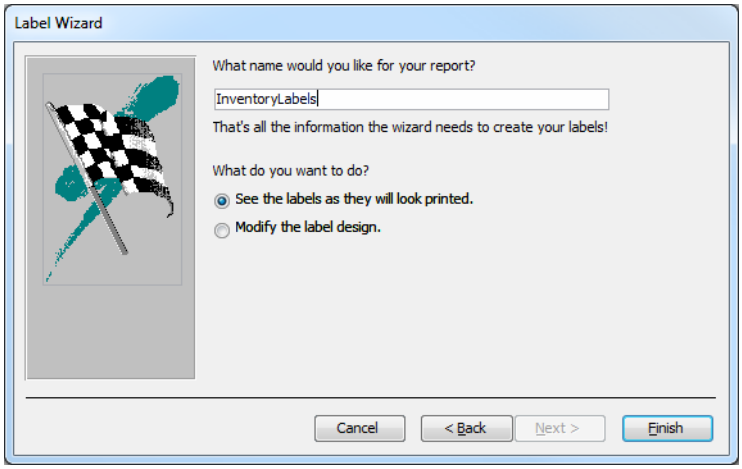


Figure 12-48. *Entering the report name*

■ **Note** An advanced application might even format the `InventoryItemID` value as a barcode to allow this to be scanned during the checkout process. Barcode support is not provided natively in Access so you would need to use a third-party ActiveX control.

Modifying the Color Scheme

The Label Wizard will then create the report and display it using the Print Preview View. You'll need to fix the color scheme for this report.

9. Click the Close Print Preview button in the ribbon. This will close that window and open the `InventoryLabels` report using the Design View.
10. In the Property Sheet, select the Detail object and set the Back Color to the standard White color.
11. Press Ctrl-A to select all the controls. From the Format tab of the Property Sheet, set the Back Color property to White.
12. Save the changes to the `InventoryLabels` report and switch to the Print Preview View.

The final report should look like Figure 12-49.

www.it-ebooks.info



Figure 12-49. *The final* `InventoryLabels` *report*

Auto-Generating a DailyLoans Report

For the final project in this chapter, let's say you've been asked to generate a daily report that lists the items that were loaned during that day. The users want this to run every day automatically at the same time each day, after the library has closed. To avoid any printer issues (like someone forgetting to leave paper in the printer) they've asked you to save the report as a PDF file. Then they can view or print it whenever they want.

So, you'll create a `DailyLoans` report that will list the items that were loaned out today. This will be a copy of the `AllLoans` report that you created earlier, except you'll add a filter to limit the result to the items checked out today. Then you'll implement a macro that will open this report and save it as a PDF file. Finally you'll create a shortcut to call this macro and use the Windows Task Scheduler to generate this report at the same time every day.

Creating the DailyLoans Report

This is a pretty easy step. You'll make a copy of the `AllLoans` report, add a filter and change the report title.

1. Open the `AllLoans` report using the Design View. From the File tab, click the "Save Object As" button. In the Save As dialog box, enter the name **DailyLoans**, as shown in Figure 12-50.

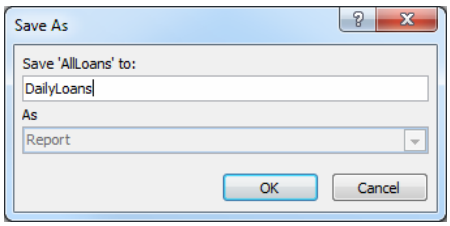


Figure 12-50. *Save a copy of the* `AllLoans` *report*

www.it-ebooks.info

2. Go back to the Home tab and notice the window caption has been changed to `DailyLoans`.
3. Select the label in the Report Header and enter **Daily Loans** for the Caption property.
4. In the Property Sheet, select the Report object and the Data tab. Set the "Filter On Load" property to Yes.
5. For the Filter property enter the following formula:

```
[CheckedOut] > Now() - 1
```

■ **Caution** This formula takes advantage of some assumptions. If your application is running in a 24x7 operation, you'll need a more complex formula that checks for loans checked out between a begin date and end date. However, this formula works for this demonstration.

---

6. Save the changes to the `DailyLoans` report and close this window.

Creating a DailyReport Macro

You'll create a macro called `DailyReport` that can be run from a Windows shortcut. This macro will simply call a VBA function, just like the `autoexec` macro you created in Chapter 10. You'll start by implementing the VBA method and then define the macro to call it.

In the Navigation pane, double-click the `Main` module. This will display the VBA editor and open the `Main` module. Add a new method to this file using the code shown in Listing 12-2. Save the code changes.

Listing 12-2. *Implementation the the* ***`DailyReport`*** *Method*

```
Public Function DailyReport() As Integer

Dim sPath As String
Dim n As Integer
Dim sMonth As String
Dim sDay As String

    ' Compute the base path
    sPath = CurrentProject.FullName
    n = InStrRev(sPath, "\", Len(sPath))
    sPath = Left(sPath, n)

    ' Compute the filename
    sMonth = Month(Now())
    sDay = Day(Now())
    pad = "0"
```


www.it-ebooks.info

```
    If (Len(sMonth) = 1) Then
        sMonth = "0" & sMonth
    End If
    If (Len(sDay) = 1) Then
        sDay = "0" & sDay
    End If

    ' Build the full report path
    sPath = sPath & "Reports\DailyLoans" & Year(Now()) & sMonth & sDay & ".pdf"

    ' Print the check out report
    DoCmd.OutputTo acOutputReport, "DailyLoans", "PDFFormat(*.pdf)", _
                  sPath, False, , , acExportQualityPrint

End Function
```

This method will generate a PDF file in the `Reports` subfolder. The filename will be `DailyLoansyyyymmdd.pdf`, where `yyyymmdd` is the current date. Most of this code performs string manipulation to build the path and filename. The last line of this method does the interesting work. It uses the `OutputTo` command to generate a PDF.

From the Create tab of the ribbon, click the Macro button to create a new macro. Select the `RunCode` action and enter **DailyReport**() for the Function Name parameter. Save the macro and enter the name **DailyReport** when prompted.

The macro design should look like Figure 12-51.

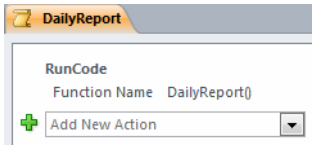
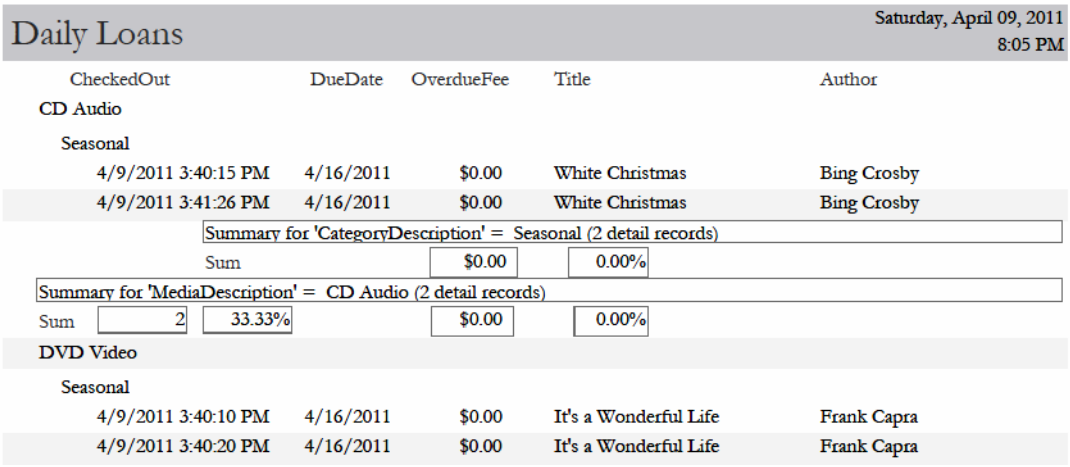


Figure 12-51. *The design of the* `DailyReport` *macro*

Make sure that you have a `Reports` subfolder relative to where the Access database is located. This will be a sibling folder to the `Images` folder that you created in Chapter 9. To test the macro, click the Run button in the Design ribbon. Check the `Reports` folder and there should be a new PDF file that looks something like Figure 12-52.

www.it-ebooks.info



Daily Loans — Saturday, April 09, 2011 8:05 PM

CheckedOut	DueDate	OverdueFee	Title	Author
CD Audio				
Seasonal				
4/9/2011 3:40:15 PM	4/16/2011	$0.00	White Christmas	Bing Crosby
4/9/2011 3:41:26 PM	4/16/2011	$0.00	White Christmas	Bing Crosby
Summary for 'CategoryDescription' = Seasonal (2 detail records)				
Sum		$0.00	0.00%	
Summary for 'MediaDescription' = CD Audio (2 detail records)				
Sum 2	33.33%	$0.00	0.00%	
DVD Video				
Seasonal				
4/9/2011 3:40:10 PM	4/16/2011	$0.00	It's a Wonderful Life	Frank Capra
4/9/2011 3:40:20 PM	4/16/2011	$0.00	It's a Wonderful Life	Frank Capra

Figure 12-52. *A sample* `DailyLoans` *report*

Creating a Scheduled Task

You can automate any macro in Access by first creating a Windows shortcut to the macro and then use the Task Scheduler utility to call the shortcut on whatever schedule you need. Creating a shortcut is as simple as dragging the macro in Access to a folder.

In the Navigation pane, find the `DailyReport` macro. Then open Windows Explorer and browse to a location on the local drive. (I created a `C:\AccessMacros` folder, but you can use a different folder if you prefer.) Click the `DailyReport` macro and drag it to the desired folder on the C: drive. This should create a shortcut named `Shortcut to DailyReport in Library.accdb`. You can rename this if you want to, the name is not important.

1. Start the Task Scheduler utility. This is normally found in the Start menu under *Accessories* ➤ *System* Tools.
2. In the Actions page on the right side, click the *Create Basic Task* link. This will start the Create Basic Task Wizard.
3. In the first dialog box, enter a name and description, as shown in Figure 12-53.

www.it-ebooks.info

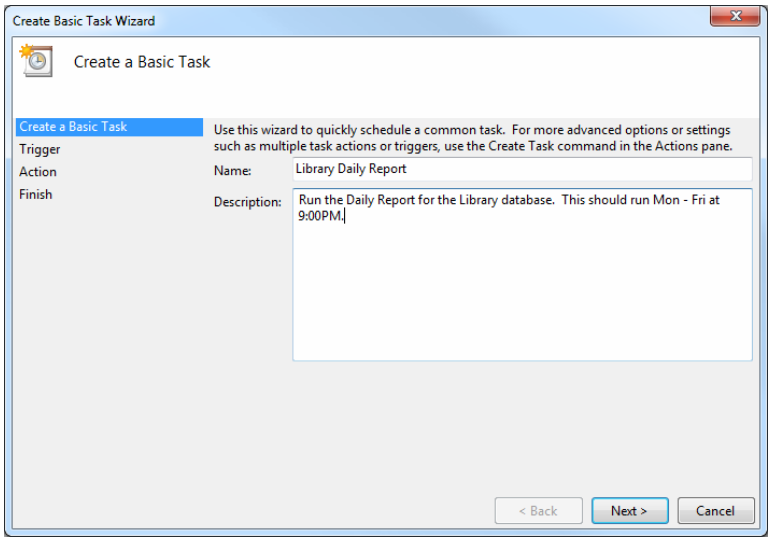


Figure 12-53. *Entering the name and description*

4. In the second dialog box select the Weekly option.

5. In the third dialog box, specify a start time of 9:00pm and select Monday–Friday, as shown in Figure 12-54. This will cause the task to be triggered at 9:00pm on each of the selected days (Mon–Fri).

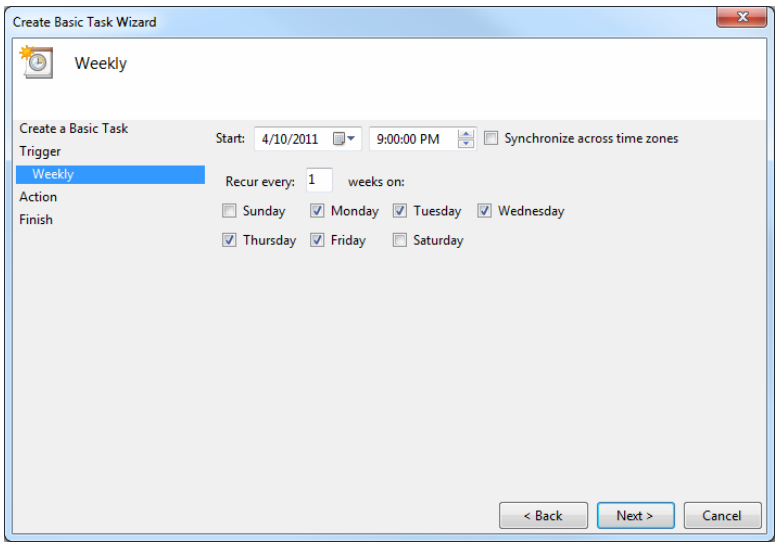


Figure 12-54. *Specifying the run schedule*

www.it-ebooks.info

6. In the fourth dialog box select the “Start a program” option.
7. In the fifth dialog box, browse to the location of the shortcut that you want to start, as shown in Figure 12-55.

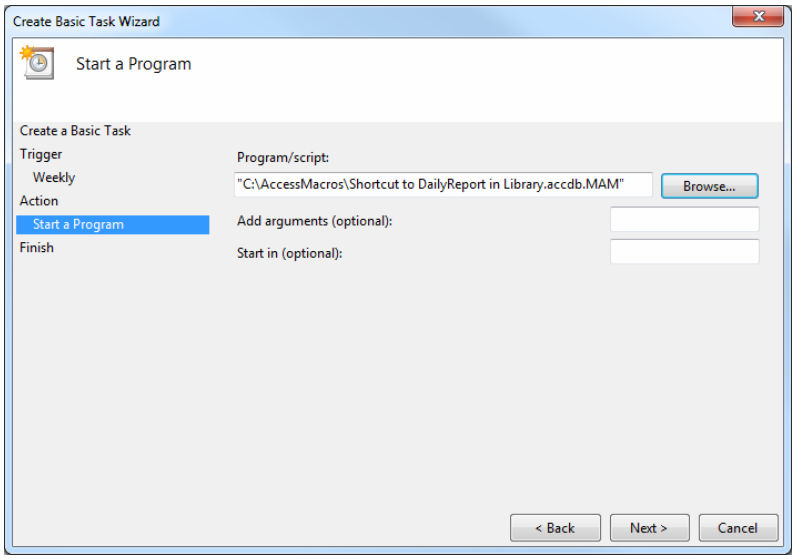


Figure 12-55. *Selecting the shortcut to be run*

8. The final dialog box, shown in Figure 12-56, provides a summary of the task that will be created. Click the Finish button to complete the wizard and create the task.

www.it-ebooks.info

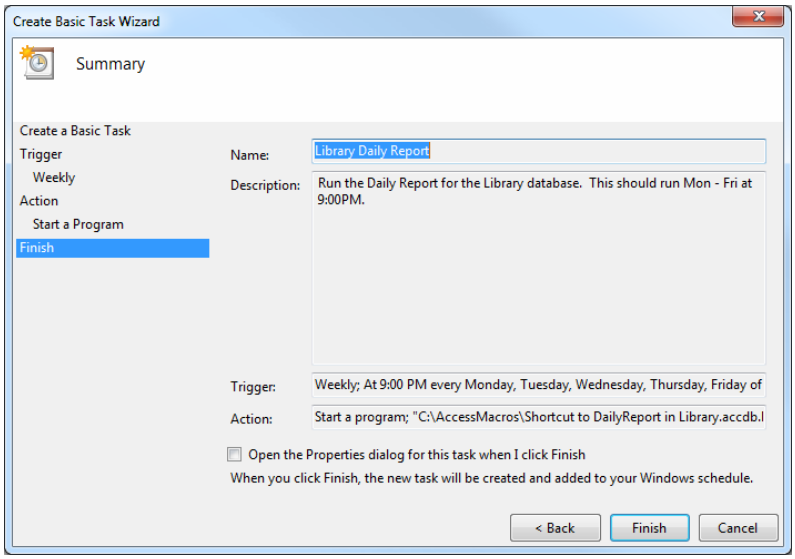


Figure 12-56*. The summary of the scheduled task*

Modifying the Daily Report Macro

When the shortcut is invoked it will start the Access application and generate a report as a PDF file. This macro will also need to then close the Access database. I purposely left that step out while testing, because you probably don't want the application to close while you're working on it.

Open the `DailyReport` macro using the Design View. This will display the Macro Designer. In the Add New Action dropdown list, select the `QuitAccess` action and select Exit for the Options parameter. The final design should look like Figure 12-57.

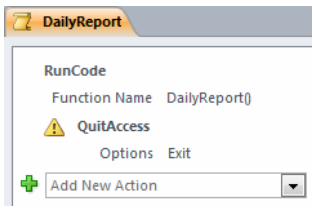


Figure 12-57*. The final macro design*

Now you're ready to test this feature. Save your macro changes and close the Access database. Using Windows Explorer, go to the C:\AccessMacros folder (or wherever you saved the shortcut). Double-click the shortcut that you created earlier to start it. You should see Access start and run for a few seconds and then close. Look in the Reports folder and you should see a file with a current date/time.

www.it-ebooks.info

Summary

Reports are essentially read-only forms. They support some unique features such as grouping and pagination but most of the techniques you used to design forms also apply to reports. Access provides several ways to create a report:

- Start with a blank report (in either the Design View or Layout View)
- Use the Report Wizard to configure a report
- Create a simple report with a single click
- Use the Label Wizard if you need to generate labels

In this chapter you created several reports that demonstrate all of these techniques. Some of the significant features that you used include:

- Using multiple nested groups with subtotals
- Using unbound controls to add calculated data
- Using a form as a subreport
- Generating labels
- Printing a report with VBA code
- Generating a report as a PDF file
- Using the Task Schedule to auto-run a report

www.it-ebooks.info

PART 4

Multiuser Considerations

The Access database you have created so far works great for a single person or a small number of users. However, as you start to deploy it for a larger group of people, you'll need to make some adjustments that will allow it to scale.

In Chapter 13, you'll look at upsizing your database. This technique moves the data to a separate data store so the front-end can be replicated as necessary. You use the following data sources to gain an understanding of the benefits of each:

- Access 2010
- SQL Server
- SQL Azure

In Chapter 14, I'll show you how to use the Access runtime, which will allow users limited capability without needing an Office 2010 license.

In Chapter 15, you will publish your database to a SharePoint site. Not only does this move the data to a central location, but you can also provide web forms and reports so users can view and update the data right from a browser.

www.it-ebooks.info

CHAPTER 13

Upsizing

From a developer's perspective, having the data and UI elements such as forms and reports in a single database is great. You can make a copy of the database, try out some ideas, test it… and if you don't like the results. just delete it and go back to the original database. It's also very convenient; you don't have to configure database connections, you just open the file like you would a Word or Excel file. However, if you have more than just a few users, these same benefits will become your biggest issues.

Understanding Upsizing

There is some debate as to just how well Access will scale to support more users, data, and throughput. Before you come close to any of these theoretical limits, however, you will need to split your application by separating the data from the UI objects. A single data store will be shared by all users, while the front-end application can be distributed. This technique is call *upsizing*. There are several advantages to upsizing your Access applications:

- By distributing the front-end Access file, each user can have their own copy, which will eliminate the file-locking issues that occur when two people try to open the same file.
- Network traffic is reduced, because the UI elements don't have to be loaded from a shared network location.
- New versions of the application can be developed, tested, and deployed that use the same common data store. This allows for incremental rollout rather than an all-or-nothing scenario.

■ **Note** Access 2010 supports up to 255 concurrent users. The other significant limitation is a maximum file size of 2GB. For a complete list of specifications go to `http://office.microsoft.com/en-us/access-help/access-2010-specifications-HA010341462.aspx`.

Using Linked Tables

In addition to the embedded tables that you have been using so far, Access supports *linked* tables. Data that exists outside of the Access database is considered external data. When you set up an external table,

www.it-ebooks.info

Access creates a link to that data. Once linked, you can then use that table just like the internal (or embedded) tables.

Access uses Open Database Connectivity (ODBC) to connect to external data. You can link to any data source for which you have an ODBC driver. Normally, these drivers come with the database software. For example, Office 2010 includes drivers for Access and Excel. Office also comes with ODBC drivers for SQL Server. There are also OBDC drivers available for Oracle, DB2, and MySQL. You can write your own ODBC driver as well, but that is beyond the scope of this book.

■ **Caution** The intent of this architecture is to be able to seamlessly use linked tables just as if they were embedded tables, and, most of the time, this is achieved. However, because you're using an external data source, you are bound by whatever limitations that data source might impose. In addition, the OBDC driver can introduce limitations. For example, older drivers might not support all of the new database features.

Upsizing an Access database takes advantage of linked tables to accomplish the separation of data and UI. Upsizing is a simple matter of moving the data to an external source and replacing the embedded tables with linked tables. The details will vary slightly depending on what type of data source you're using.

Project Overview

In this chapter, you will upsize your Access to the following data sources:

- Access
- SQL Server
- SQL Azure

■ **Note** This project will require you to have access to these data sources. You can skip one or more of these sections if these are not available.

Before you start, make three copies of your `Library.accdb` file and give them the following names:

- `Library_Access.accdb`
- `Library_SQL.accdb`
- `Library_Azure.accdb`

Throughout the remainder of this chapter, I'll show you how to move the data in each of these files to the corresponding external data source. When you're done, each version will work like it does now, but the actual data will reside in an external data source.

www.it-ebooks.info

Upsizing with Access

The simplest and easiest way to upsize an Access application is to split it into two files. A backend file will contain the tables and the remaining objects will be in the other, frontend file. Access provides a simple utility called Database Splitter that makes this job easy.

In the Database Tools tab of the ribbon, click the Access Database button, as shown in Figure 13-1.

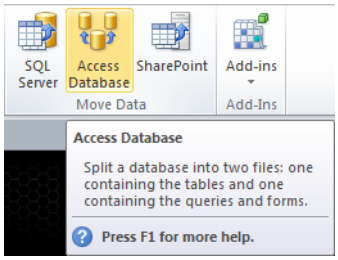


Figure 13-1. *Selecting the Access backend database*

This will display the Database Splitter dialog box shown in Figure 13-2. This utility does not require any configuring; it simply moves all of the tables to a new Access database. The dialog box explains what the utility does and recommends you first make a backup. You still have the original `Library.accdb` file so no additional backup is necessary.

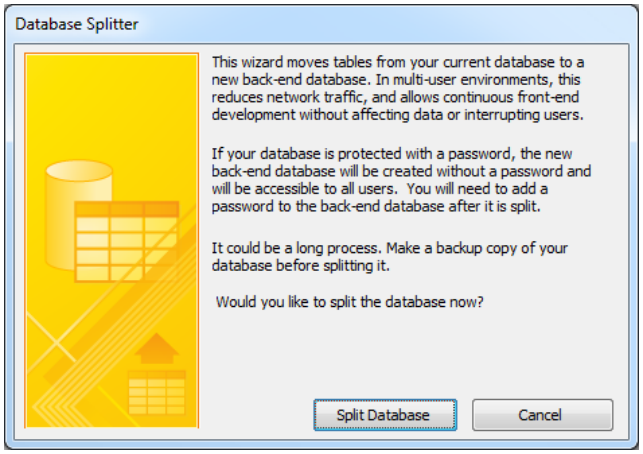


Figure 13-2. *The Database Splitter dialog box*

Click the Split Database button. You will then be prompted to select the name and location for the backend file. Normally, you would save this file to a shared network location to which all users will have access. For this exercise, you can put it in the same folder as the original file if you prefer. The proposed name of `Library_Access_be.accdb` is fine. The "_be" suffix indicates that this is a backend file. Click the

www.it-ebooks.info

Split button to start the process. When the split is completed, you'll see the dialog box shown in Figure 13-3.

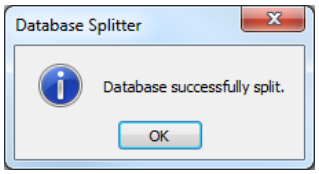


Figure 13-3. *Confirmation that the split was successful*

Close this dialog box and the Database Splitter window will close as well. Notice in the Navigation page that the tables all have an arrow next to them, as shown in Figure 13-4. This indicates that these are now linked tables.

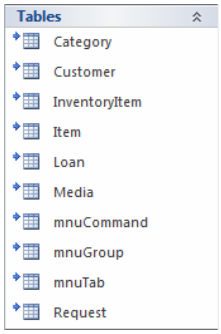


Figure 13-4. *The Navigation pane showing the linked tables*

Right-click any of these tables and click the *Linked Table Manager* link. This will display the Linked Table Manager shown in Figure 13-5. Click the Cancel button since you don't need to change anything.

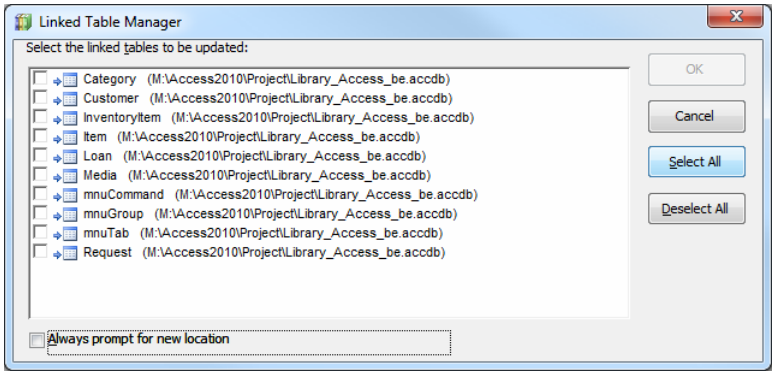


Figure 13-5. *The Linked Table Manager*

www.it-ebooks.info

The Linked Table Manager is a handy utility that shows all of the linked tables in your database and lets you know to where each is linked. Access allows each table to be linked to a different external source.

If you decide to move the backend database, you'll need to update the links. The easiest way to do that is to use the Linked Table Manager. Select all the tables that are linked to that data source and select the "Always prompt for new location" check box. When you click the OK button, Access will prompt you to select the appropriate Access file for each linked table.

Open the new back end database, `Library_Access_be.accdb`. Notice that the tables are in this database and there are no other objects in the database, as shown in Figure 13-6.

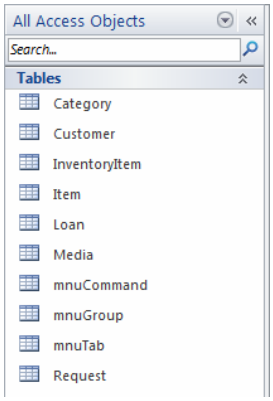


Figure 13-6. *The contents of the* `Library_Access_be.accdb` *database*

Because the front-end Access "database" has no data in it, you can distribute it so each user has their own copy and still share a common data repository. I will explain that process in the next chapter.

Upsizing with SQL Server

Upsizing an Access database with SQL Server uses the same principle, except that data is moved to a SQL Server database. You can use just about any version of SQL Server from 6.5 and later, including SQL Server Express.

There are two utilities available that will move the data and set up linked tables like the Database Splitter that you just used. Access comes with an Upsizing Wizard, which you'll use first. Microsoft also provides a free application called SQL Server Migration Assistant (SSMA), which I will demonstrate later.

Using the Upsizing Wizard

If you have access to an existing SQL Server instance, you can easily create a database to host your Access data. This will give you all the benefits of splitting the data from the UI that I described earlier. In addition, you'll be able to take advantage of the performance, scalability, and availability that SQL Server provides.

www.it-ebooks.info

■ **Tip** If you don't have a SQL Server, you can install SQL Server Express, which is a free version of SQL Server. It has most of the same benefits of SQL Server and will work quite well as a backend database for your Access application. To install SQL Server Express, download the installation file from `www.microsoft.com/express/Database` and then run the executable to install SQL Server Express on your workstation.

1. Open the `Library_SQL.accdb` database. From the Database Tool tab of the ribbon, click the SQL Server button as shown in Figure 13-7. This will start the Upsizing Wizard.

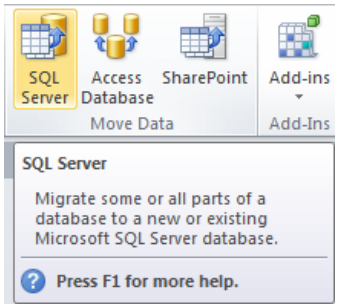


Figure 13-7. *Starting the Upsizing Wizard*

2. In the first dialog box, you can choose to create a new database or add these tables to an existing database. When you're upsizing an application for the first time you should create a new database. If you add tables and want to move only the new ones, you'll select an existing database. Select the "Create new database" option, as shown in Figure 13-8.

www.it-ebooks.info

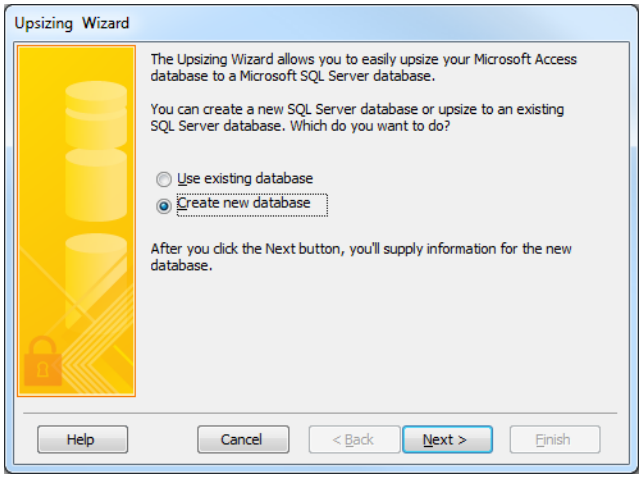


Figure 13-8. *Creating a new database*

3. In the second dialog box, shown in Figure 13-9, you'll specify the database server and the appropriate credentials. You'll need to use a SQL Server account that has `CREATE DATABASE` access. You can use a trusted connection if your Windows login is set up with SQL Server with the necessary access. Otherwise, you can specify a SQL Server login such as **sa** and the corresponding password. Also enter **Library** for the database name.

■ **Tip** Normally, SQL Server is installed as a default instance, which means there is only one installation on a server. In this case, you only need to supply the server name. If you have more than one version installed on the same server then one or more will have a named instance and you'll specify the server as <server name>\<instance name>. SQL Server Express is usually installed as a named instance and is typically named SQLEXPRESS.

---

www.it-ebooks.info

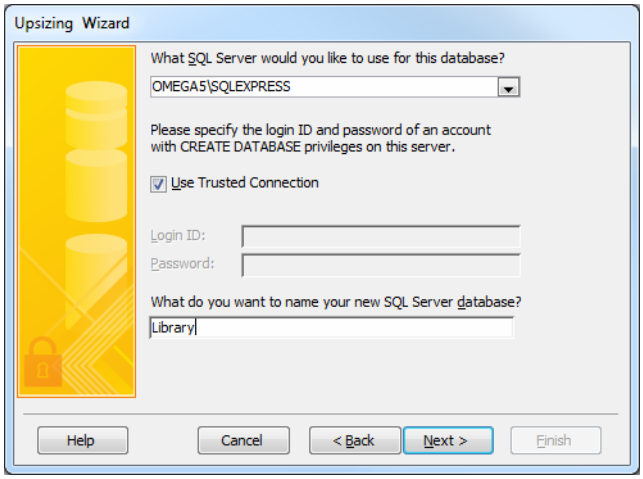


Figure 13-9*. Specifying the database server and database name*

4. The third dialog box lists the existing tables in the Access application. Click the ">>" button to select all of the tables, as shown in Figure 13-10.

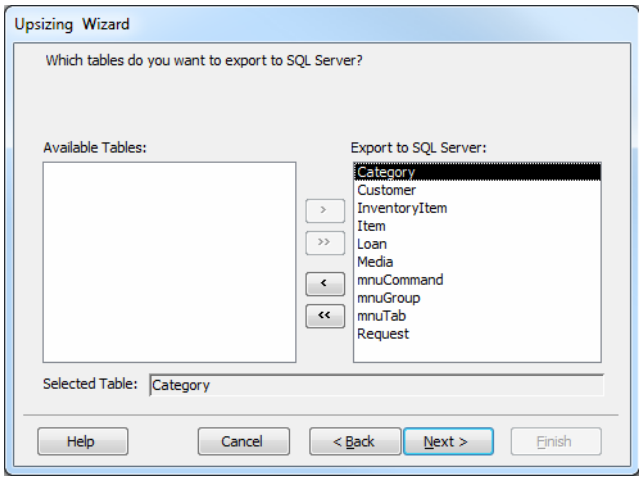


Figure 13-10*. Selecting the tables to export*

5. In the fourth dialog box, you can specify the database attributes that you want carried over to SQL Server. The dialog box should look like Figure 13-11, after you've done the following:

   - Select all four checkboxes (Indexes, Validation rules, Defaults, and Table relationships).

www.it-ebooks.info

- Select the Use DRI option.
- Select "Yes, let system decide" for the "Add timestamp fields to tables?" dropdown list.
- Make sure the "Only create the table structure; don't upsize any data." checkbox is not selected.

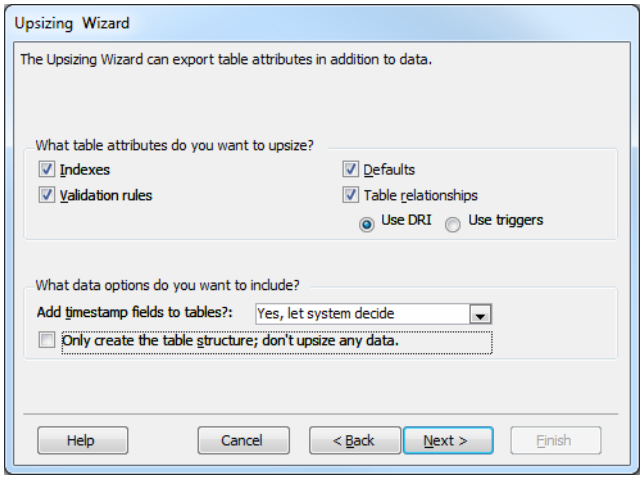


Figure 13-11*. Choosing the database attributes*

REFERENTIAL INTEGRITY

Recall from Chapter 2 that, when defining lookup columns and configuring the referential integrity, there were two options. When deleting a record, the Cascade Delete option would automatically delete any records that referenced that record. This is useful for a parent/child relationship. If the parent is deleted, all the child records are also deleted. In contrast, the Restrict Delete option would prevent a record from being deleted if other record referenced it. This option would require any references to be removed first.

The "Use DRI" and "Use triggers" options in Figure 13-11 follow this same logic. The "Use DRI" option uses SQL Server referential integrity to prevent deletes and is equivalent to the Restrict Delete option. In contrast, the "Use triggers" option will implement code in a database trigger to cascade the delete to records that reference the record being deleted. When designing the tables, the Restrict Delete option was used so you should use the Use DRI option here.

www.it-ebooks.info

■ **Note** There is a good article on MSDN that explains all the options on this page. If you want more details checkout the information at `http://msdn.microsoft.com/en-us/library/aa157041.aspx`.

6. The fifth dialog box, shown in Figure 13-12, provides the following three options for modifying the client application; select the "Link SQL Server tables to existing application" for this example:

   - Create a new Access file that contains only the UI objects and linked tables. As this is not technically a database (because the data resides on SQL Server), the file extension will be `.adp`.
   - Add linked tables to the existing Access database. This will also rename the existing tables with a `_local` suffix.
   - "No application changes" will copy the tables and data to SQL Server but the applications will continue to use the internal tables.

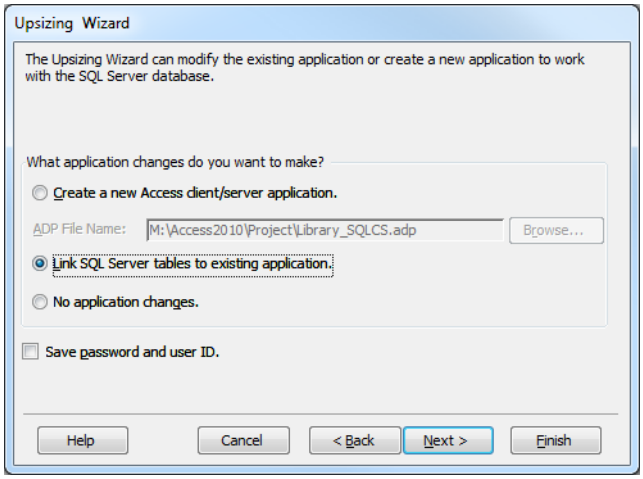


Figure 13-12. *Specifying the client application changes*

7. The final dialog box, shown in Figure 13-13, lets you know that the configuration details have been entered. Click the Finish button to start the upsizing process.

www.it-ebooks.info

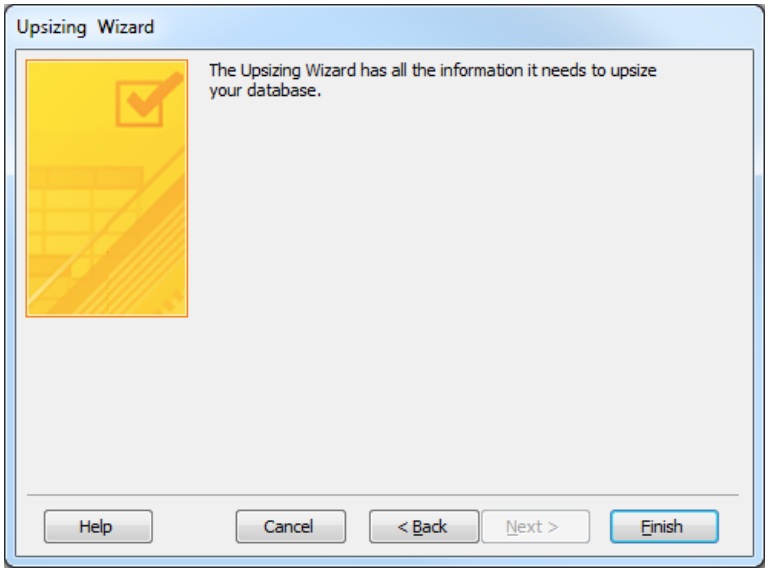


Figure 13-13*. Completing the Upsizing Wizard*

When the wizard has completed the update, the `Upsizing Wizard` report will be displayed in Print Preview. A portion of that report is shown in Figure 13-14.

www.it-ebooks.info

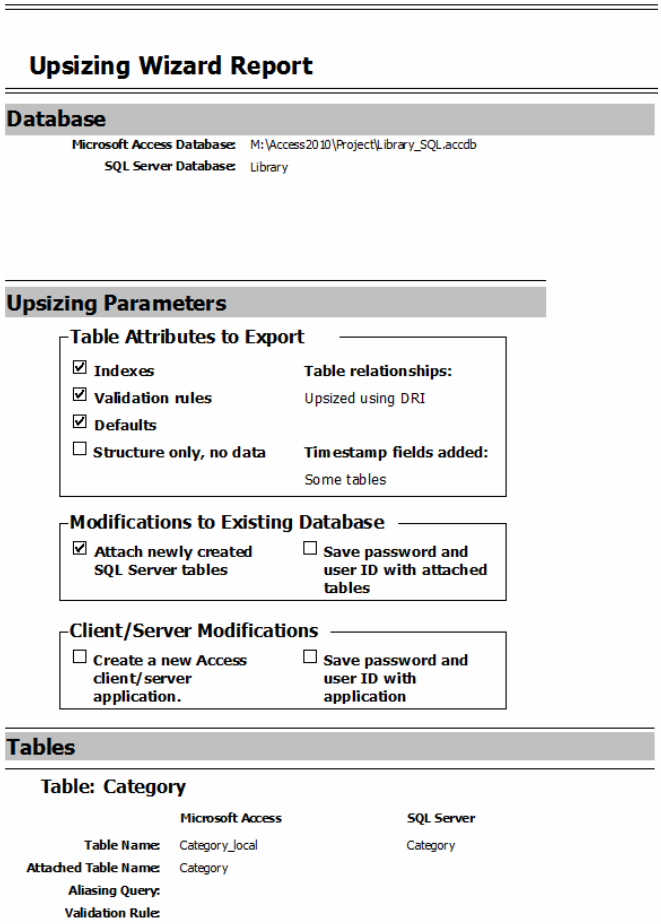


Figure 13-14. *The Upsizing Wizard Report*

Viewing the Upsizing Results

The upsizing is complete; the data has been moved to SQL Server and the client application is now using the linked tables.

Go to SQL Server using the SQL Server Management Studio (SSMS) to verity the database is set up properly. Expand the Object Explorer and you should see the `Library` database. In Figure 13-15, I have expanded the list of tables and then expanded the `InventoryItem` table. This shows the columns, keys, constraints, and indexes that were generated.

www.it-ebooks.info

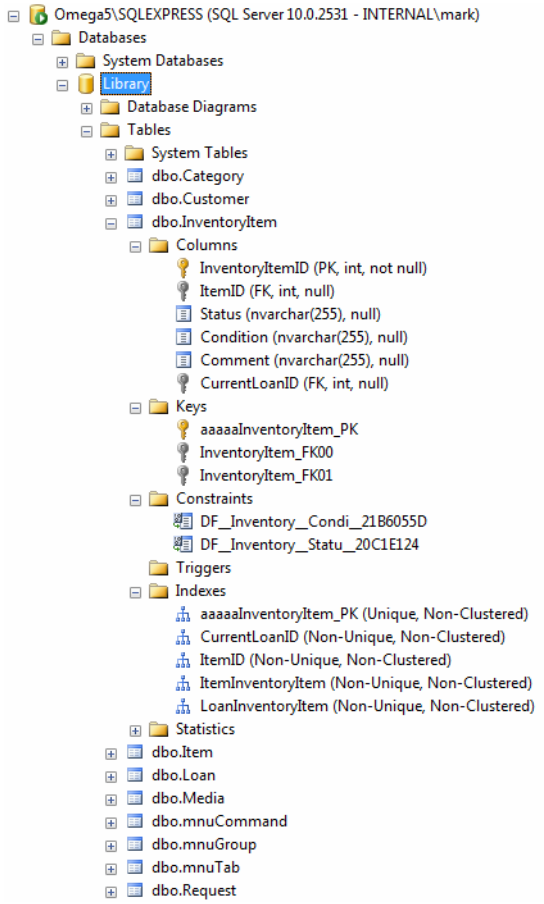


Figure 13-15. *The* `Library` *database objects in SQL Server*

In the Access application, right-click one of the linked tables and select the *Linked Table Manager* link. All of the linked tables should all show that they are linked to the `Library` database, as shown in Figure 13-16.

www.it-ebooks.info

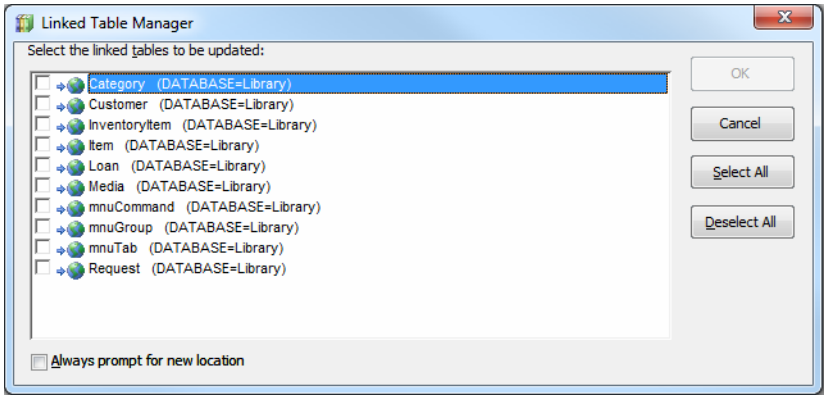


Figure 13-16. *The Linked Table Manager*

Hover the mouse over one of the linked tables to see the actual connection string that is generated, as shown in Figure 13-17.

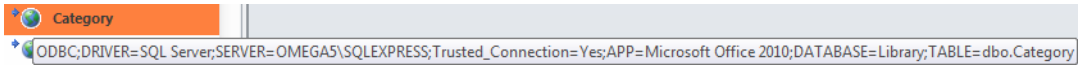


Figure 13-17. *The database connection for a linked table*

■ **Note** The Upsizing Wizard does not generate a Data Source Name (DSN). Historically, when using ODBC, you would create a DSN, configure it to get to the correct data source, and then reference it in your application. In this case, however, the connection string is embedded in the link details and does not use a DSN. To move the list to a different source, you'll need to update the link using the Linked Table Manager.

Adjusting the OpenRecordset Code

You will probably encounter an error like the one shown in Figure 13-18.

www.it-ebooks.info

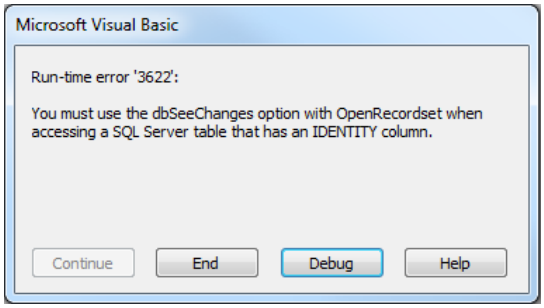


Figure 13-18. *The VBA error with OpenRecordset*

When accessing a table that uses an `IDENTITY` column from ODBC, you need to add the `dbSeeChanges` option. When inserting a record, the primary key is generated by the server. The `dbSeeChanges` option allows the client to see the value that was assigned when the record was saved.

Click the Debug button, which should show you the code that is generating this error. Add two more parameters to the `OpenRecordset` method call like this:

```
Set rs = CurrentDb.OpenRecordset("mnuCommands", dbOpenDynaset, dbSeeChanges)
```

There is one more place where the `OpenRecordset` is called. Search your code for the `OpenRecordset` method and modify it just like this one.

Upsizing with SQL Azure

This example will migrate your Access tables to a SQL Azure database. SQL Azure is a SQL Server 2008 R2 database server in a cloud implementation. Upsizing using SQL Azure has all the benefits of using SQL Server, plus you'll have access to the data from anywhere there is Internet access. You can distribute the frontend application to multiple physically-remote locations, which will all access the same central data store.

■ **Note** SQL Azure is not free and this will require you to choose a rate plan based on your expected usage. At the time of this writing, Microsoft was offering a free 30-day trial to provide an opportunity to test it out. Also, for MSDN subscribers, a free 6-month plan was available. These free plans have usage limits and there are charges for exceeding them. Even though the base plan is free, you are still required to provide a credit card number when you sign-up, in case there are additional charges.

Installing SQL Server Migration Assistant

Microsoft provides a free application that will migrate data from an existing Access application to SQL Server. The Upsizing Wizard that you used previously is a generic tool that will work with almost any version of SQL Server. SSMA, however, is designed for a specific version of SQL Server, and is therefore optimized for that version. This minimizes the use of deprecated methods and takes advantages of the

www.it-ebooks.info

latest SQL Server features. Most importantly for this project, SSMA 2008 has been enhanced to allow direct migration from Access 2010 to a SQL Azure database.

■ **Note** Although SSMA is a free application, Microsoft requires that you register it before it will work. The registration is fairly easy, and I'll explain how to install and register it.

---

1. To download SSMA go to the following link:

```
www.microsoft.com/downloads/en/details.aspx?FamilyID=5abe098d-c7e1-46c6-994a↩
-09a2856eef0b&displaylang=en
```

2. Select the download for `SSMA 2008 for Access 4.2 QFE1.zip`. As I mentioned, each version of the application is tailored to a specific SQL Server version. This page also includes a link for SSMA 2005, which is designed for SQL Server 2005.

3. Open the `.zip` file, which will include an `.exe` file with the same name. Run this executable to start the installation. This will display a welcome dialog box. Click the Next button and the second dialog box requires you to read and accept the End-User License Agreement (EULA).

4. The third dialog box, shown in Figure 13-19, asks you to participate in Microsoft's Customer Experience Improvement Program that gathers data about the installation process. This is optional – you can leave this unchecked if you prefer to not participate.

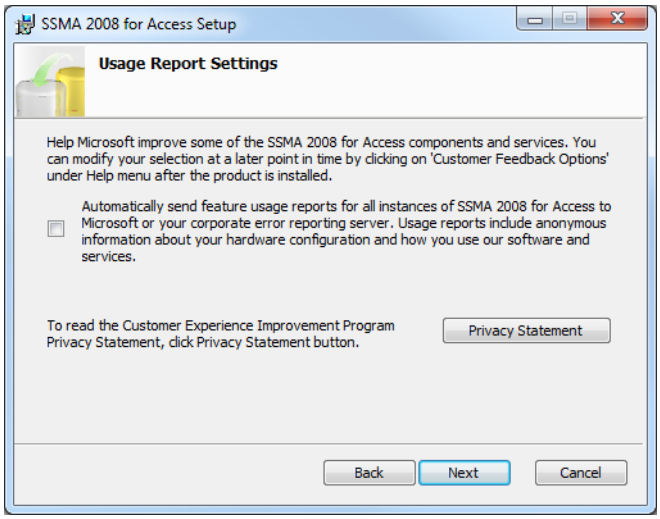


Figure 13-19*. The Customer Experience Improvement Program option*

www.it-ebooks.info

5. In the fourth dialog box, choose the Typical setup type, as shown in Figure 13-20.

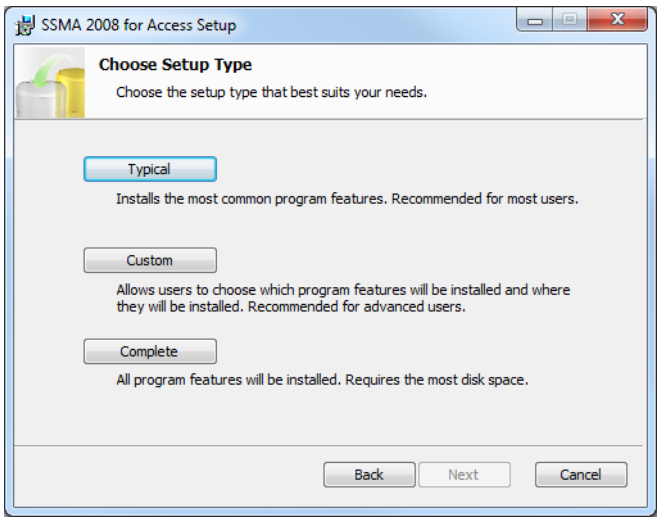


Figure 13-20. *Selecting the Typical setup type*

6. The fifth dialog box lets you know that the wizard has all the information is needs to install the SSMA. Click the Install button to continue. When the install is complete, the dialog box shown in Figure 3-21 will be displayed.

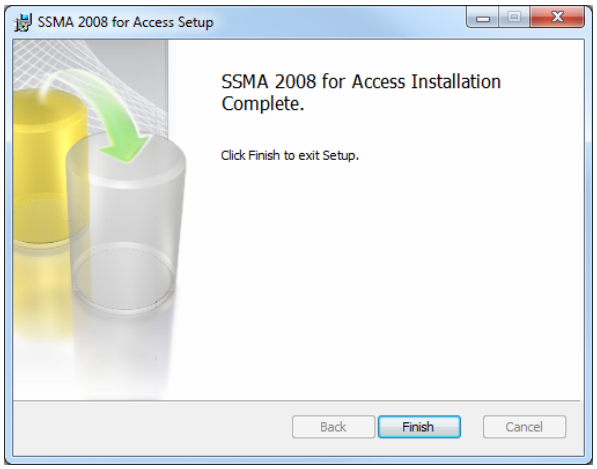


Figure 13-21. *The installation completion dialog box*

www.it-ebooks.info

7. Click the Finish button. The installation program will then check the license key and since there is none, it will display the dialog box shown in Figure 13-22.

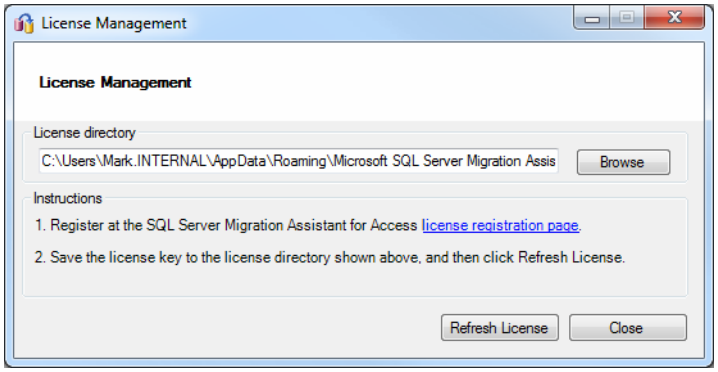


Figure 13-22. *The License Management dialog box*

8. Click the *license registration page* link, which will open the web page where you can fill out the registration form. When the registration is complete, the web site will open a file dialog box when you can choose a location to save the license key. Save this key in the location specified in the License directory field. Then click the Refresh License button. This will look for the license key again and should then display the dialog box shown in Figure 13-23, letting you know that the license key was installed correctly.

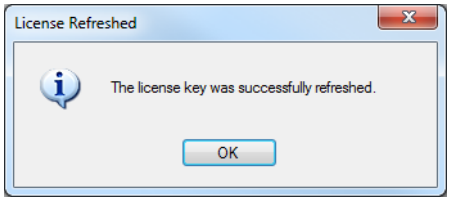


Figure 13-23. *The License Refreshed notification*

Configuring a SQL Azure Database

You'll need to have a Windows Azure account with Microsoft and then create a database server from the Azure portal page. From your online server administration page you can then create a database for the `Library` application.

Setting Up a SQL Azure Database Server

If you don't already have an Azure account, visit `www.microsoft.com/en-us/SQLAzure/database.aspx` to get more information and to sign up. Once your subscription has been activated, go to the SQL Azure

www.it-ebooks.info

portal. Follow the instruction to create a database server. You will specify an administrative username and password, and a server name will be assigned to your account.

The Server Administration page, shown in Figure 13-24, provides the basic configuration, including the server name. Initially, the database server is created with only a `master` database.

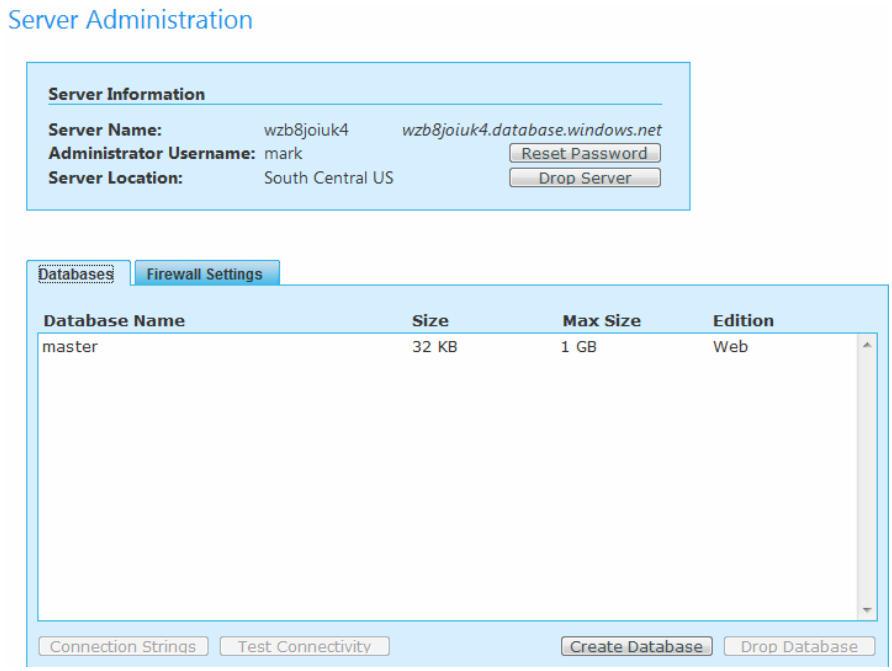


Figure 13-24. *The SQL Azure Server Administration page*

You should not use the `master` database for your applications. Instead, click the Create Database button to create a new database that your Access database will use. In the pop-up window enter **Library** for the database name, as shown in Figure 13-25.

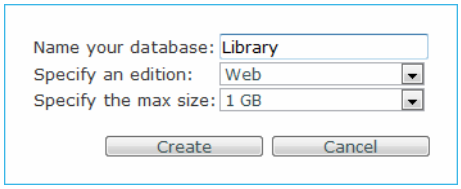


Figure 13-25. *Creating the* `Library` *database*

Once the database server has been setup, you can access it using a local copy of SQL Server Management Studio (SSMS). Enter the server name, including the "database.windows.net" suffix, as shown in Figure 13-26. Enter the administrative username and password.

www.it-ebooks.info

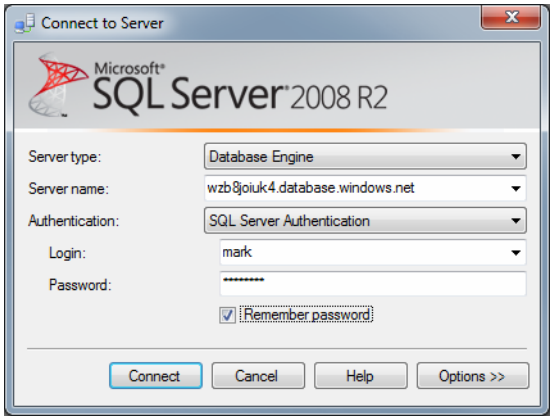


Figure 13-26. *Connecting to SQL Azure from SSMS*

Once you are connected, you can browse the database objects and query the tables just like you would with local databases.

Creating a User Account

The administrative login that was created when you set up the Azure account has full access to all the databases on your server. It is equivalent to the `sa` account. When connecting to the `Library` database from Access, you should create a different account that has only the specific permissions needed by the Access application. Unfortunately, you cannot create users or logons from the Server Administration page. Instead, you'll set these up using Transact-SQL.

From SSMS connect to the `master` database and execute the following script:

```
CREATE LOGIN LibraryUser WITH password='Library!123'
go
CREATE USER LibraryUser FROM LOGIN LibraryUser
go
EXEC sp_addrolemember 'db_datareader', 'LibraryUser'
Go
```

This will create a `LibraryUser` login with the specified password (you can use a different password if you prefer). Logins are always created in the `master` database. You then need to create a user for the login in each database that it will need access to. This script then creates a user for the `LibraryUser` login, giving it read-only access to the `master` database.

Then, create a new query that is connected to the `Library` database and execute this script:

```
CREATE USER LibraryUser FROM LOGIN LibraryUser
go
```


www.it-ebooks.info

```
EXEC sp_addrolemember 'db_datawriter', 'LibraryUser'
Go
```

This script will then add a user for this login in the `Library` database and add it to the `db_datawriter` role.

Migrating the Data to SQL Azure

Now that you have the SSMA installed and a SQL Azure database configured, you're ready to use the SSMA application to migrate your data.

1. The installation program should have added an icon to your desktop. If not, you should be able to find this application in the Start menu. Start it and the first dialog box, shown in Figure 13-27, gives you an overview of the migration process.

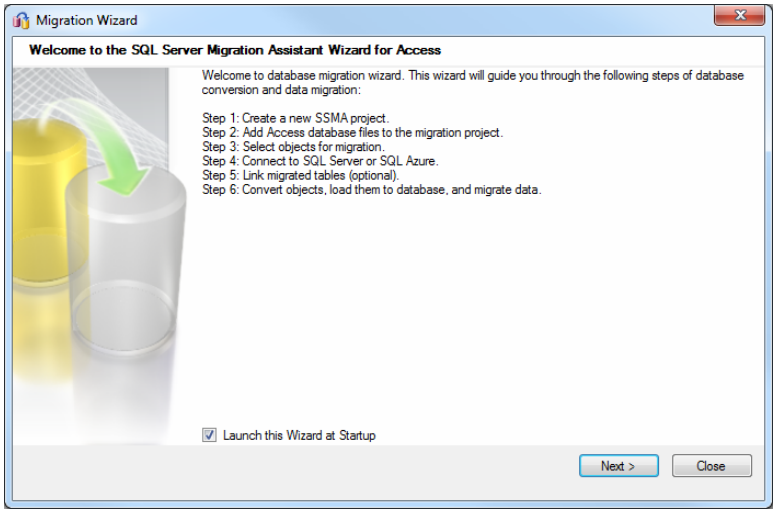


Figure 13-27. *The SSMA welcome dialog box*

2. In the second dialog box you'll set up the SSMA project that will perform the migration. Enter a name and location for the project file. Select SQL Azure as the destination, as shown in Figure 13-28.

www.it-ebooks.info

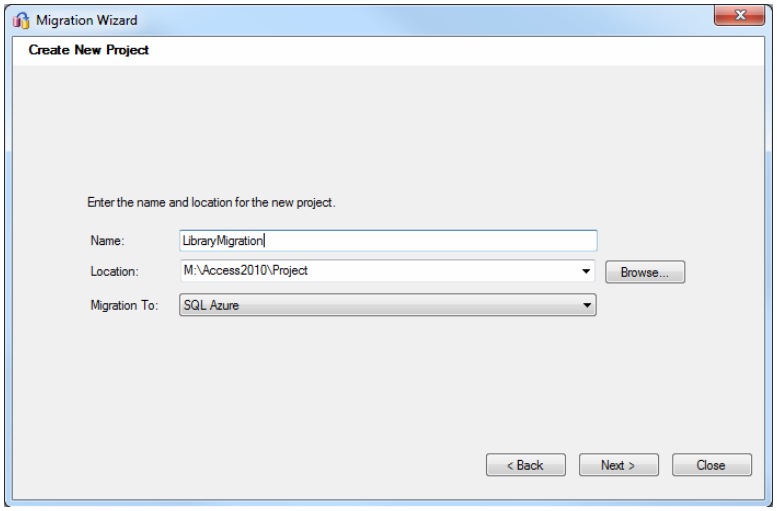


***Figure 13-28**. Setting up the SSMA project*

3. In the third dialog box, you can specify the Access database(s) that should be migrated. You can select more than one. Click the Add Database button and browse to the `Library_Azure.accdb` file that you created in the beginning of the chapter. The dialog box should look like Figure 13-29.

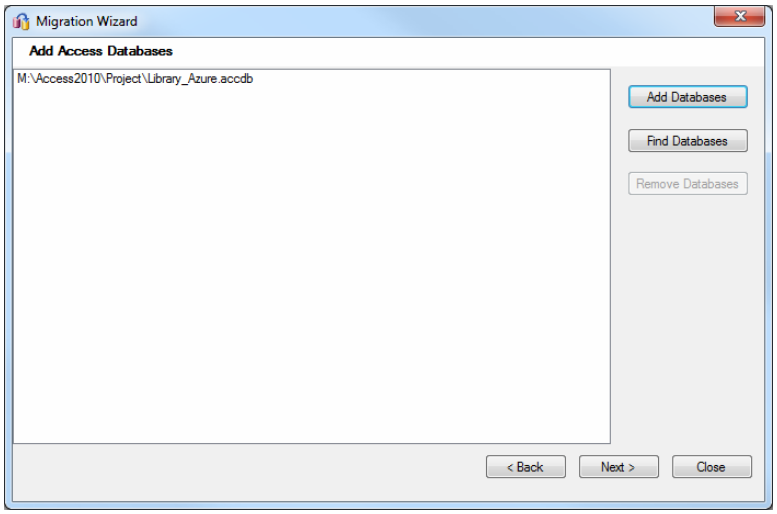


***Figure 13-29**. Adding the `Library_Azure.accdb` database file*

www.it-ebooks.info

4. The fourth dialog box allows you to select the tables and queries that should be migrated. Select all the tables and the following queries, as shown in Figure 13-30:

   - `AllLoans`
   - `CheckedOutItems`
   - `InventoryItemDetail`
   - `LoanDetail`
   - `mnuCommands`

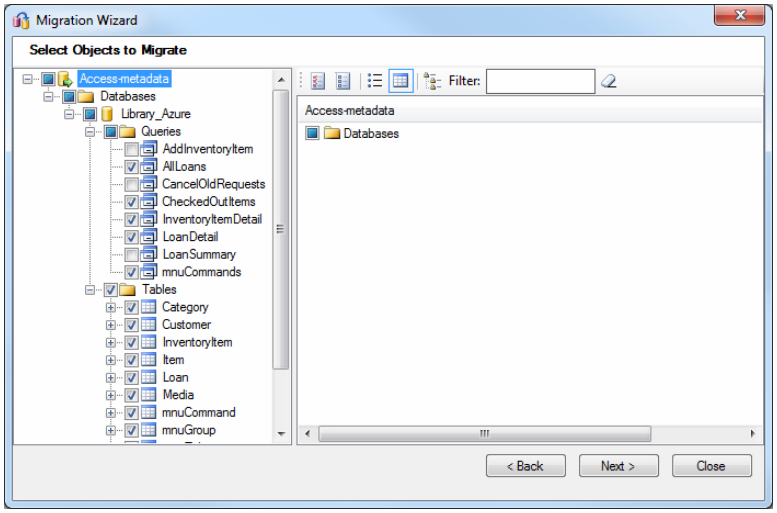


***Figure 13-30**. Selecting the objects to be migrated*

5. In the fifth dialog box, shown in Figure 13-31, enter the SQL Azure details and credentials. Use the administrative login for this because the wizard will need higher-level access to create and configure the tables.

www.it-ebooks.info

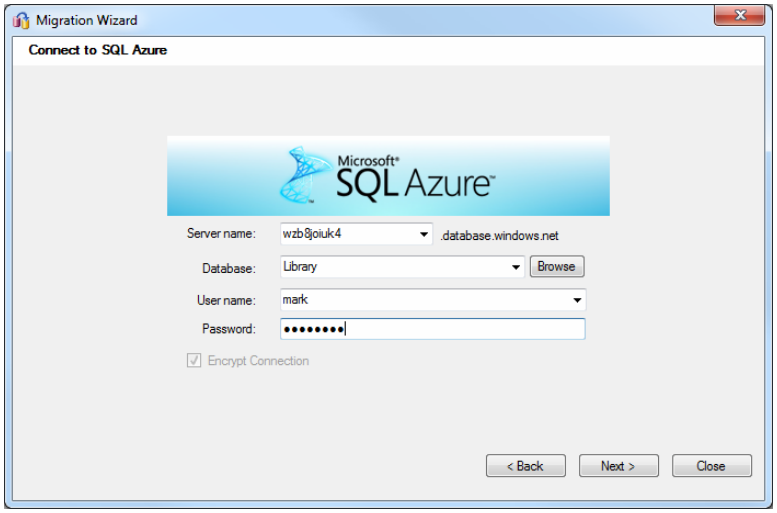


Figure 13-31. Specifying the SQL Azure database details

6. The Migration Wizard is designed to move the data from Access to SQL Server. The sixth dialog box gives you the option to also replace the source tables in the Access database with linked tables. Select the Link Tables check box, as shown in Figure 13-32.

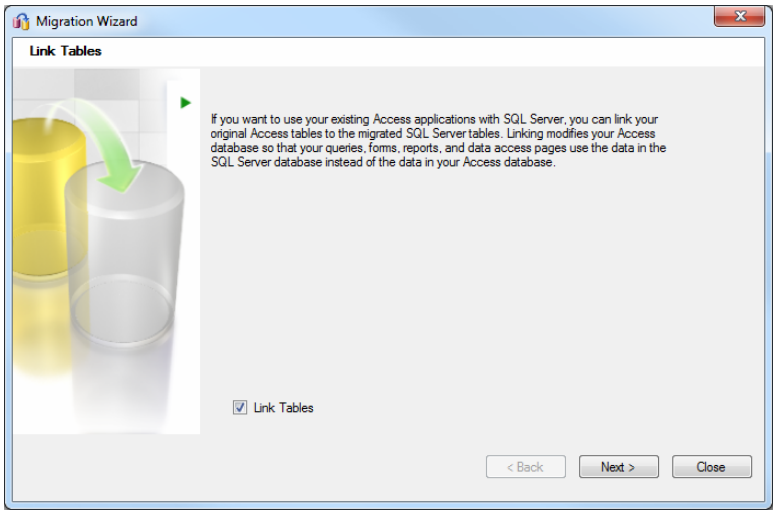


Figure 13-32. Selecting the Link Tables checkbox

7. After you click the Next button, the wizard will run for a while analyzing the source and destination databases, then the report shown in Figure 13-33 is

www.it-ebooks.info

displayed. You'll need to expand the tables and views to see the list of objects that will be migrated. Click the OK button and the wizard will start the data transfer.

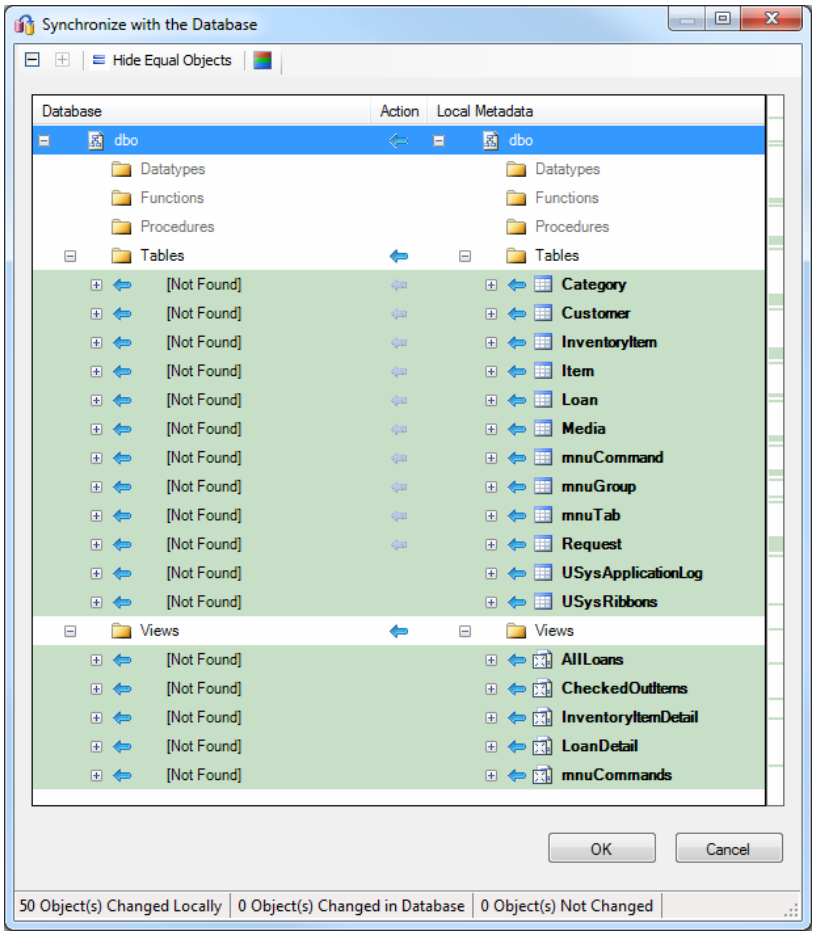


Figure 13-33*. The report of objects to be migrated*

8. Once the objects have been created and all the data copied, the wizard will then start to create the linked tables in the Access database. The dialog box shown in Figure 13-34 will appear, warning you about the password included in the linked tables. Click the Yes button to continue.

www.it-ebooks.info

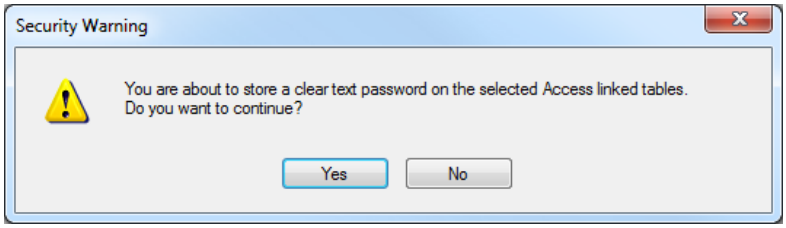


Figure 13-34. *Confirming the clear text password*

■ **Caution** SQL Azure does not support Windows authentication so the connection string will need to include the username and password. A knowledgeable person could read the connection string in the Access file and gain access to the database. Obviously this is not a great solution. I believe Microsoft will address this in future releases. In the meantime, the risk is somewhat mitigated by using a login that has minimal access, so the hacker only has the same access that they had going through the Access application. Also, the firewall restrictions help to reduce the risk as well.

---

9. The SSMA application will once again prompt you for the credentials to the SQL Azure database. This time, however, it's looking for the credentials to use when setup the linked table (even though it doesn't actually tell you that). You'll enter the `LibraryUser` that you setup for this purpose. Before you do, however, you'll need to first give this user access to the tables. When the login was first created, the tables had not yet been migrated. From SSMS, connect a query to the `Library` database and run the script shown in Listing 13-1.

Listing 13-1. *Granting Access to the* `Library` *Tables and Views*

```
GRANT select, insert, update, delete on Category to LibraryUser
GRANT select, insert, update, delete on Customer to LibraryUser
GRANT select, insert, update, delete on InventoryItem to LibraryUser
GRANT select, insert, update, delete on Item to LibraryUser
GRANT select, insert, update on Loan to LibraryUser
GRANT select, insert, update, delete on Media to LibraryUser
GRANT select, insert, update, delete on mnuCommand to LibraryUser
GRANT select, insert, update, delete on mnuGroup to LibraryUser
GRANT select, insert, update, delete on mnuTab to LibraryUser
GRANT select, insert, update, delete on Request to LibraryUser
GRANT select, insert, update, delete on USysApplicationLog to LibraryUser
GRANT select, insert, update, delete on USysRibbons to LibraryUser
go
GRANT select on AllLoans to LibraryUser
GRANT select on CheckedOutItems to LibraryUser
GRANT select on InventoryItemDetail to LibraryUser
GRANT select on LoanDetail to LibraryUser
```


www.it-ebooks.info

```
GRANT select on mnuCommands to LibraryUser
go
```

■ **Note** You will probably get an error when granting access to the `AllLoans` view because it doesn't exist yet. The code generated by SSMA has a syntax error in it. I will explain how to remedy this later. For now, just comment out that line of the script.

10. In the Connect to SQL Azure dialog box, the server name that you were assigned and the `Library` database should already be selected. If not, enter that information. For the user name, enter **LibraryUser@**<server name>. This is just an ODBC oddity; it requires the server name to be included with the user name. Don't include the "database.windows.net" portion however. The completed dialog should look similar to Figure 13-35.

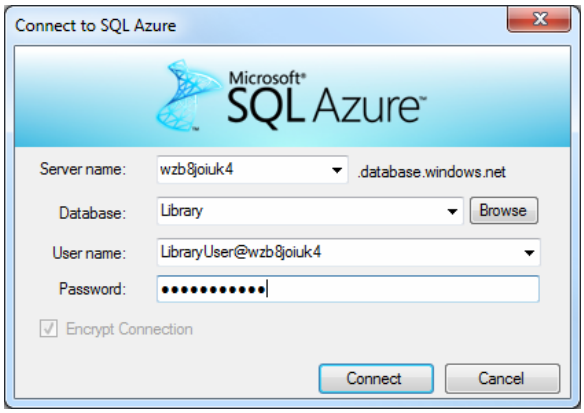


Figure 13-35. *Entering the linked tables credentials*

Viewing the Migration Results

After you have entered the credentials, the wizard will create the linked tables in the Access database.

1. When it has finished, close the wizard and open the `Library_Azure.accdb` file.

■ **Note** You will probably get the same error with the `OpenRecordset` method that I explained previously. Click the Debug button to go to the offending code and add the additional parameters. Correct all other places where `OpenRecordset` is used as well.

www.it-ebooks.info

2. Right-click on one of the linked tables and click the *Linked Table Manager* link. All of the linked tables should be displayed, as shown in Figure 13-36. Notice this is identical to the list of linked tables in the `Library_SQL.accdb` database that you migrated previously. Cancel the Linked Table Manager dialog box.

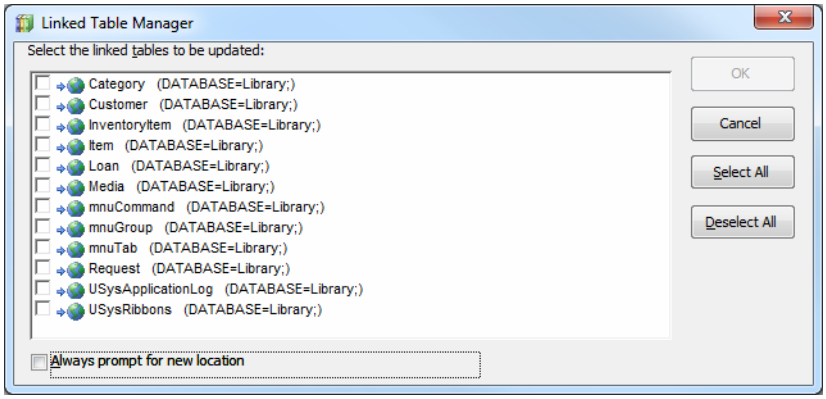


Figure 13-36. *The Linked Table Manager*

3. Hover the mouse over one of the linked tables to view the connection string. It should be similar to Figure 13-37.

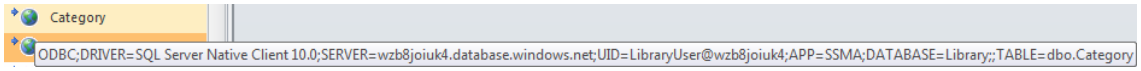


Figure 13-37. *Displaying the connection string*

4. Go back to SSMS and refresh the objects in the Object Explorer. You should be able to see the tables in the `Library` database and execute queries against these tables as demonstrated in Figure 13-38.

www.it-ebooks.info

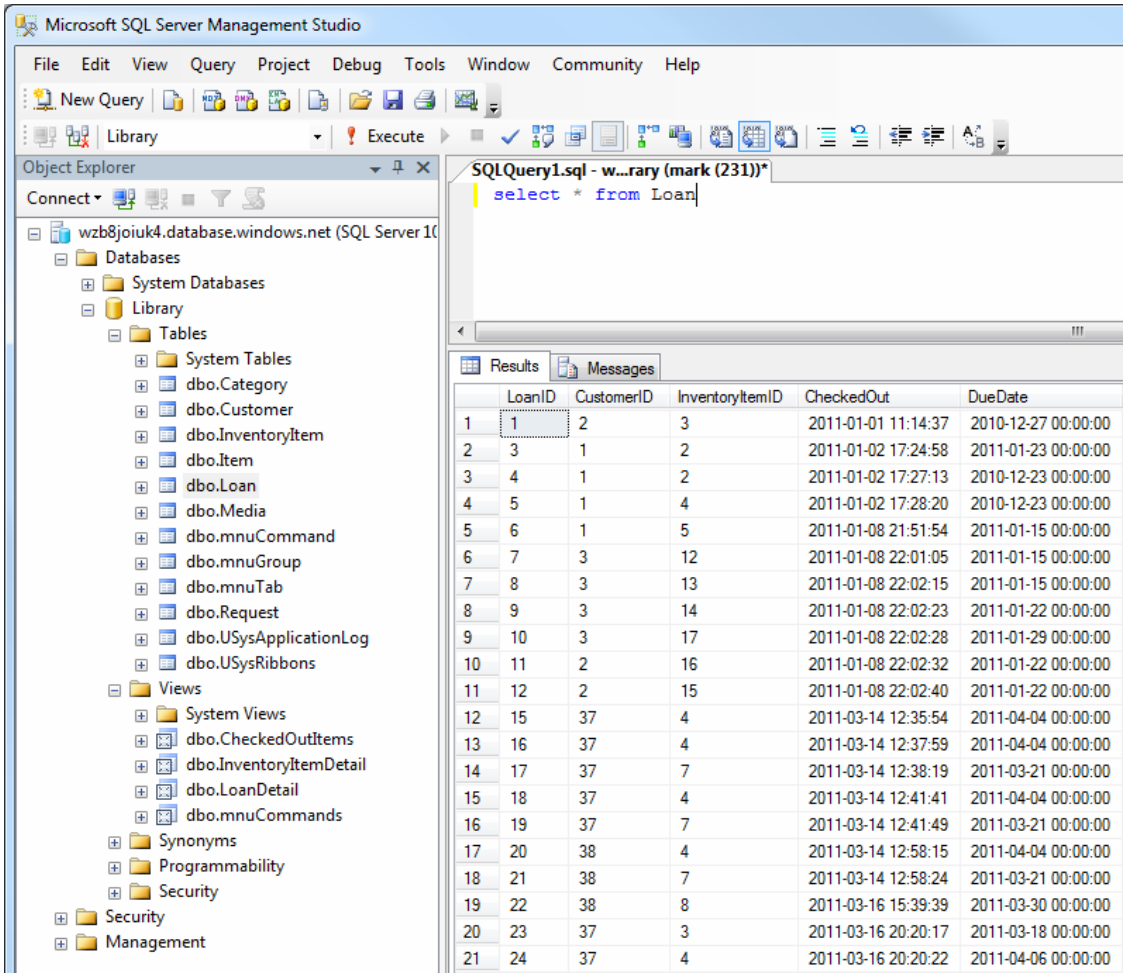


Figure 13-38. *SSMS connected to the SQL Azure database*

Linking the Views

The SSMA created (or at least attempted to create) the specified views in the `Library` SQL Server database. These are all `SELECT` queries and are implemented as a *view* in SQL Server. However, it did not replace the queries in the Access database with linked tables.

To use the view in SQL Server, you'll need to delete the query (or rename it) and then create a linked table. Your Access database will work fine, using a local query. The query will still use the linked tables so you are not using any internal data.

www.it-ebooks.info

However, you should fix the `AllLoans` view in SQL Azure so all the views will be working when you're ready to use them. From SSMS, run the script shown in Listing 13-2 to create the `AllLoans` view. Make sure to use the administrative login and to run the script in the `Library` database.

Listing 13-2. *Creating the* `AllLoans` *View*

```
CREATE VIEW [dbo].[AllLoans]
AS
SELECT
   Loan.LoanID,
   Loan.CheckedOut,
   Loan.DueDate,
   Loan.OverdueFee,
   Item.Title,
   Item.Author,
   Item.ReplacementCost,
   Category.CategoryCode,
   Category.CategoryDescription,
   Media.MediaCode,
   Media.MediaDescription,
   cast(-1 * (CASE
      WHEN ((NOT [CheckedIn] IS NULL AND datediff(dd, [DueDate], [CheckedIn]) > 0)
         OR ([CheckedIn] IS NULL AND [DueDate] < getdate() - 1)) THEN 1
      ELSE 0
   END) AS bit) Overdue
FROM
   (Category
      RIGHT JOIN (Media
      RIGHT JOIN Item
      ON Media.MediaID = Item.[MediaID])
      ON Category.CategoryID = Item.CategoryID)
      RIGHT JOIN (InventoryItem
      RIGHT JOIN Loan
      ON InventoryItem.InventoryItemID = Loan.InventoryItemID)
      ON Item.ItemID = InventoryItem.ItemID
go

GRANT select on AllLoans to LibraryUser
Go
```

In the script generated by SSMA, the calculated field, `Overdue`, had a syntax error when performing the date calculation. The syntax worked fine in Access but not in SQL Server. These are some of the little issues that you'll need to look out for when using external data.

Data Macros with Linked Tables

Along with all the benefits of upsizing that I discussed in this chapter, there is one fairly significant limitation: data macros do not work with linked tables. Because the `Library` application relies heavily on data macros, none of the sample configurations that you built in this chapter will function completely without issues. The effects and resolutions for this limitation are different for Access and SQL Server backend databases.

www.it-ebooks.info

Data Macros in a Split Access Database

When upsizing to an Access backend database, the macro issue is relatively minor. As I showed you in Chapter 3, data macros are part of the table definition. When the tables were moved to the backend database (`Library_Access_be.accdb`) the data macros moved with them. To verify this, open up the backend file, then open the `Loan` table using the Datasheet View. Go to the Table tab of the ribbon and you'll see the Before and After macros are defined. You'll also find that the named macros are there as well.

When a record is added to the `Loan` table, the macros on that table will run as expected. As long as the tables that are being referenced are local tables they will work just like they did before. Because all the tables are in the one backend database, they are all local to the macro so everything is good. If you were to split the database into two Access files, the macros will only work if the tables used by one macro are all in the same database as the table for that macro.

However, you cannot call a named macro in a linked database. The `Library` application includes two UI macros that call named macros (`CalculateLateFees` and `CanceOldRequests`). These will not work as currently implemented. The workaround would be to re-write these features in VBA code.

Data Macros in Linked SQL Server Tables

When using SQL Server as the backend database (or SQL Azure), the data macros were not moved and were effectively deleted. In these environments, none of the before or after macros will run. This includes calculating the due date, determining the overdue fees, changing the status on the `InventoryItem` record, and any of the other data macro logic.

The solution is to implement triggers on the SQL Server tables. You can accomplish the same purpose using triggers. Unfortunately, neither of the upsizing wizards will convert your data macros into SQL Server triggers. You'll need to do that yourself.

Summary

In this chapter you upsized your `Library` database to three different data sources:

- Access 2010
- SQL Server
- SQL Azure

These approaches were increasingly complex but provide different benefits. Using a second Access file as the backend data source is a simple and easy way to upsize your database and it will significantly improve your ability to handle more users.

If you have access to a SQL Server database, you achieve even greater benefits by moving your data there. However, along with these benefits you have more administrative overhead such as scheduling database backups. This is fairly easy to do, but you've moved from a simple desktop Office file (like Excel) to an enterprise-scale application.

Using SQL Azure requires a little more work getting started but provides an easy way to access the data from anywhere. While the Azure platform does much of the administrative work for you, there is still overhead involved. For example, you'll need to update the firewall rules when adding or changing client installations.

Now that you have removed the data from the frontend "database," the frontend application can be distributed to multiple clients. I will show you how to do that in the next chapter.

www.it-ebooks.info

CHAPTER 14

Distributing the Application

In the last chapter, you learned several ways to support more users by splitting the data out of the front-end application. With this approach, you'll have a single data repository that is shared by multiple clients; each user can then have their own copy of the application. Now you'll need an efficient way to distribute the client-portion of your solution to each of your end users.

Microsoft provides a runtime environment that allows you to run your application without needing to install the Office suite. You can open the same database file with both Office and the Access Runtime. The only difference is that runtime disables most of the design features. This is a free application that can be downloaded from the Web. You are also allowed to redistribute the runtime with your application, royalty-free. With the 2010 release, this runtime comes in both 32 and 64-bit versions.

The Office suite also comes with a Package Solution Wizard that you can use to create a Microsoft Installer (`.msi`) file. This makes it easy to package your Access database along with other supporting files. This can then be installed manually by an end user or automatically deployed across the network using Active Directory's group policy facility.

In this chapter, I'll show you how to install and use these utilities to get your solution ready for the masses.

Using the Access Runtime

The Access Runtime is a stripped down version of Access. It will allow you to open and run an existing Access database file. You cannot use it to create a database or make any changes to an existing one. The Navigation pane is removed as well as the standard ribbon. Also, the backstage view only provides print options.

Simulating the Access Runtime

If you have the full version of Access installed, there is no need to install the runtime version. You can open an existing file passing in the `/runtime` command line parameter to simulate how it will work when a user opens it with only the runtime installed. To test this, open a command prompt and run the following command:

```
"C:\Program Files\Microsoft Office\Office14\MSACCESS" /runtime <Your path>\Library.accdb
```

This will launch Access in runtime mode and open the specified file. The application should look like Figure 14-1.

www.it-ebooks.info

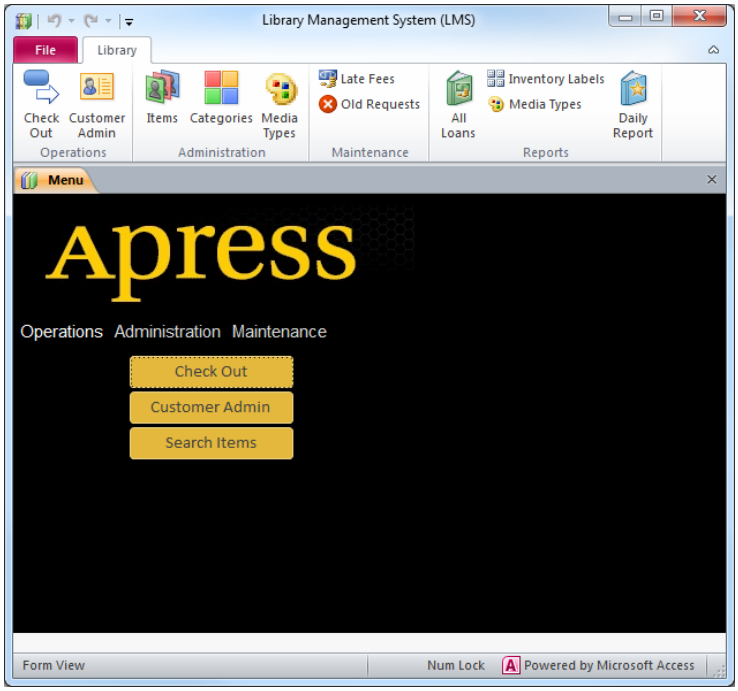


Figure 14-1. *Running the* `Library` *application in runtime mode*

Notice that the custom ribbon that you created in Chapter 10 is displayed, but the standard ribbon is gone. Click the File tab to see the backstage view. The only buttons available here are for printing; all the other features are removed as demonstrated in Figure 14-2.

www.it-ebooks.info

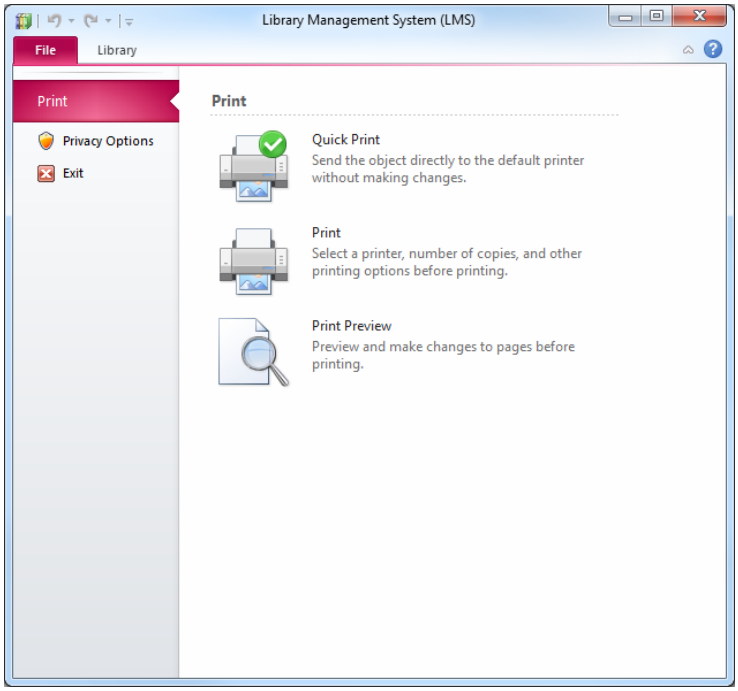


Figure 14-2. *The modified backstage view*

The Privacy Options button that you used in Chapter 10 to re-enable the Navigation and ribbon no longer provides that ability. It simply displays the Trust Center dialog box that provides links about privacy statements and the Customer Experience Improvement Program, as shown in Figure 14-3.

www.it-ebooks.info

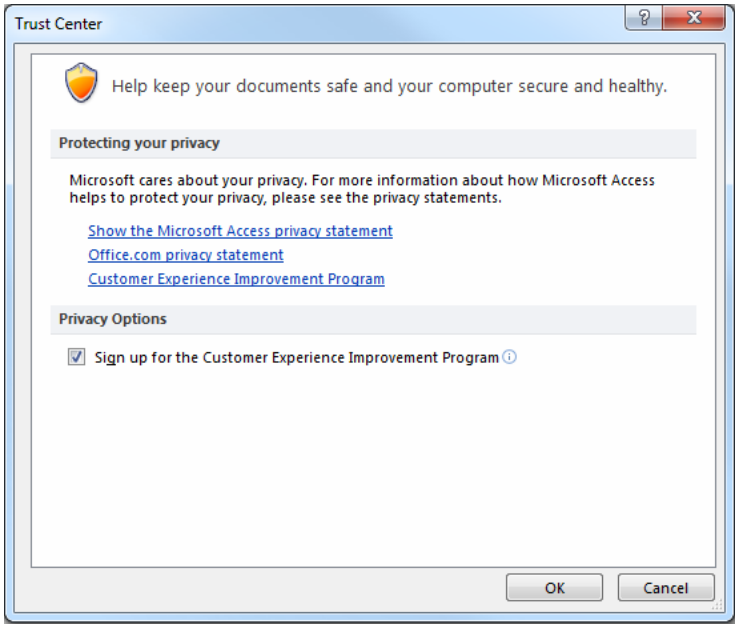


Figure 14-3. *The Trust Center dialog box*

Although all of the editing capabilities are removed, this is perfect for deploying to all of your end users.

Understanding the Filename Extensions

Access uses several filename extensions that are used to indicate the type (or mode) of a database file. The standard extension is `.accdb`, which is the normal database file that you can use and modify. In Chapter 10, I showed you how to compile an executable version that will not allow any of the objects to be modified. This uses an `.accde` extension.

These extensions are a simply a convenient naming convention and do not affect how file is used. For example, you can rename an `.accdb` file to `.accde` and it will still function as an `.accdb` file. To create an executable version you need to generate one through the backstage view. The file extension is helpful, because you can look at the file name and know if it is an executable only version.

There is another useful file extension, `.accdr`. This extension indicates that the file should be opened in runtime mode. You can rename either the `.accdb` or `.accde` extension to `.accdr`. When a file with the `.accdr` extension is opened, Access automatically uses the runtime mode. This is equivalent to passing in the `/runtime` command line parameter.

Try changing the file extension of your `Library.accdb` to `.accdr` and then opening it. It should open in the runtime mode and look just like Figure 14-1.

www.it-ebooks.info

■ **Caution** If you rename a standard .accdb file to .accdr, when you open it, it has the appearance that is has been locked down to prevent end users from modifying it. However, if the user has the full version of Access installed, they can simply change the extension back to .accdb and gain full editing capabilities. You should always create an executable version first (.accde) and rename this to .accdr. Then, even if the user changes the extension, they will not be able to modify the database.

Downloading the Access Runtime

As I mentioned, you won't need to install the runtime, because you already have the full version. However, if you want to redistribute the runtime along with your application, you'll need the installation files. You can download these here:

```
www.microsoft.com/downloads/en/details.aspx?FamilyID=57a350cd-5250-4df6-bfd1↩
-6ced700a6715&displaylang=en
```

There is both a 32-bit and 64-bit version. As with all Office products, you can't mix 32- and 64-bit versions. If you have any of the Office 2010 product installed, such as Outlook for example, you must use the same version of the Access Runtime. The same rule applies to the end users.

Modifying the Application

In the previous chapter, you implemented three different front-end applications, which access the data from three different sources:

- A shared Access backend database
- SQL Express
- SQL Azure

For this exercise, I will use the Access database solution; however, you could use any of these as the steps are the same.

■ **Note** For each solution, you need to ensure that the client application can access the shared database. This will vary, depending on the database engine you're using. For SQL Azure, you'll need to add the IP address of each client to the Azure firewall. For SQL Server, which is using integrated security, you'll need to setup a user in SQL server for each end user with the appropriate permissions. For Access, you'll need to make sure the client can access the file system where the database is located.

www.it-ebooks.info

Supporting Images in a Distributed Environment

The `Library` application uses images that fall into three categories:

- Images included in the Image Gallery such as the Apress banner. These are stored in the database.
- Static images stored in the file system. This includes the `NotFound.tif` that is used when the specified image file cannot be found.
- Dynamic images stored in the file system where the filename is stored in the database.

The images included in the Image Gallery will be obtained from the shared database. For the remaining images the image path, currently, is relative to where the application is loaded. This means that the application will use the local file system. For the static images such as `NotFound.tif`, this is fine. However, for the dynamic images this is not a practical solution. As new items are added to the database, you won't want to copy the image to every client. Instead, you'll need to move these to a shared file server as illustrated in Figure 14-4.

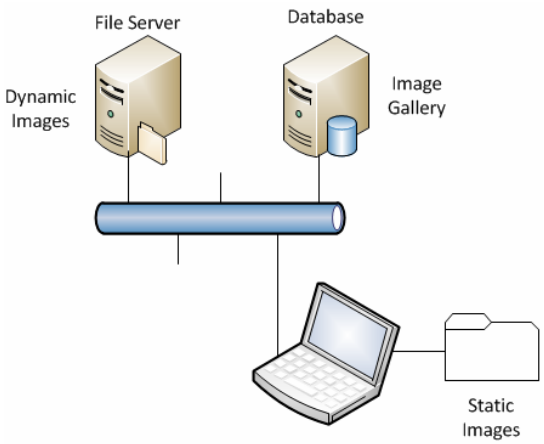


Figure 14-4. *The image locations is a distributed environment*

In preparing to deploy the front-end application, you'll need to make some adjustments. First, as I just explained, the dynamic images should be located on a shared network location. Secondly, the backend database should also be on a shared network location. I recommend creating a mapped drive to a location on the network where both of these will be stored. Each client will have to create the mapped drive, which requires an extra, manual step during installation. However, should you need to change the location later, you just have to re-map the drive to the new location without making any changes to the application.

www.it-ebooks.info

■ **Note** Instead of using a mapped drive, you could simply reference the files using a UNC filename such as `\\server1\folder\subfolder`. This would eliminate the need to create a mapped drive on each client. Should the folder change, you would need to modify the application. Also, if mapped drives are not allowed on your network, you will need to use UNC.

1. Copy the `NotFound.tif` file from the `Images\Static` folder to the root folder where the `Library_Access.accdb` file is located.
2. Create a folder on a network location and copy the backend database (`Library_Access_be.accdb`) there.
3. Create a subfolder named **Images** and copy all the files from your local `Images` folder.
4. Create a network share to this folder.
5. Create a mapped drive to this location. I used the drive letter L:, but you can use a different drive letter if you prefer.

Modifying the Application

First, you'll modify the VBA code to use the mapped drive when loading images. Then you will need to re-link the linked tables to use the backend database in the mapped drive.

Open the `Library_Access.accdb` file. In the Navigation pane, double-click the `Main` object, which will display the VBA Editor. There is currently a function named `GetFullImagePath`. Replace this with the functions shown in Listing 14-1.

Listing 14-1. *Modified Image Functions*

```
Function GetFullImagePath(ByVal sImage As String) As String

    GetFullImagePath = "L:\Images\" & sImage

End Function

Function GetStaticImagePath(ByVal sImage As String) As String

Dim sPath As String
Dim n As Integer

    sPath = CurrentProject.FullName
    n = InStrRev(sPath, "\", Len(sPath))
    sPath = Left(sPath, n)

    GetStaticImagePath = sPath & sImage
End Function
```


www.it-ebooks.info

For the `GetFullImagePath` function, the new implementation will simply append the image name to the hardcoded path (`L:\Images`). The new function, `GetStaticImagePath`, uses the path where the application is run from and appends the image name. You will use this method to get the `NotFound.tif` file.

In the `Form_Item` code file, change the error logic in the `DisplayImage` method to use the new function. The code will be changed from:

```
imgPicture.Picture = GetFullImagePath("Static\NotFound.tif")
```

to

```
imgPicture.Picture = GetStaticImagePath("NotFound.tif")
```

Save the code changes and then test the application. Try displaying an item that has an image and one that does not to verify both the dynamic and static images are working correctly.

Relinking the Linked Tables

Now you'll modify the linked tables to use the backend database in the shared folder.

1. In the Navigation pane, right-click any of the linked tables and click the Linked Table Manager link.
2. In the Linked Table Manager dialog box, click the Select All button and select the "Always prompt for new location" check box, as shown in Figure 14-5. Then click the OK button.

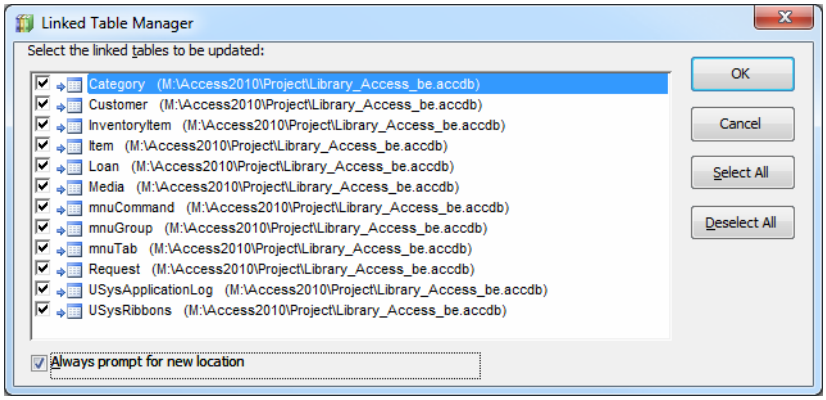


Figure 14-5. *Relinking all tables*

3. A file dialog box will then appear for you to select the location of the linked file for the `Category` table. Select the `Library_Access_be.accdb` file in the shared folder, as shown in Figure 14-6.

www.it-ebooks.info

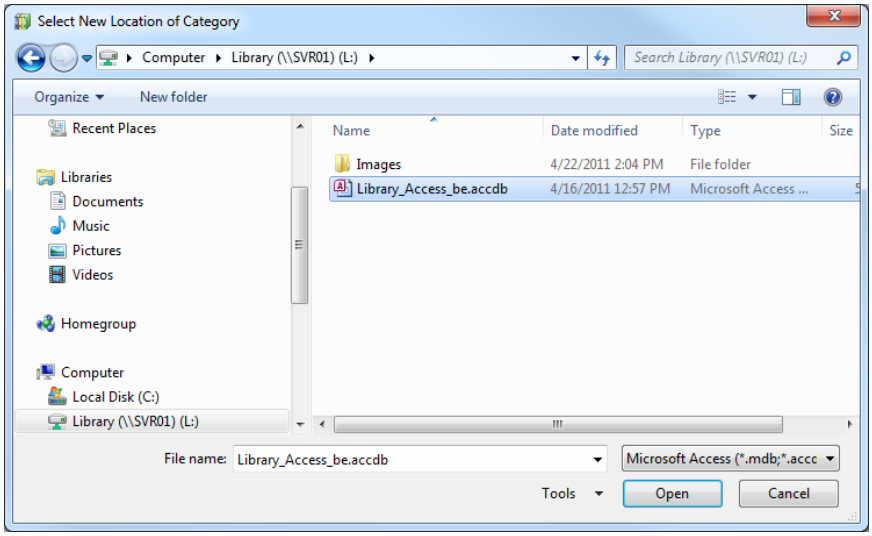


Figure 14-6. *Selecting the linked file location*

Because all the other linked tables use the same linked file, the Linked Table Manager does not need to prompt for the remaining tables. You should see the dialog box shown in Figure 14-7, to let you know that all tables have been re-linked.

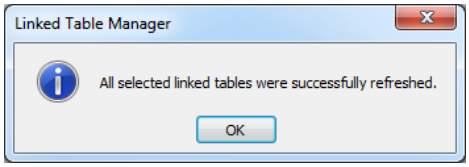


Figure 14-7. *All linked tables updated*

Creating an Executable File

As I explained in Chapter 10, when deploying an Access database, you should first compile an executable-only file. This will prevent anyone from accessing the source code or modifying and of the objects such as tables or forms. In the next section you'll create an installation file using a wizard, which will change the extension to .accdr to force the runtime mode. If you don't first create a compiled version, a knowledgeable user with a full version of Office installed can simply change the file extension and have access to all of your code.

1. Go to the backstage view and select the Save Database As tab. Then select the Make ACCDE option and click the Save As button.

2. You will then be prompted to select a filename and location for the .accde file. Accept the default value of `Library_Access.accde` in the same location as the `Library_Access.accdb` file.

www.it-ebooks.info

■ **Caution** An `.accde` file, it is generated for a specific bit version. If you compile it with a 64-bit version of Access, it will only run on a client that also has 64-bit version of Access or the Access Runtime. If you have both 32- and 64-bit clients you'll need to build two separate `.accde` files. To do that, you'll need both versions of Access, which must be installed on different machines.

3. Close the Access application. With these changes in place, the `Library_Access.accde` file is now ready to be deployed to client locations.

Creating an Installation Package

To simplify the deployment of the front-end application to multiple users, you should create an installation package that will install the application along with any additional files that may be required. This package can also install the Access Runtime, if appropriate.

Installing the Package Solution Wizard

Microsoft Office provides a Package Solution Wizard that makes it easy to generate an installation package (`.msi` file) from your Access file. However, this is not included when you install Office with the Typical installation options. To add this, you'll need to run the Office setup program and add this feature.

4. From the control panel, select the Programs and Features option. Find the Microsoft Office program, right-click it, and click the *Change* link, as shown in Figure 4-8. Depending on the version of Office that you have installed, the title of the program could be a different.

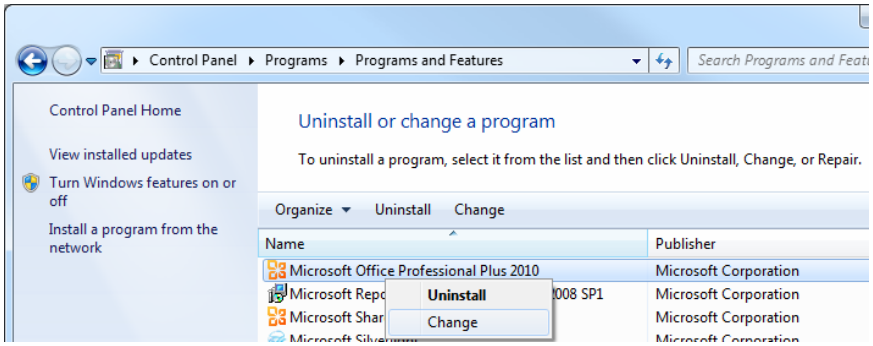


Figure 14-8. *Starting the Office setup program*

5. This will launch the Office setup application. Select the Add or Remove Features option, as shown in Figure 4-9.

www.it-ebooks.info



Figure 14-9*. Selecting the Add or Remove Features option*

6. Drill down into Microsoft Access and then Add-ins. Click the dropdown icon next to the Package Wizard feature and click the *Run from My Computer* link, as shown in Figure 4-10.

www.it-ebooks.info

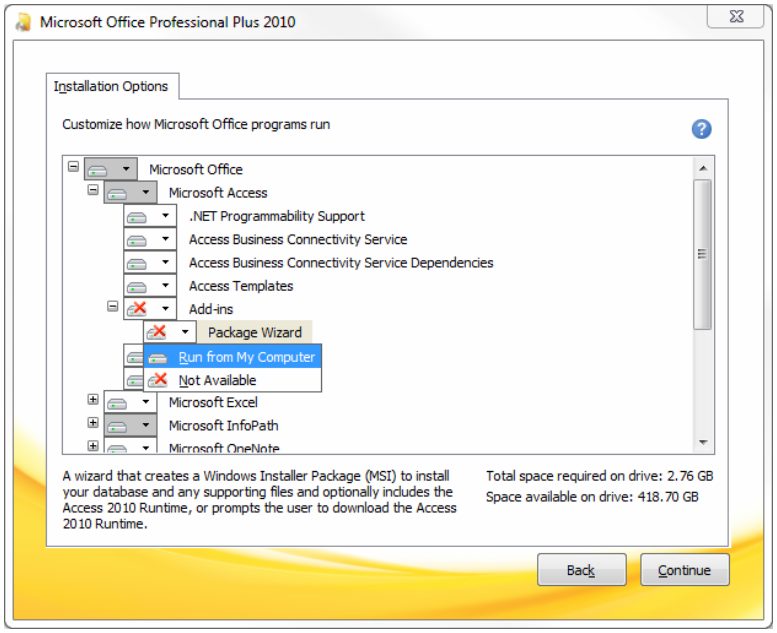


Figure 14-10. *Adding the Package Wizard feature*

7. Click the Continue button, and the setup program will run for a while, installing the additional feature.

Using the Package Solution Wizard

Now that you have this feature installed, you can run it from the Access backstage view. The Package Solution Wizard will gather details from a series of dialog boxes and then generate the files needed to create an install disk.

8. Open the `Library_Access.accdb` file, go to the Backstage view, and select the Save & Publish tab. You should now have a Package Solution tab. Select this and then click the Package Solution button, as shown in Figure 4-11.

www.it-ebooks.info

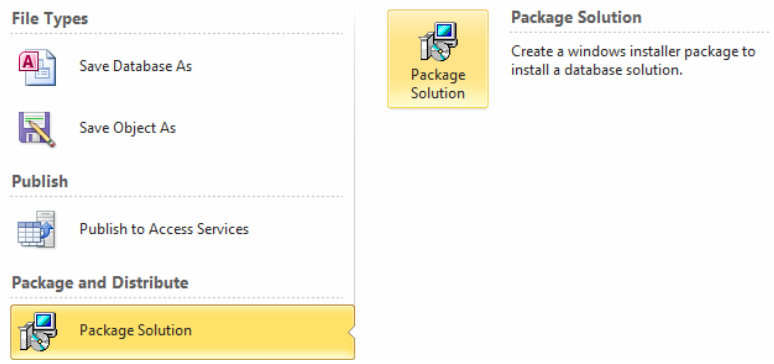


Figure 14-11. *Selecting the Package Solution wizard*

9. This will launch the Package Solution wizard that will take your through four dialog boxes to collect the installation options. In the first dialog box, shown in Figure 4-12, you need to specify a folder where the installation files should be generated.

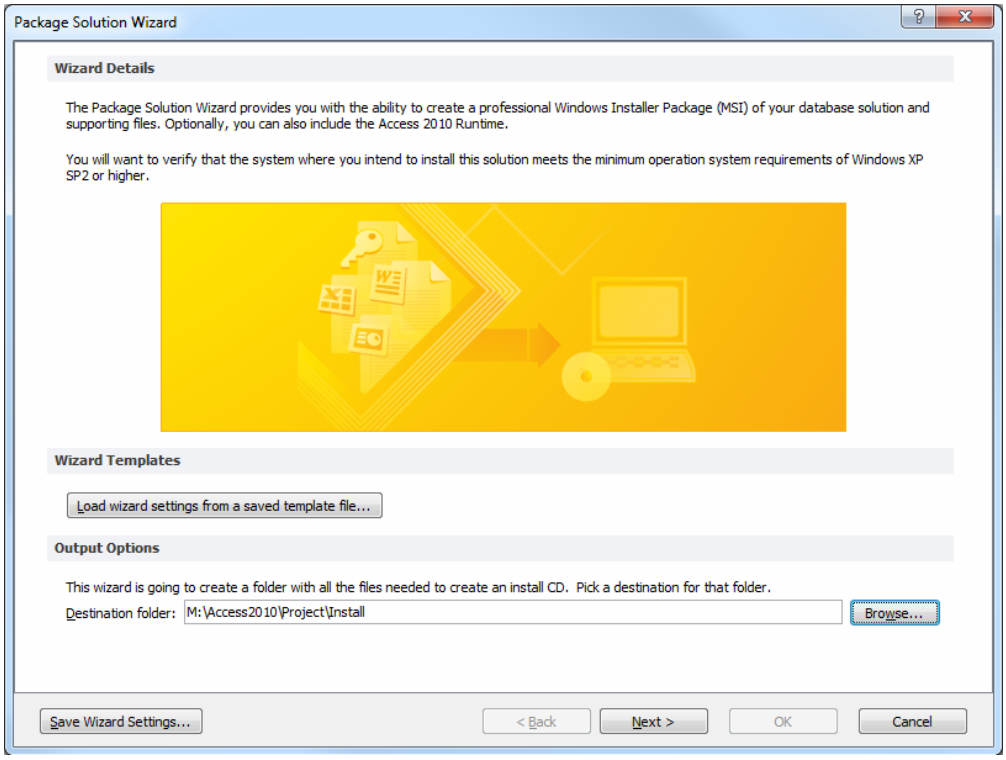


Figure 14-12. *The Package Solution sizard, Dialog Box*

www.it-ebooks.info

■ **Tip** Alternatively, you could load a template file that defines the values for all four dialog boxes. When you complete the wizard, you will be prompted to save the settings to a wizard template file with an `.adepsws` file extension. This is really helpful if you want to rebuild the installation package. For example, if you make some modifications to the application you'll need to rebuild the installation file. In this case, just open the previous template file and you won't have to reenter these settings.

10. In the second dialog box, you'll need to specify the Access file that you want installed. You can use the Browse button to look for the file to use. However, it will not show files with the `.accde` extension. You can browse to the `.accdb` file and then change the extension. Enter the `Library_Access.accde` that you compiled earlier, as shown in Figure 4-13.

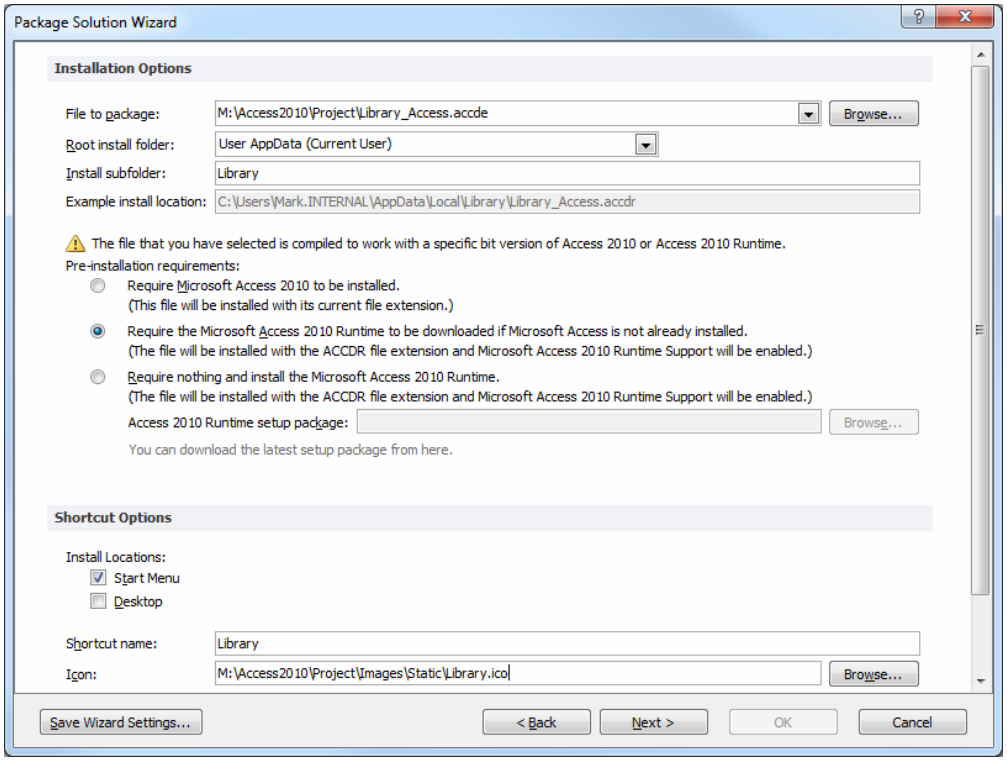


Figure 14-13. *The Package Solution wizard, Dialog Box*

www.it-ebooks.info

■ **Note** Notice the warning about the specific bit version. Make sure that you have compiled the `.accde` file with the same version of Access that the client will use.

11. You'll also need to select the root install folder. The default option is to place this in the current user's `AppData` folder. This is probably the most logical place to install the file; however, there are several other choices available such as My Documents, Desktop, and Program Files. You can also choose the System Drive, which will put it on the local C: drive. You use the Install subfolder field to specify the subfolder to use under the selected root folder. The dialog will show you the complete file path based on your selection of these options. Select the User AppData root folder and Enter **Library** for the subfolder.

12. For the Pre-installation requirements, you have three options that are explained fairly well in the dialog box. If you choose to distribute the runtime in this installation package, you'll need to download its installation file and specify the location of the installation file in Package Solution wizard. You can use the link provided earlier in this chapter or click the *You can download the latest setup package from here* link. Just choose the Save option instead of Run and save the setup package to a file.

13. You can choose to create a shortcut in the Start Menu and/or the Desktop. Enter **Library** for the Shortcut name and select an icon for the shortcut. If you have downloaded my `Images` folder from `www.apress.com` there should be a `library.ico` file that you can use.

14. In the third dialog box, shown in Figure 14-14, you can specify additional files or registry entries that will be installed with the application.

www.it-ebooks.info

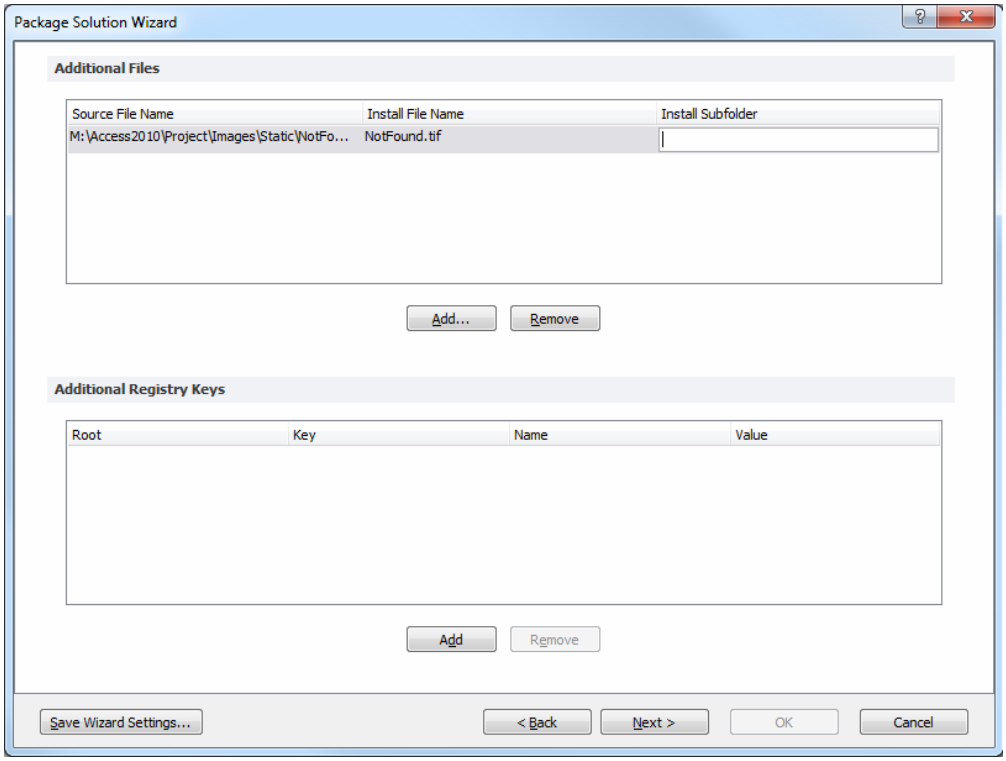


Figure 14-14*. The Package Solution wizard, Dialog Box*

15. To add a file, just click the Add button and browse to the file that you want to include. You can select multiple files as well and all the selected files will be added. For each file you can enter a subfolder that the file should be placed in. If you leave this blank, the file will added in the same folder as the Access database.

■ **Note** At the time of this writing, the subfolder option did not work. If you entered a subfolder name, the wizard would crash when generating the installation file. This is a known issue and Microsoft has a hotfix to address it but the hotfix only applies to the 32-bit version of Access. There was no solution for 64-bit users. For more information, see the article at `http://answers.microsoft.com/en-us/office/forum/office_2010-access/access-2010-package-solution-bug-cant-create/d009265c-6e87-432a-8515-9265b372836f`. `Because of this bug, the Library_Access` database was coded to get the static images from the current folder instead of an `Images` subfolder.

---

www.it-ebooks.info

16. Add the `NotFound.tif` file that is in the `Images\Static` folder and leave the subfolder field blank. The fourth and final dialog box, which is shown in Figure 14-15, allows you to provide details that are displayed during the installation process or afterward in the control panel.

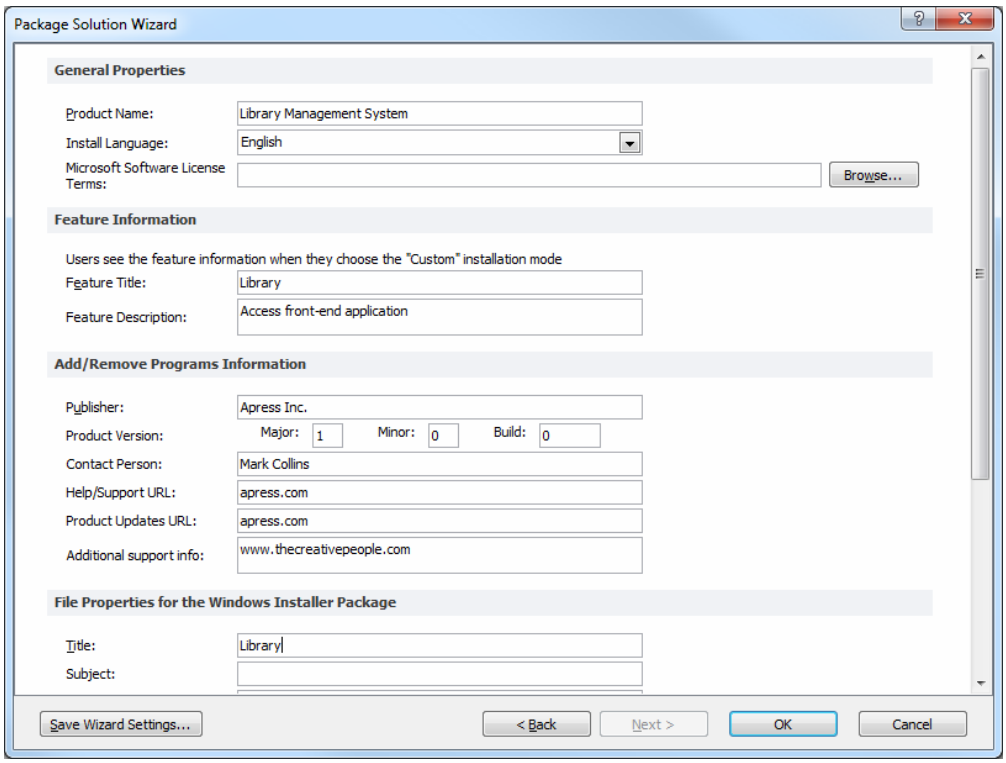


Figure 14-15. The Package Solution wizard, Dialog Box

17. The information in this dialog box is mostly informational and you can fill in whatever values you want to use. Enter **Library** for the Title of the Windows Installer Package.

18. When you have finished, click the OK button. The wizard will then prompt you to save these settings in a wizard template file, as shown in Figure 14-16. Click the Yes button and then select a location to save this file. As I said earlier, this will save you some time when you need to rebuild the installation file.

www.it-ebooks.info

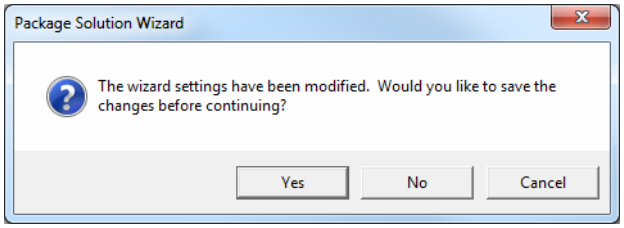


Figure 14-16. *Saving the the wizard template file*

Deploying the Installation File

The wizard will run for a few seconds and then open Windows Explorer to the installation folder that was created. This was generated to be copied to an installation disk. The root folder contains an `autorun.inf` file and a `setup.exe` executable. The `Files` folder contains the actual `.msi` file.

If you want to create an installation disk, just copy the entire folder and subfolders to a disk. When inserted, the setup program should start automatically depending on how the autorun options are configured on the client.

In most cases, however, you'll probably install this from a network location. In this case, all you'll need is the `Library_Access.msi` file. Copy this to the L: drive or wherever you created the shared folder that contains the backend database and images.

To test the installation process, you can run it directly on your workstation. For a better test:

1. Install it using the shared location from another client, preferable one that does not have Office installed. If you try to install this on a client that does not have Access installed, you should get the error shown in Figure 14-17.

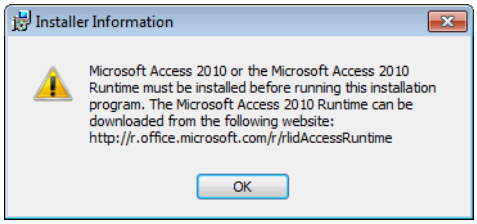


Figure 14-17. *Error diaplyed when Access is not installed*

2. You can then download and install the Access Runtime and re-start the installation. The Welcome dialog is shown in Figure 14-18.

www.it-ebooks.info

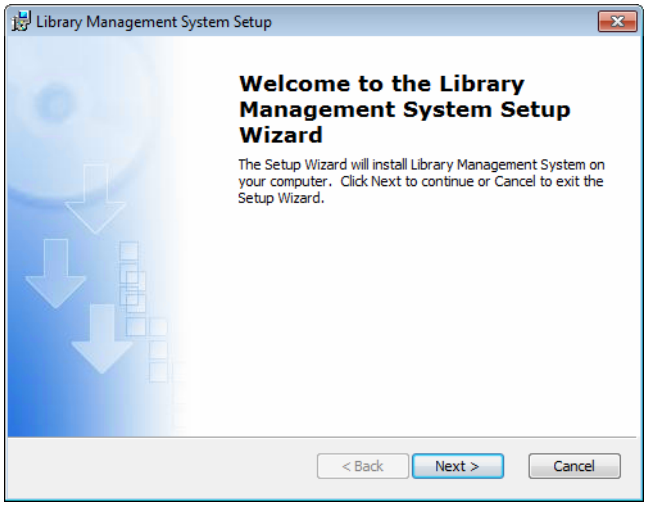


Figure 14-18. *The intallation package welcome dialog*

3. The second dialog, shown in Figure 14-19, collects the user's information.

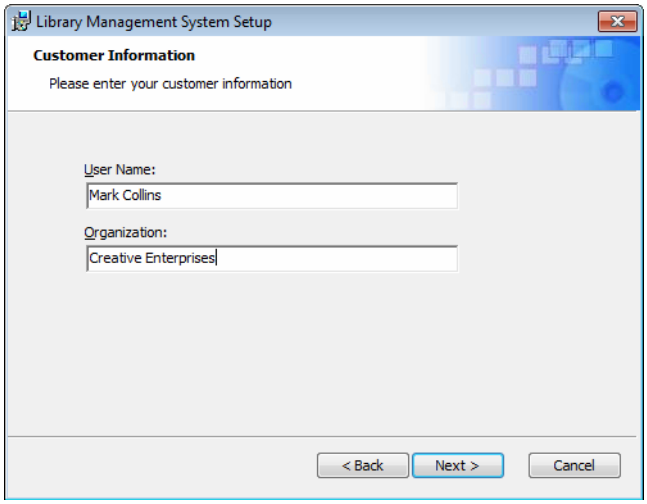


Figure 14-19. *Entering the user information*

4. The third dialog, shown in Figure 14-20, gives the user a choice of using the Typical or Custom installation. This package only contains a single feature, so there's no benefit to using the Custom option. Click the Typical button to continue.

www.it-ebooks.info

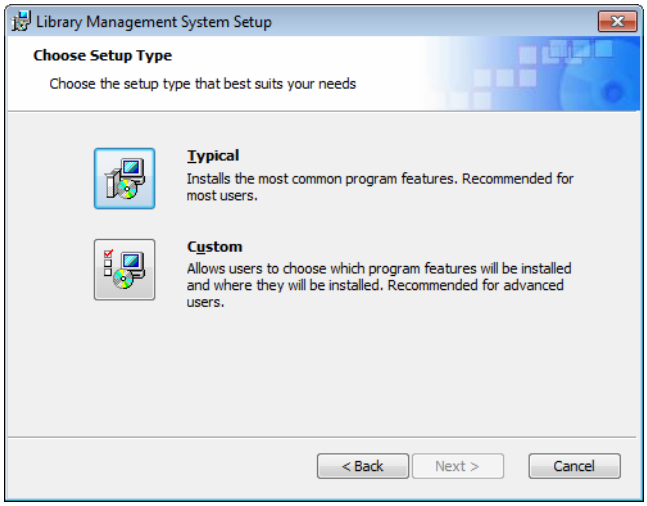


Figure 14-20. *Selecting the Typical installation option*

5. In the fourth dialog box, shown in Figure 14-21, click the Install button to start the installation process.

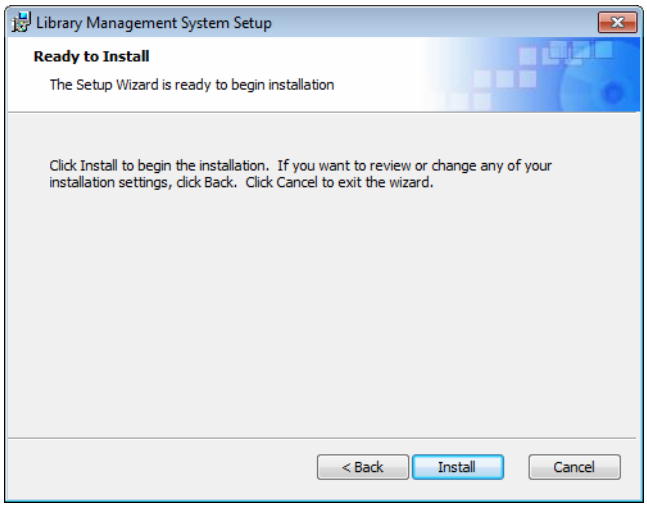


Figure 14-21. *Ready to install dialog box*

After a few seconds, the final dialog box, shown in Figure 14-22, shows that the installation has completed successfully.

www.it-ebooks.info

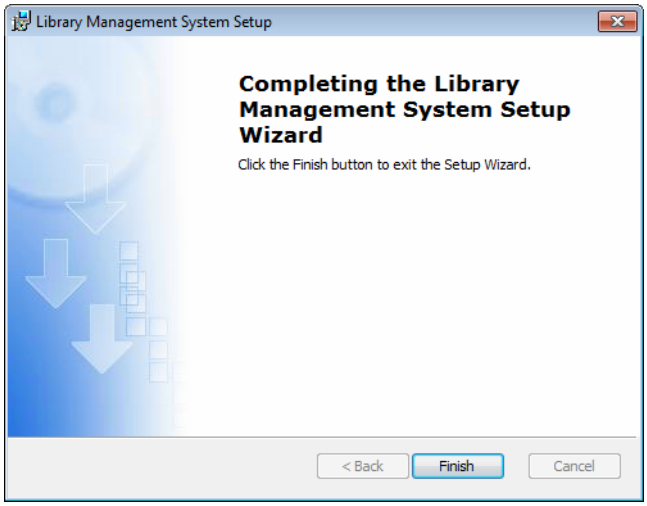


Figure 14-22. *The final installation dialog box*

Now go to the Start Menu and you should see the `Library` application. Start the application and verify that it is working. Try searching for items to verify that it is able to display the images correctly.

Summary

In this chapter, you prepared the front-end application for deployment to multiple clients. You modified it to use a network location for both the dynamic images and the backend database. You then built an installation file using the Package Solution Wizard. Now multiple uses can run the installation file on their local machine and access the shared database and dynamic images from a network location.

You used the `Library_Access` application, which uses Access for the backend database. This solution works just as well for the SQL Server options that you created in Chapter 13. The client machines do not need to have Office installed to use your application. Instead the Access Runtime can be used, which can be downloaded and distributed royalty-free.

The key concepts covered in this chapter include:

- Using the runtime mode in Access by specifying the `/runtime` command line parameter or changing the filename extension to `.accdr`.
- Installing and using the Package Solution Wizard to build an `.msi` installation file.
- Moving images from a local file to a shared location on the network.

In the next chapter, you will publish your Access database to a SharePoint site, which will accomplish both upsizing (moving the data to central data repository) as well as provide browser-based client access to the data.

www.it-ebooks.info

CHAPTER 15

Publishing to the Web

In Chapter 13, I showed you three ways to upsize your Access application by moving the data to a shared repository. This allowed you to distribute the front-end application to multiple clients. It enabled you to support far more clients than with a single database solution. In this chapter, I'll show you a fourth way to upsize your application by moving the data to SharePoint. You can also move portions of the application to SharePoint as well, giving you a browser-based solution.

Access 2010 provides a facility to publish your Access database to a SharePoint site. This can be used to implement a web application with very little effort. Conceptually, this works much the same way as the other solutions you used in Chapter 13. For each Access table, the publishing process creates a SharePoint list and copies the data to that list. The Access table is then replaced with a special type of linked table, called a *web table*. Once published, the Access front-end works like it used to, except the data resides in a collection of SharePoint lists. An Access front-end application that has been published to SharePoint is often referred to as a *web database*.

There are two significant differences with this method of upsizing, however. The first is that you can also publish forms and reports to SharePoint, as well. These need to be specially designed for the Web, and are called *web forms* and *web reports*. Your existing forms and reports are referred to as *client forms*, and are not compatible with SharePoint, but can still be used from the Access client. Web forms and reports can be used from either SharePoint or the Access client.

The second difference is that data macros in a web database are executed by SharePoint and not Access. This can cause some conversion challenges, because they don't always work exactly the same in both environments. It is important that these macros run in SharePoint because, with web forms, the SharePoint lists can be updated without using the Access client. To ensure data integrity, the macros need to run regardless of where the change was initiated.

■ **Caution** The publishing process is not quite as seamless as it could be and there are a few issues that you'll need to work through. I will explain these in this chapter. The most significant are:

- Data macros will need some adjusting
- VBA code on the client forms is often deleted
- Default values to `datetime` fields not converted correctly

---

www.it-ebooks.info

Preparing to Publish

There are some restrictions on the tables that can be published to SharePoint. Before publishing, you'll need to run the Compatibility Checker, which will verify that your tables are compatible. You will also need to make some adjustments to the data macros so they will work correctly in SharePoint.

■ **Note** At the time of this writing, these adjustments were necessary. It's possible that future releases will resolve these differences so that these changes are not necessary. The modified macros will also work with the Access client.

Make a copy of the `Library.accdb` that you ended up with in Chapter 12 and name the copy **Library_SharePoint.accdb**. You will publish this database to a SharePoint site.

Updating the Data Macros

The major limitation with running data macros from SharePoint is that you cannot use nested `LookupRecord` actions. For example, the `Loan.BeforeChange` macro looks up the `InventoryItem` record. Within that data block, it then looks up the `Item` table and, within that block, it looks up the `Media` table. You can accomplish the same thing by using local variables to hold intermediate results.

Open the `Library_SharePoint.accdb` file and then open the `Loan` table using the Datasheet View. From the Table tab of the ribbon, click the Before Change button. This will display the current implementation of this macro.

■ **Tip** The macro editor is great for entering new actions, but I've found is somewhat difficult to move things around. As the changes to this macro are fairly significant, it is probably easiest to delete the current implementation and start over. In the macro editor, press Ctrl-A to select all of the actions and then press the Delete key.

This macro is fairly long so I will describe it in sections. I will show you what actions to enter with minimal explanation. Refer to Chapter 3 if you have questions about data macros in general.

1. Define the local variables and initialize them, as shown in Figure 15-1.

www.it-ebooks.info

```
/*  Initialize the local variables
SetLocalVar
        Name  itemID
  Expression  = 0
SetLocalVar
        Name  mediaID
  Expression  = 0
SetLocalVar
        Name  loanPeriod
  Expression  = 0
SetLocalVar
        Name  overdueFee
  Expression  = 0
SetLocalVar
        Name  renewalsAllowed
  Expression  = 0
SetLocalVar
        Name  inventoryStatus
  Expression  = ""
```

Figure 15-1*. The Before Change macro – Part 1*

2. Look up records in the `InventoryItem`, `Item` and `Media` tables. Store the columns in the local variables ,as shown in Figure 15-2.

www.it-ebooks.info



Figure 15-2. *The Before Change macro – Part 2*

3. Now you have all the data elements that you'll need in local variables and you can implement the business rules. The first one, shown in Figure 15-3, ensures that the item is available to be checked out. If it is on hold, this logic makes sure that it being held for this customer. If the item should not be checked out, the macro raises an error, which will stop the update.

www.it-ebooks.info

```
/*   Make sure they don't check out an item that is not available                '
 If   [IsInsert]   Then
    If   [inventoryStatus]<>"Available" And [inventoryStatus]<>"On Hold"   Then
       Group:   Raise an error and stop the macro
          RaiseError
                Error Number   -1
             Error Description   This item cannot be checked out
          StopMacro
       End Group
    Else If   [inventoryStatus]="On Hold"   Then
      /*   If the item is on hold, make sure it is for this customer                */
       Look Up A Record In   Request
             Where Condition   = [Request].[InventoryItemID]=[Loan].[InventoryItemID]
                      Alias   Request
          If   [Request].[CustomerID]<>[Loan].[CustomerID]   Then
             Group:   Raise an error and stop the m...
                RaiseError
                      Error Number   -1
                   Error Description   This item is on hold for another customer
                StopMacro
             End Group
          End If
    End If
 End If
```

Figure 15-3. *The Before Change macro – Part 3*

4. The next rule, shown in Figure 15-4, sets the `DueDate` and `OverdueFee`, if applicable.

www.it-ebooks.info

```
/*  When inserting a new loan, set the due date if not already specified  */
If  [IsInsert] And IsNull([DueDate])  Then
  /*  Calculate the due date by adding the loan period to the current dateMake sure
      to strip off the time portion  */
    SetField
                Name   Loan.DueDate
                Value  = FormatDateTime(Now()+[loanPeriod],2)
  End If
/*  Calculate the late fee if the item is overdue  */
If  Updated("CheckedIn") And Not IsNull([CheckedIn])  Then
  If  [DueDate]+1<Now()  Then
    /*  Compute the number of days overdue  */
      SetLocalVar
                Name   daysOverdue
           Expression  = Date()-[DueDate]
    /*  Calculate the overdue fee  */
      SetField
                Name   Loan.OverdueFee
                Value  = [daysOverdue]*[overdueFee]
    End If
End If
```

Figure 15-4. *The Before Change macro – Part 4*

5. Finally, enter the actions shown in Figure 15-5. If the `DueDate` is being changed, the macro verifies that this item can be renewed. If it can, the renewal count is incremented, otherwise an error is raised.

www.it-ebooks.info

```
/*   If updating the DueDate, increment the renewals                    */
If   Not [IsInsert] And Updated("DueDate")   Then
  /*   See if we are allowed to renew this loan                         */
  If   [Loan].[Renewals]>=[renewalsAllowed]   Then
    Group:   Raise an error and stop the macro
      RaiseError
            Error Number   -1
          Error Description   This item cannot be renewed
      StopMacro
    End Group
   /*   Increment the number of renewals                                */
  Else
    SetField
              Name   Loan.Renewals
              Value   = [Loan].[Renewals]+1
  End If
End If
```

Figure 15-5. The Before Change macro – Part 5

6. Save the macro changes and close the macro editor.
7. Close the Loan table.

Using the Compatibility Checker

The Compatibility Checker will scan your tables and make sure that they are compatible with SharePoint.

1. Go to the backstage view and click the Save & Publish tab. Then select the Publish to Access Services tab. The page should look like Figure 15-6.

www.it-ebooks.info

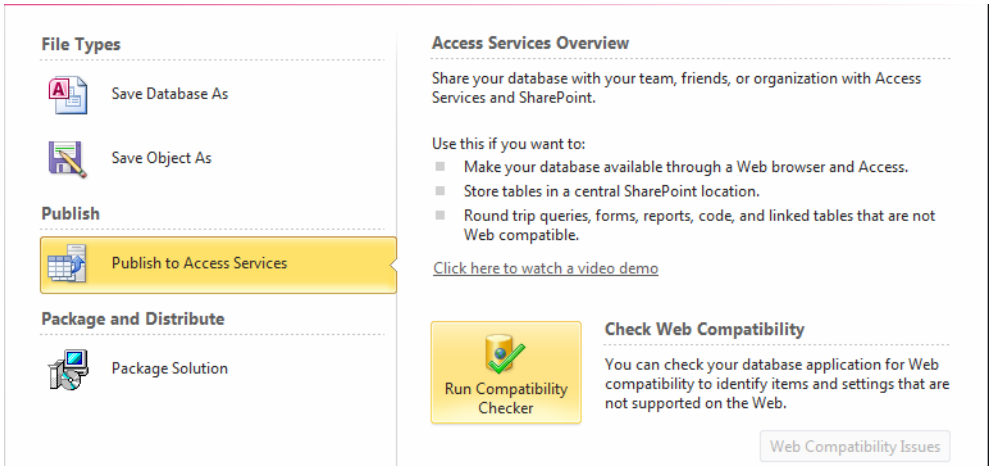


Figure 15-6*. The Publish to Access Services backstage view*

2. Click the Run Compatibility Checker button. If there are any issues, the background of this section will turn red. You can click the Web Compatibility Issues button, shown in Figure 15-7, to see the details of these issues.

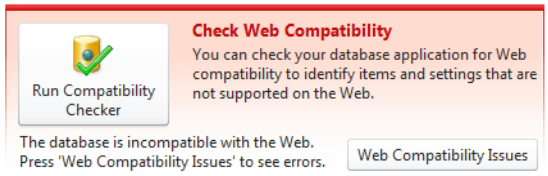


Figure 15-7*. The Compatibility Checker with errors*

Your database should be compatible with the Web. If any issues are reported, click the Web Compatibility Issues button and resolve the errors that are listed there.

Publishing the Access Database

Now you're ready to publish your database to SharePoint. You'll need a server that has SharePoint Server 2010 installed. You will also need sufficient privileges on this server to create a new SharePoint site.

Publishing to Access Services

Access Services is the SharePoint component that enables an Access database to be linked to SharePoint lists. It is available with the Enterprise edition of SharePoint Server. Once the database has been verified for compatibility, you'll be able to specify the server and site name to use for your Access database.

www.it-ebooks.info

Enter the server name that will host the new SharePoint site and enter **Library** for the site name, as shown in Figure 15-8. Then click the Publish to Access Services button.

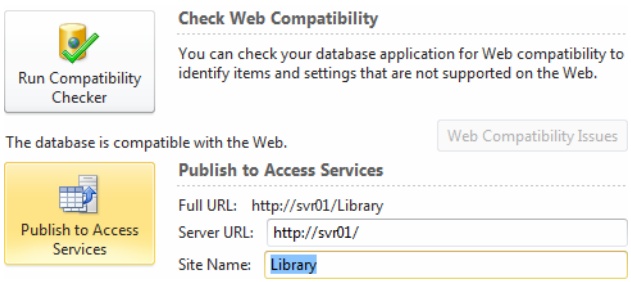


Figure 15-8*. Specifying the SharePoint site details*

It will take a minute or two to publish the database objects. When this has finished, the dialog box shown in Figure 15-9 will be displayed.

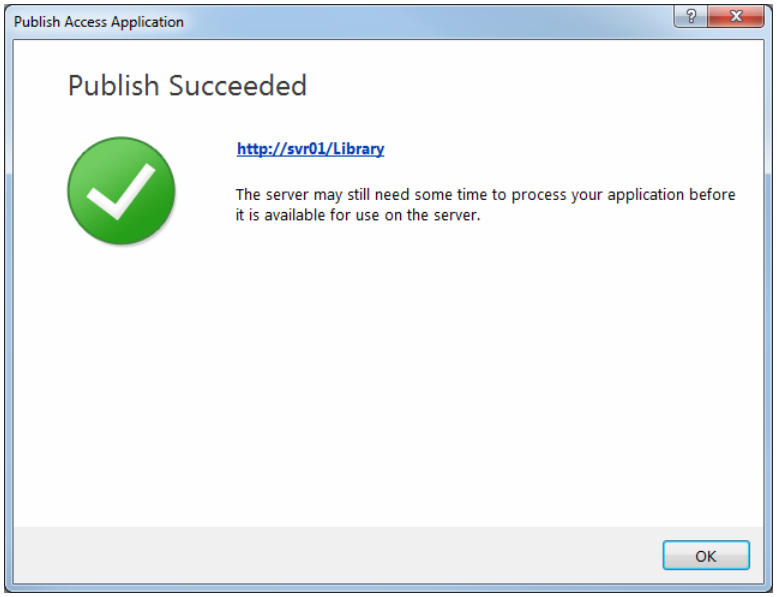


Figure 15-9*. Dialog box showing that the database was published successfully*

The URL displayed on this form is the address of the new SharePoint site. This is the link that the end users will use to access the web forms and reports that you will later deploy here.

www.it-ebooks.info

Exploring the SharePoint Site

To explore the SharePoint site, click the link shown in Figure 15-9. The page should look like Figure 15-10.

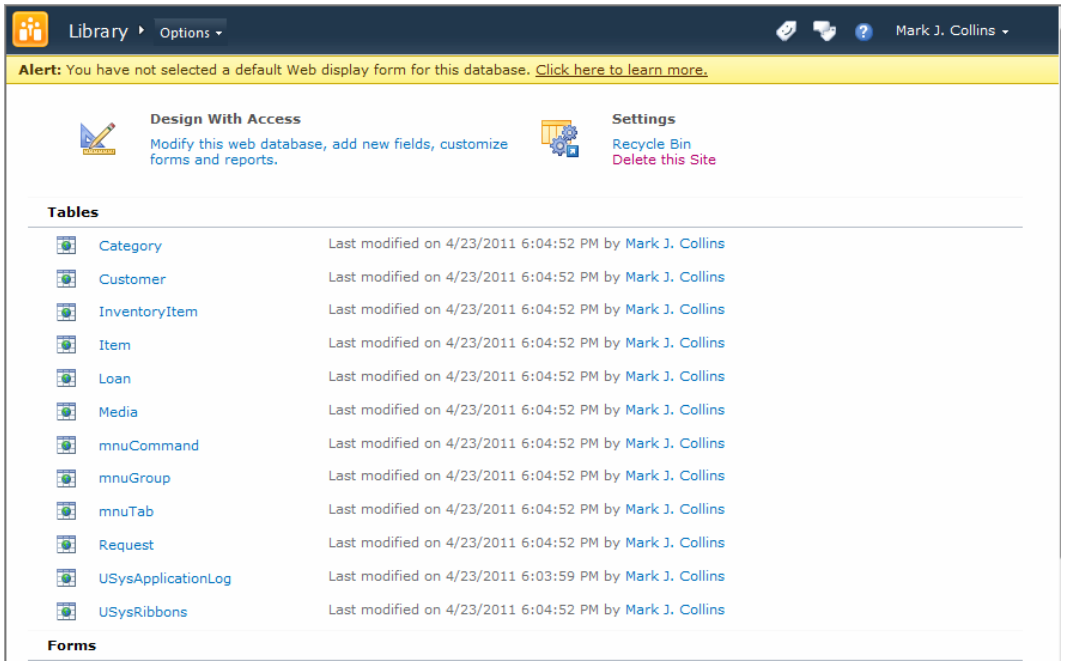


Figure 15-10. *The SharePoint site landing page*

Notice the warning at the top of the page. The Access database did not have any web forms, and so none were published. Consequently, there is no default page either. You will create a default web form later in this chapter. This will work like the `Menu` form you created in Chapter 10. It will be the portal from which the users can find the available forms and reports.

The current page simply lists the objects that are in the Access database. Even though the client objects (forms, reports, queries and so on) cannot be used on the web, they are cataloged here. If you try to edit them, the page will open the Access database so you can edit them there.

The actual SharePoint lists are not directly accessible from this page. However, there is a back door that you can use to view them. Click the Options dropdown and click the *Site Permissions* link, as shown in Figure 15-11.

www.it-ebooks.info

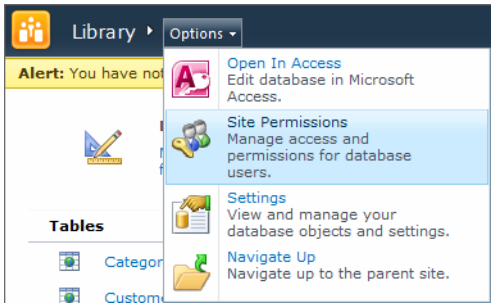


Figure 15-11*. Selecting the Site Permissions page*

The Site Permissions page provides a Site Actions dropdown. Click that and then click the *View All Site Content* link, as shown in Figure 15-12.

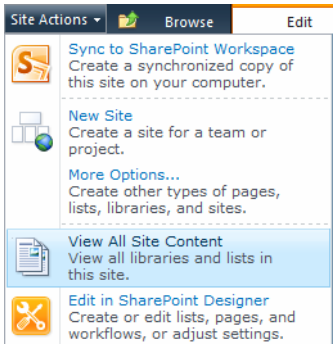


Figure 15-12*. Selecting the All Site Content page*

The All Site Content page, shown in Figure 15-13, enumerates all of the Lists and Document Libraries on your SharePoint site.

www.it-ebooks.info

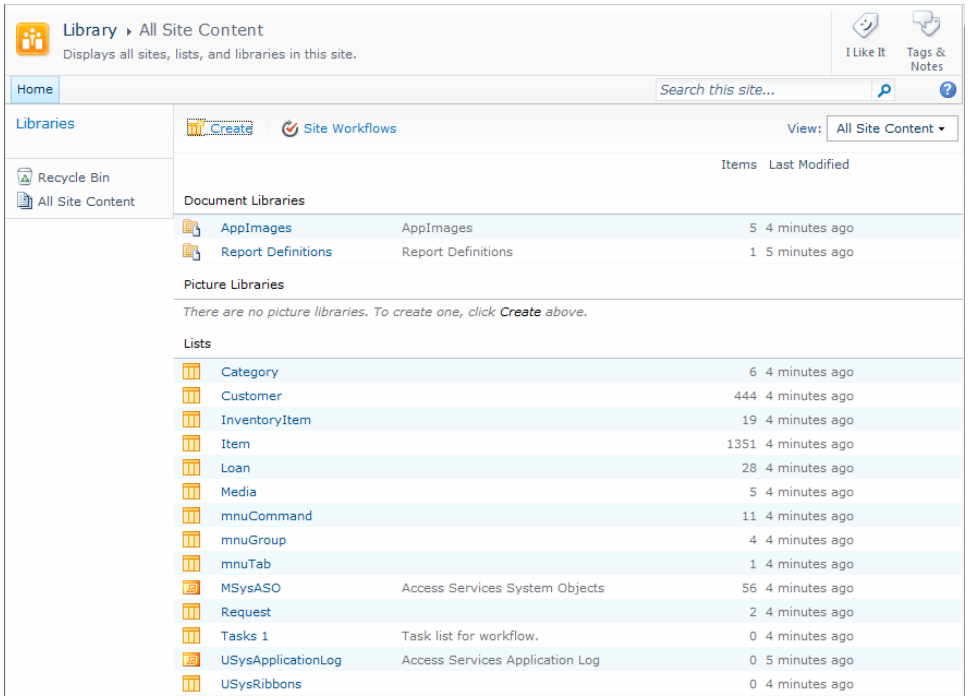


Figure 15-13. *The All Site Contents page*

There are a couple of system lists (`MSysASO` and `Tasks 1`), but the rest of the lists are copied from your Access tables. Click the `Customer` list to display the contents of this list, which is shown in Figure 15-14.

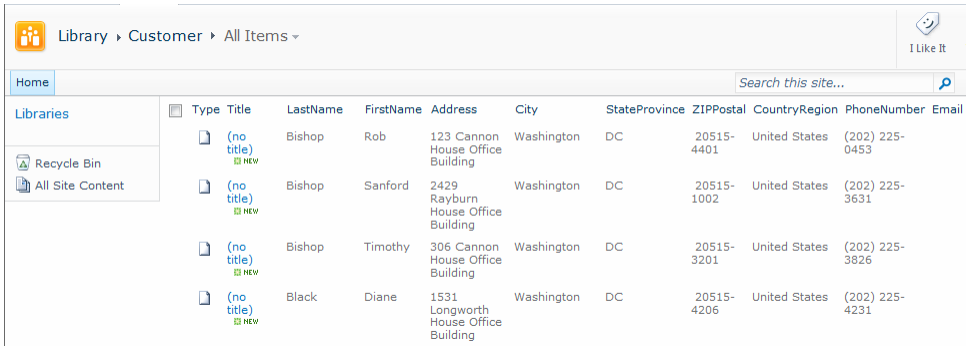


Figure 15-14. *The contents of the* `Customer` *list*

This is a fully functional SharePoint site, and you can use commands in the ribbon to display and edit the data, set up views, and even define workflows.

www.it-ebooks.info

■ **Tip** Because the data is in SharePoint, you can use SharePoint's security feature to grant or restrict access to each list based on user groups.

Exploring the Access Database

Go back to the Access database and see what changes were made there. Go to the backstage view and notice the synchronization information at the top of the page, which is shown in Figure 15-15. Access, along with Access Services on the SharePoint site, supports a two-way synchronization between the SharePoint site and the Access client database.

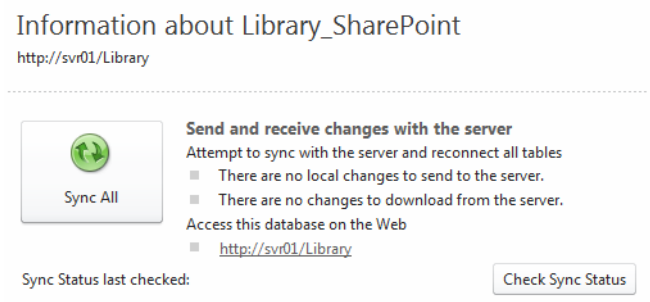


Figure 15-15. *Synchronization details*

Go to the Home tab of the ribbon. Notice in the Navigation pane that the tables have been replaced with web tables. There is no data stored in this database. These web tables are just links to the SharePoint site, just like the linked tables you created in Chapter 13. If you hover the mouse over one of these tables, the link details will be displayed as shown in Figure 15-16.

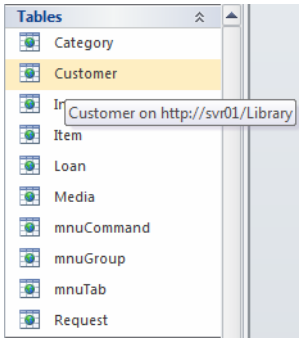


Figure 15-16. *Displaying the web table details*

Close the Access database and then re-open it. You should see the dialog box shown in Figure 15-17.

www.it-ebooks.info

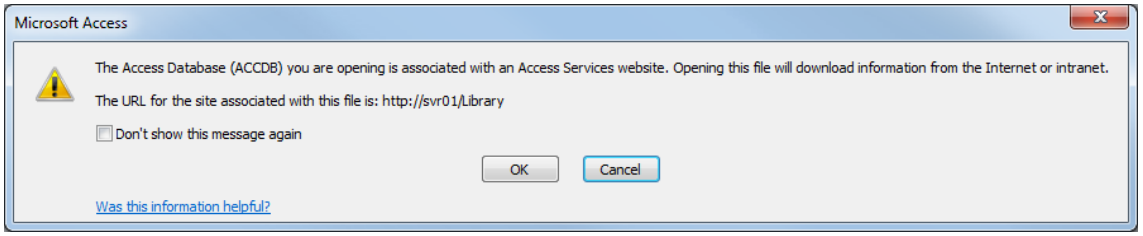


Figure 15-17. *Linked SharePoint site information*

This dialog box is letting you know that this Access database is linked to a SharePoint site. When you view data in this database, the data is actually being downloaded from the SharePoint site.

Restoring the VBA Code

One of the limitations of web forms and reports is that they cannot use VBA; instead, you must use macros. The publishing wizard has a nasty habit of deleting your VBA code. It doesn't always delete all of the code; to be safe, however, you should copy all of the code files.

■ **Note** I will show you how to create web forms later in the chapter.

Open the `Library_Access.accdb` file that you modified in the previous chapter. You'll copy the code from there. In the Navigation pane, double-click the `Main` object, which will display the VBA editor. Do the same in the `Library_SharePoint.accdb` file. You should have two VBA windows open. In the `Library_Access` database, select all the code in the `Main` module, copy it, delete the code in the `Main` module of the `Library_SharePoint` database, and then paste in the copied code.

For the following forms, create a code file in the `Library_SharePoint` database if one doesn't exist. To do that, open the form in the Design View and then click the View Code button in the ribbon. Then select the code from the `Library_Access` database, copy it, and then paste it into the code file in the `Library_SharePoint` database.

- `CheckOut`
- `CustomerAdmin`
- `CustomerDisplay`
- `CustomerLoans`
- `CustomerSearch`
- `InventoryItemLookup`
- `Item`
- `ItemSearch`

www.it-ebooks.info

Fixing the Default Value

The `CheckedOut` column of the `Loan` table was defined with a default value of `Now()`. This is a required field, and the `CheckOut` form relies on Access setting the value to the current date/time when a record is inserted. When publishing the `Loan` table to Access Services, this default value was not set up correctly, and the result is that the column has no valid default value. When a `Loan` record is created, the `CheckedOut` field will have no value, which will cause the insert to fail. The seemingly obvious solution is to set a default value in SharePoint so it will work the same way as it does with Access or SQL Server databases.

1. To do that, go to the All Site Content page using the backdoor method that I showed you earlier. Then click the Loan list.
2. In the List tab of the ribbon, click the List Settings button as shown in Figure 15-18.

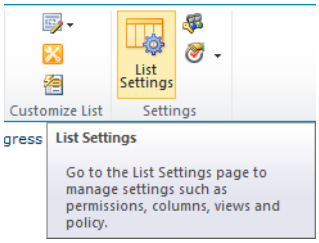


***Figure 15-18**. Selecting the List Settings page*

3. Then select the `CheckedOut` column. In the Additional Columns Settings section, the default value is set to a hard-coded date. Clear out this date and then select the Today's Date option as shown in Figure 15-19. Click OK to save the column changes.

www.it-ebooks.info

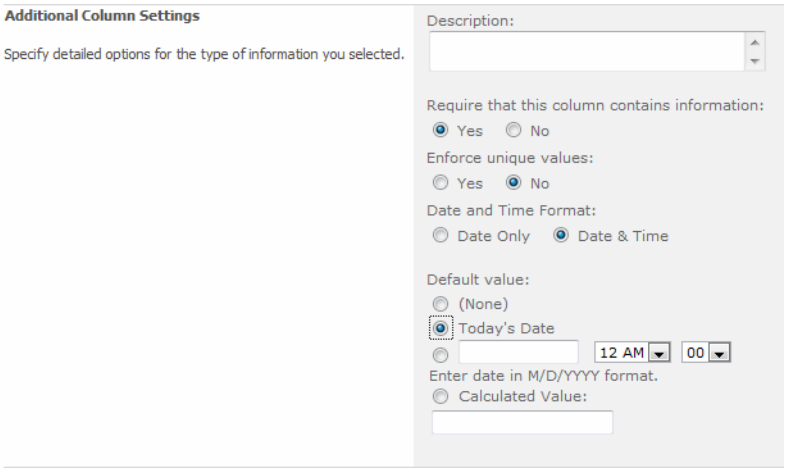


Figure 15-19. *Specifying Today's Date for the default value*

4. Then go to the `Library_SharePoint` database, right-click the `Loan` web table in the Navigation pane, and click the *More options* ➤ *Refresh List* links, as shown in Figure 15-20.

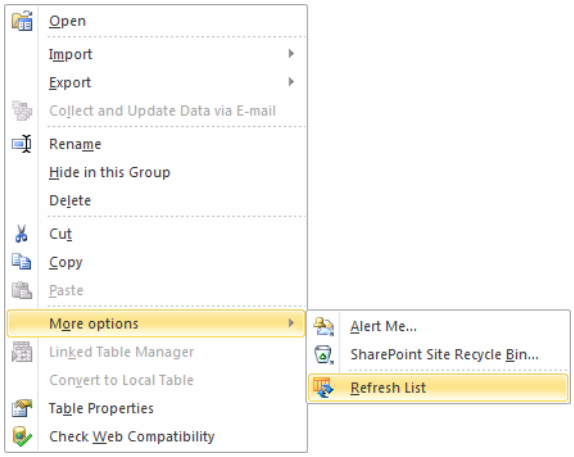


Figure 15-20. *Refreshing the `Loan` web table in Access*

This will resolve the issue by defaulting the `CheckedOut` column to the current date. The only problem with this is that this does not set the time portion of this field so it will always be midnight of the current date. SharePoint does not have a way to default a column to the current date/time. A better solution is to modify the `CheckOut` form to set this value when creating the record.

1. In the Access database, open the `CheckOut` form in the Design View.

www.it-ebooks.info

2. In the Design tab of the ribbon, click the Add Existing Fields button. Then drag the `CheckedOut` field to the Detail section of the form.
3. This will also create an associated label; delete the label.
4. Set the Visible property of the `CheckedOut` control to No.
5. Right-click the `CheckOut` button, which is in the Form Header, and click the *Build Event* link, which will display the VBA editor.
6. Add a line of code to set the `CheckedOut` field. The complete method is shown in Listing 15-1 with the additional line in bold.

Listing 15-1. *Setting the `CheckedOut` Value*

```
Private Sub CheckOut_Click()
    DoCmd.GoToRecord acDataForm, "CheckOut", acNewRec
    CustomerID = txtCustomerID
    InventoryItemID = txtInventoryItemID
    CheckedOut = Now()

    ' Cause the focus to move off the current record, which
    ' will cause it to be saved
    On Error Resume Next
    DoCmd.GoToRecord acDataForm, "CheckOut", acNext
    DoCmd.GoToRecord acDataForm, "CheckOut", acLast

    ' Reset the form controls
    txtInventoryItemID = ""
    DoCmd.Requery "InventoryItemLookup"
    InventoryItemLookup.Visible = False
    CheckOut.Visible = False
End Sub
```

Using Web Databases

In the previous chapter, I explained some of the file extensions used by Access; in this chapter, you'll use another one, `.accdw`, which is used to designate a web database. When you publish an `.accdb` file to a SharePoint site, the file is converted to a web database, even though the filename extension of original file was not changed. When you open a web database (`.accdw`), Access works a little differently, as I'll demonstrate later.

As I mentioned earlier, when you publish an Access database to a SharePoint site, all of your Access objects are stored in SharePoint, including the client forms. If you scroll down through the initial Settings page (refer to Figure 15-10) you'll see all of the client forms, reports, queries, and macros. If you select any of these objects, SharePoint will launch Access since you'll need to edit them using the Access application.

Close the `Library_SharePoint` database and then, from Windows Explorer, change the file extension to `.accdw`. This will help you when looking at the file name to know that this is a web database. However, because SharePoint can now be used as a repository for your Access objects, you don't even need to keep the original Access file. Instead, you can simply retrieve it from SharePoint as I will now show you. Anyone who needs to modify the Access file can get the latest version from SharePoint.

www.it-ebooks.info

Go to the home page of the SharePoint site. You can do this by clicking the icon at the top-left corner of the page or using the *Home* link above the Navigation pane. There is a link at the top of the page labeled *Modify this web database, add new fields, customize forms and reports*, as shown in Figure 15-21.

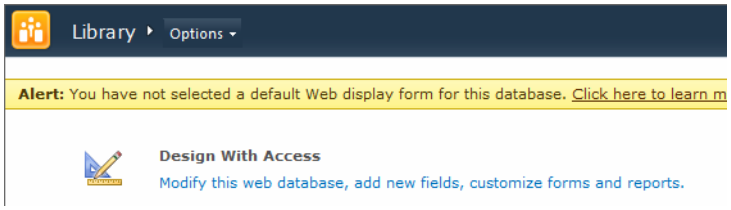


Figure 15-21. *The link to modify the database in Access*

Click this link and the File Download dialog box, shown in Figure 15-22, will be displayed.

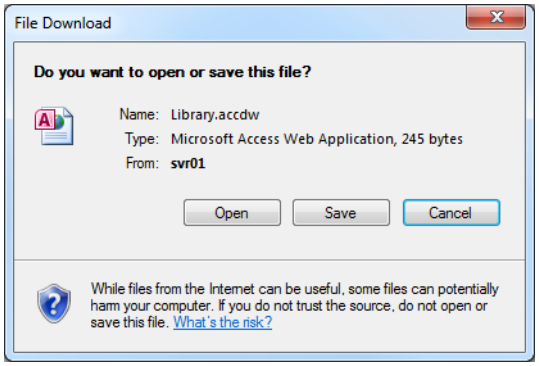


Figure 15-22. *The File Download dialog box with the* `Library.accdw` *file*

SharePoint has produced an Access web database, with the `.accdw` extension. The filename defaults to `Library` since that's the name of the SharePoint site. Click the Open button to open the Access database. You should see the dialog box shown in Figure 15-23.

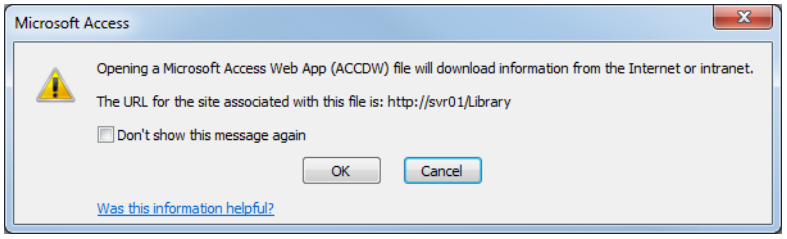


Figure 15-23. *Linked SharePoint site information*

www.it-ebooks.info

This is dialog box is similar to one displayed earlier when you opened the `Library_SharePoint.accdb` file. It is letting you know that this database will be accessing information from SharePoint. Of course the tables have been converted to SharePoint lists; that's the point of publishing the database. What may not have been immediately obvious is that the entire content of the Access database is also stored in SharePoint. This includes the client forms and reports as well as macros and VBA code.

Whenever you need to change something, the modified object is synchronized with SharePoint. So the SharePoint site is now the central repository for your Access database. Anyone with the necessary permissions can download a web database (`.accdw`), modify it, and synchronize the changes back to SharePoint. The URL of the associated SharePoint site is stored in the `.accdw` file. You can forward this file to someone else or rename it, but when you open it, it will still try to synchronize with the associated site.

■ **Caution** In Chapters 10 and 14, I showed you how to create an executable-only database (`.accde`) and recommended that you distribute this version instead of the original `.accdb` file. This will prevent the end users from modifying the database design. SharePoint does not support the `.accde` file type. The file that is made available through the SharePoint site (`.accdw`) is not locked down in any way. This is a fairly significant security hole. Hopefully this will be addressed in a future release.

Another difference with web databases is that the ribbon is changed. Go to the Create tab of the ribbon, which is shown in Figure 15-24.

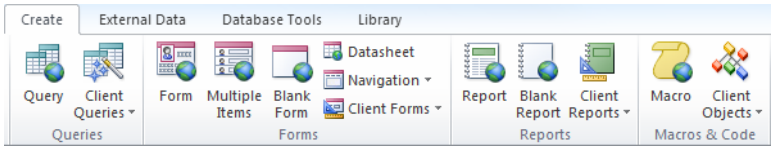


Figure 15-24. *The Create tab of the ribbon in a web database*

Most of the buttons on the ribbon are used to create web-based objects such as forms and reports. You can still create client forms as well, but navigating to the command buttons is a little different. If you click the dropdown icon next to the Client Forms button, you will see them listed as shown in Figure 15-25.

www.it-ebooks.info

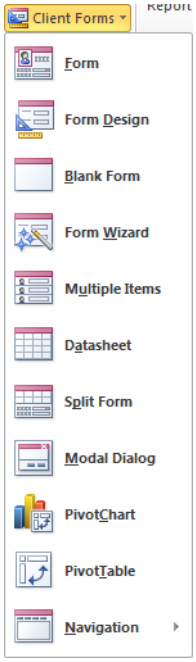


Figure 15-25*. The ribbon controls for creating client forms*

Creating Web Forms

At this point you have published the tables in your Access database to a SharePoint site. This is roughly equivalent to upsizing your database to SQL Server. The Access client forms work just like they used to except the data now resides in SharePoint. You have not yet created any web forms so, currently, SharePoint is simply a data repository.

In the rest of the chapter you'll create web forms that will be published to the SharePoint site. Web forms provide a browser-based solution that allows users from anywhere to access and update the data. This functionality does not require any client-side setup or installation. Web forms can be used from both the Access client and SharePoint. Client forms can only be used by the Access client. Both client forms and web forms will use the same data, which resides in the SharePoint lists.

Web forms are implemented just like the client forms except there are two primary limitations:

- They must use the layout control. You used this on the client forms as well, but in web forms it is required.
- They cannot use VBA. Any required logic must be implemented with macros.

www.it-ebooks.info

■ **Note** One of the things that you'll probably notice is that there is no Design View when working with web forms. You must do all of the design using the Layout View.

You'll start by creating a couple of simple forms using the form templates and then create a navigation form to provide a way for the end-user to navigate to these forms. The Create tab of the ribbon provides four buttons for creating forms:

- **Form:** Creates a single item form that displays one record at a time.
- **Multiple Items:** Creates a Continuous Form where the Detail section is repeated for each record.
- **Datasheet:** Creates a form that uses the Datasheet View.
- **Blank Form:** Creates a blank form that you can design from scratch.

You'll create one form for each of the first three options.

Creating a Category Web Form

You'll start with a single item form to display the `Category` records. If the Access database is not already open, go to the home page of the SharePoint site and click the link to modify the database in Access.

1. In the Navigation pane, select the `Category` table and then click the Form button in the Create tab of the ribbon. This will generate form and display it in the Layout View, as shown in Figure 15-26.

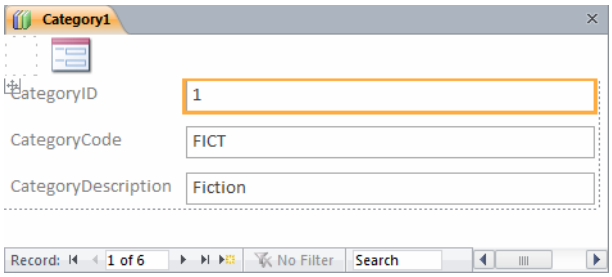


Figure 15-26. *The initial* `Category` *web form*

2. The top row of the layout control has an empty cell, a generic form logo, and the form caption. You'll make some adjustments to make the form look a little better. Select one of the cells in the top row. From the Arrange tab of the ribbon, click the Insert Below button to create a new row. Merge the last two cells of the new row into a single cell.

www.it-ebooks.info

3. Drag the Label control, which is named `Auto_Header0`, to the merged cell in the second row. Increase the cell height to about half an inch. Change the Back Color property for both cells in the second row to the standard Black color.
4. Delete the `Auto_Logo0` control and merge the three cells of the top row into a single cell.
5. From the Design tab of the ribbon, click the Insert Image button and then select the `ApressLogo` image, as shown in Figure 15-27. Then select the cell in the top row to place the image there.

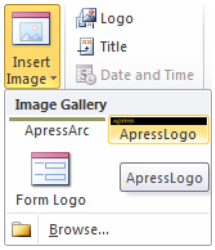


Figure 15-27. *Adding the Apress logo*

■ **Note** Notice that you can use the same Image Gallery for web forms that you used for client forms. The `ApressLogo` image that you added in Chapter 11 is also available to be used on web forms. In a web database, these shared images are stored in a Document Library, which I'll demonstrate later in this chapter.

---

6. Change the Size Mode property to Clip and increase the cell height to about 1 inch.
7. Select the top two rows and, from the Arrange tab of the ribbon, change the Control Padding to None.

The final layout should look like Figure 15-28.

www.it-ebooks.info



Figure 15-28. *The final layout of the* `Category` *web form*

Save the form and enter the name **webCategory** when prompted.

Creating a Media Web Form

Next you'll create a continuous form that will list all of the media types along with their setup values.

1. Select the `Media` table in the Navigation pane and click the Multiple Items button in the Create tab of the ribbon.
2. To save space, you'll remove the `MediaID` field since it is not meaningful to the end users. Right-click the `MediaID` field on the form and click the *Delete Column* link. This will remove the field as well as the associated label in the Form Header.
3. Change the label for the `RenewalsAllowed` field to **RenewalsAllowed**.
4. Adjust the top row to look just like the `webCategory` form. The first row will contain the Apress logo and the second row will have an empty cell and the form caption.

The final design should look like Figure 15-29.

www.it-ebooks.info

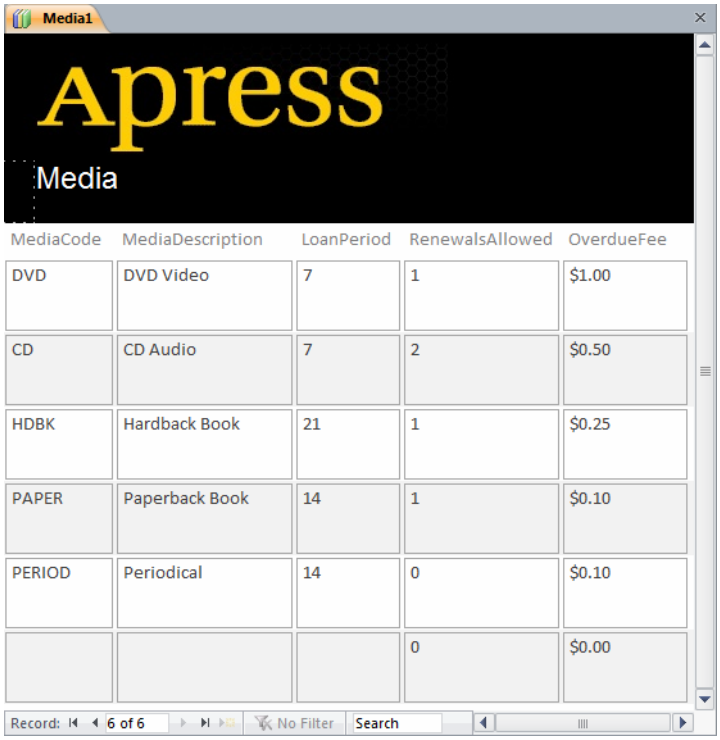


Figure 15-29. *The layout of the* `Media` *web form*

Save the form and enter the name **webMedia** when prompted.

Creating a Customer List Web Form

You'll now create a web form using the Datasheet View to display the records in the `Customer` table. Select the `Customer` table in the Navigation page and then click the Datasheet button in the Create tab of the ribbon. The generated form, shown in Figure 15-30, needs no modification.

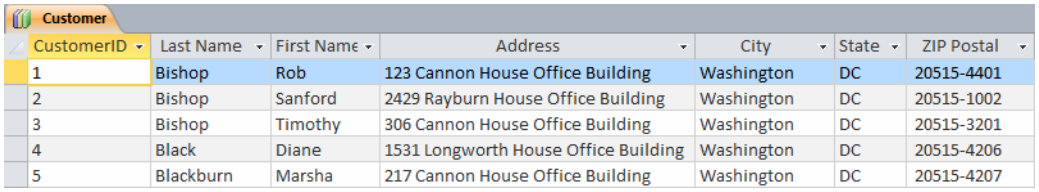


Figure 15-30. *The generate* `Customer` *web form*

Save the form and enter the name **webCustomer** when prompted.

www.it-ebooks.info

Creating a Navigation Form

I mentioned in Chapter 10 that Access 2010 provides a new navigation form. This is an improvement on the Switchboard that was used in earlier versions if Access. You were not able to use it in Chapter 10 because of the subform limitations. You will use it now as a portal for your web users so they can find and use the web forms and reports that have been published to SharePoint.

1. In the Create tab of the ribbon, click the dropdown icon next to the Navigation button to see the list of available templates shown in Figure 15-31.

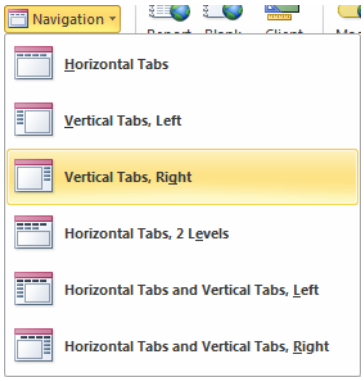


***Figure 15-31**. The available navigation templates*

2. The last three options allow for two levels of selections. Since you will only have a few forms, initially, a single level will be sufficient. I chose to put the tabs on the right side but you can use any layout that you prefer, so pick one now. The initial form will look like Figure 15-32 (depending on the layout you chose). As other forms are selected they will be displayed as subforms in the main area of the Navigation form. The header and navigation control will always be displayed.



***Figure 15-32**. The initial `Navigation` web form*

www.it-ebooks.info

3. Remove the logo control. Select all the cells in the top row and change the Back Color property to use the standard Black color. Set the Control Padding to None.
4. Change the Caption to **Library Management System (LMS)**.
5. To set up the navigation control, simply drag the web forms from the Navigation pane to the navigation control on the form. Add the web forms in this order:
   - `webCategory`
   - `webMedia`
   - `webCustomer`
6. After adding these forms to the navigation control, edit the captions to remove the "web" prefix.
7. There is a second row of empty cells. Right-click one of these cells and click the *Delete Row* link.
8. There is also an empty cell to the left of the subform. Right-click this and then click the *Delete Column link*.
9. Save the form and enter the name **webNavigation** when prompted. The form should look like Figure 15-33.

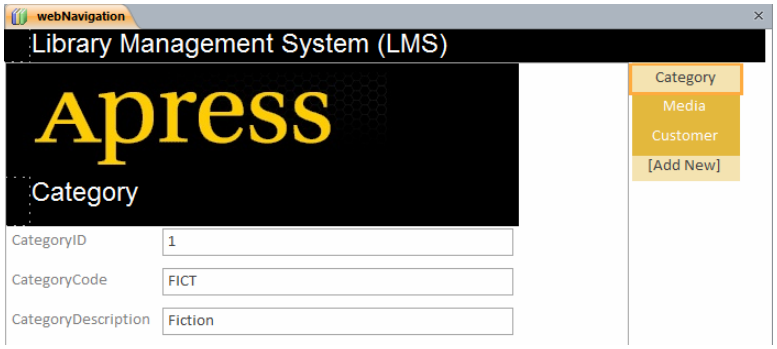


***Figure 15-33**. The layout of the navigation form*

Setting the Default Web Form

The last step is to make the `webNavigation` form the default web form so it will be displayed on your home page.

1. Click the File tab to display the backstage view. Click the Options button to display the Access Options dialog box.

www.it-ebooks.info

2. For the Web Display Form property, select the `webNavigation` form from the dropdown list as shown in Figure 15-34. Click the OK button to save your changes.

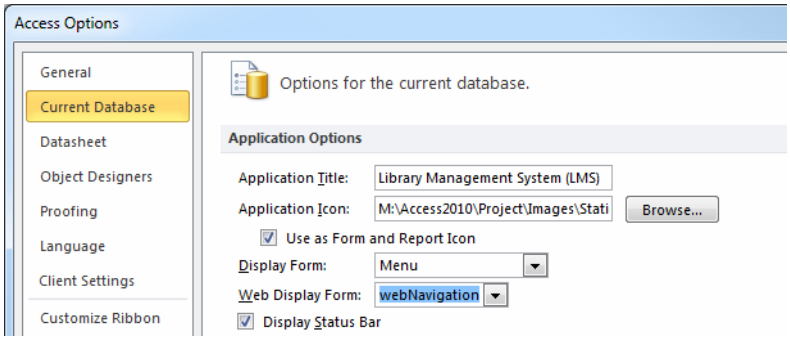


Figure 15-34*. Setting the default web form*

3. From the Info tab of the backstage view, click the Sync All button to copy your web forms to the SharePoint site.
4. Close the Access application.
5. Go back to the SharePoint site and refresh the page.

You should see the navigation form showing the first menu option, which is the `webCategory` form as demonstrated in Figure 15-35.

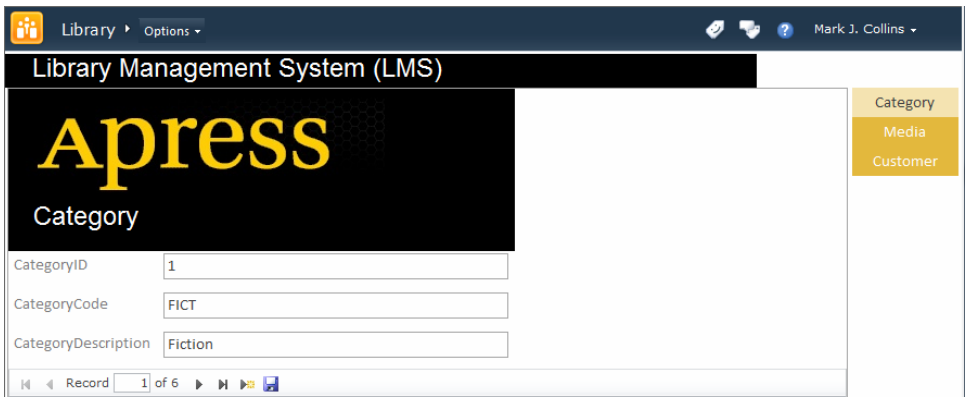


Figure 15-35*. The home page of the SharePoint site*

Try out the other forms to make they are working as expected.

www.it-ebooks.info

Adding an Item Web Form

For your final form, you'll recreate the `Item` client form as a web form so the web users can view the available items. You'll start by generating a standard web form using the single item template. You'll then re-arrange the controls and add an image and a web browser. You will also create an `InventoryItem` subform to show the status of the existing copes of the item. You will then provide the ability to create a new `InventoryItem` record. Finally, you will add a simple search facility.

From the SharePoint site, use the Options dropdown and click the *Settings* link as shown in Figure 15-36.

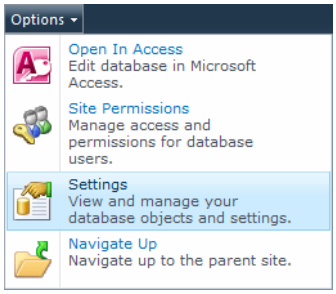


Figure 15-36. *Selecting the Settings page*

From this page, click the link to edit the database in Access. This will open the `.accdw` file where you'll add the new forms.

Creating the Initial Form

The first step will be to generate a form using the Form button and then rearrange the controls to match the existing client form.

1. Select the `Item` table in the Navigation pane and then click the Form button in the Create tab of the ribbon.
2. Delete the logo control and change the Fore Color property of the `Auto_Header0` control to Text Black.
3. Delete the row that contains the `ItemID` field; you will not need this on the form.
4. Change the labels for the `CategoryID` and `MediaID` and fields to **Category** and **Media** respectively.
5. Use the controls on the Arrange tab to add columns to the layout. You'll need a total of five columns.
6. Select all of the labels and set the Text Alignment property to Right.
7. Rearrange the controls to look like Figure 15-37.

www.it-ebooks.info

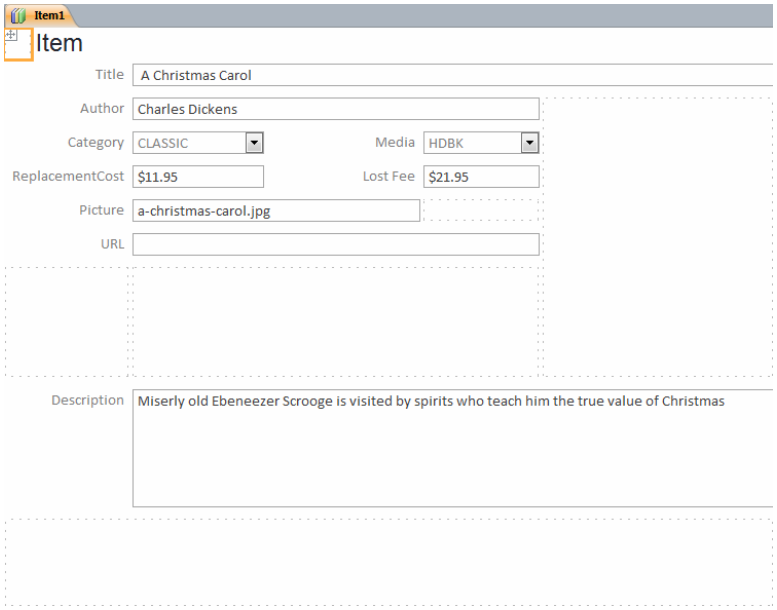


Figure 15-37. *The initial* `Item` *web form layout*

This layout leaves three empty cells for the picture, `InventoryItem` subform, and the web browser control. Save the form and enter the name **webItem** when prompted.

Creating the InventoryItem Subform

The `InventoryItem` subform will be a Continuous Form so you can see all of the records at the same time. You'll use the Multiple Items button to create it. The `Status` will be generated as a Combo Box, but you'll replace this with a Text Box and set the Locked property since this should not be user-modifiable.

1. Select the `InventoryItem` table in the Navigation pane and click the Multiple Item button in the Create tab of the ribbon. The Layout View will list all of the records in the database but this will be filtered once the form is linked to the webItem form.
2. Change the label for the `InventoryItemID` field to **ID**.
3. Select the `Status` control and delete it. Then select the Text Box control in the Design tab of the ribbon and click in the empty cell on the form. This will also insert a new column for the new control's associated label. Right-click one of the cells in this new column and click the *Delete Column* link.
4. Select the new Text Box control and in the Data tab of the Property Sheet, select the `Status` field for the Control Source property. Set the Locked property to Yes.
5. Delete the `ItemID` and `CurrentLoanID` columns.

www.it-ebooks.info

6. Delete the row that contains the logo and caption (the first row in the Form Header).
7. In the Data tab of the Property Sheet, select the Form object and set the Allow Additions and Allow Deletions properties to No.

The final layout should look like Figure 15-38.



Figure 15-38. *The layout of the* `InventoryItem` *web form*

There are some final touches to finish off:

1. Save the form and enter the name **webInventoryItem** when prompted. Close this form.
2. Go back to the `webItem` form. In the Design tab of the ribbon, click the Subform control and then click the empty cell just above the `Description`.
3. In the Data tab of the Property Sheet, select the `webInventoryItem` form. This should automatically set the linked fields.
4. Resize the cell so you can fit at least three `InventoryItem` records.
5. For the associated label that is generated, change it to **Inventory**.

Adding the Picture and Web Browser

In the client form you used an Image control to display the picture and set the path to the image file using VBA code. However, web forms do not support the Image control and, as I said earlier, you cannot use VBA code in a web form. Instead, to display a picture you will use the Web Browser control. For this to work you will need to be able to access the picture using a URL In other words, they must be served up through a web server. Fortunately, SharePoint is the perfect solution for that.

Storing Images in SharePoint

In addition to the lists that store your table data, SharePoint also provides for Document Libraries, which are great places to store any type of document, including images. In fact, when the Access database was published, a Document Library was already created for storing the static images that are included in the Image Gallery.

1. Go to the SharePoint site and navigate to the All Site Content page using the back door method I showed you earlier. Then select the `AppImages` library. The contents of this library should look like Figure 15-39.

www.it-ebooks.info

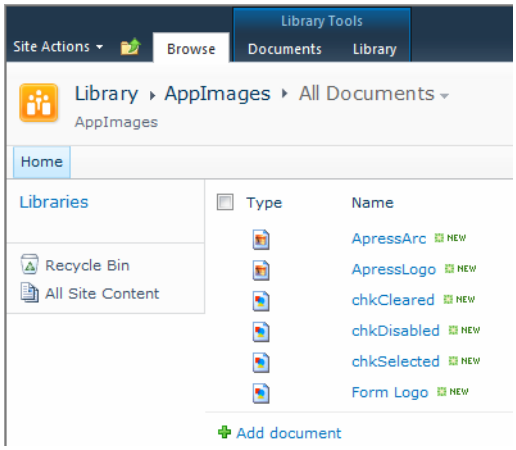


Figure 15-39*. The content of the* `AppImages` *library*

2. Click the *Add document* link. This will display the Upload Document dialog box shown in Figure 15-40.

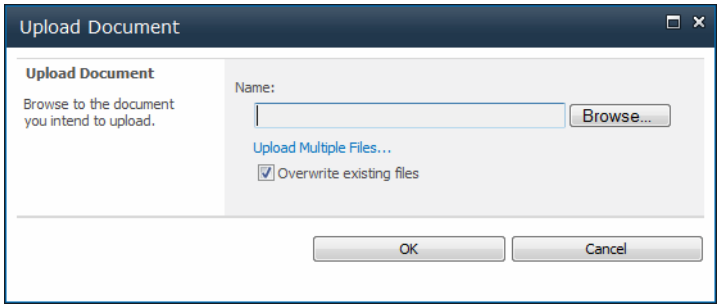


Figure 15-40*. The Upload Document dialog box*

3. Click the *Upload Multiple Files* link. This will display the Upload Multiple Documents dialog box shown in Figure 15-41.

www.it-ebooks.info

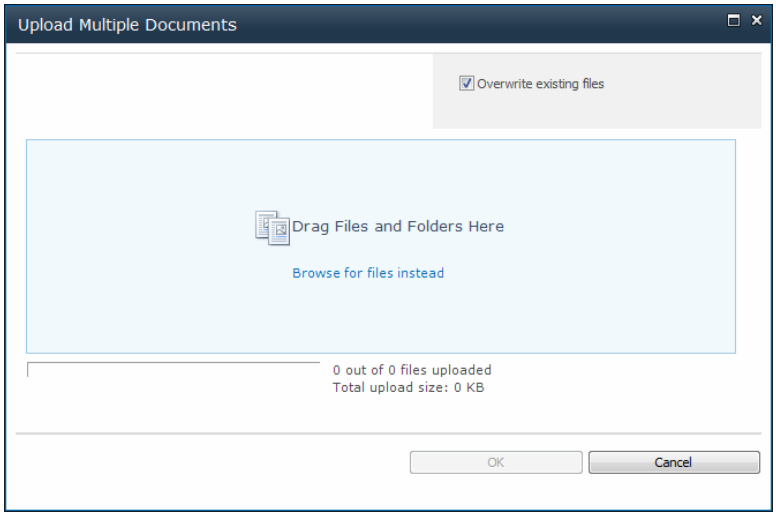


Figure 15-41*. The Upload Multiple Documents dialog box*

4. Open Windows Explorer and navigate to the `Images` folder (either the local folder or the shared folder you setup in Chapter 14). Drag all of the images to the Upload Multiple Documents dialog box.

5. Click the OK button to start the upload process. When the upload has finished, click the Done button to close the dialog box.

The `AppImages` library should now include all of your item pictures.

Displaying the Item Picture

Go back to the Access application. Now you'll add a Web Browser control to the web form and configure it to display the image specified by the `Picture` field. To avoid ambiguity, you'll change the name of the `Picture` control.

1. Select the Picture control and in the Other tab of the Property Sheet, change the Name property to **txtPicture**. When accessing this in an expression, this will make it clear that you're referring to the control and not the field in the `Item` table.

2. In the Design tab of the ribbon, click the Web Browser control and then select the empty cell on the right side of the form. When the Insert Hyperlink dialog pops enter the following expression in the Base URL property:

```
="http://<server name>/Library/AppImages/" & [txtPicture]
```

3. Enter your server name in this formula. This expression takes the address of the `AppImages` Document Library and appends the specific file name for the current item.

www.it-ebooks.info

4. Switch to the Form View to see how the form looks. It should be similar to Figure 15-42.

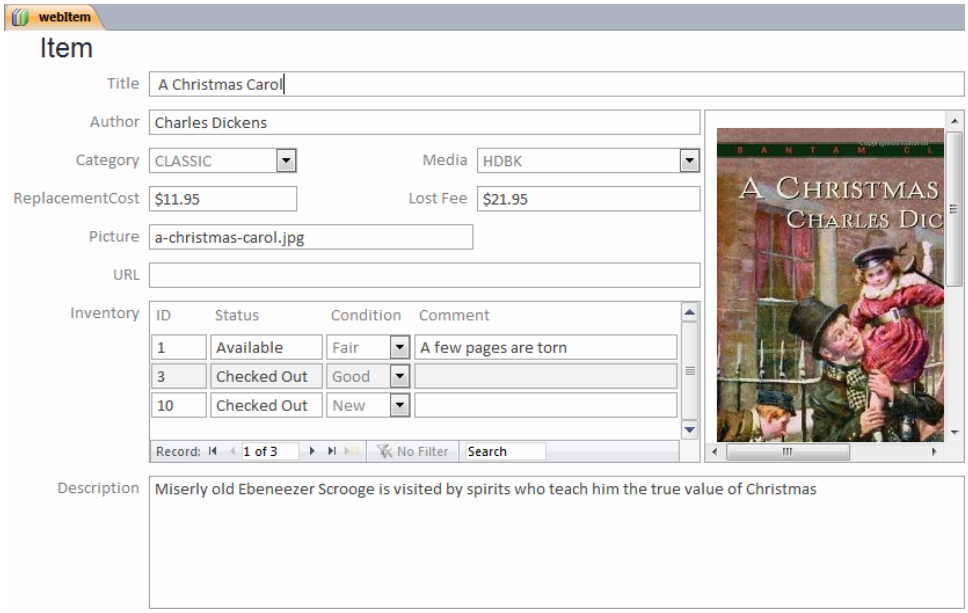


Figure 15-42. *The* `webItem` *form with picture*

Adding the Web Browser Control

Now you'll add another Web Browser control to display the web page specified in the URL field for each item. If no URL is specified for an item, the picture is displayed here instead.

1. Select the URL control and change its Name property to **txtURL**.
2. In the Design tab of the ribbon, click the Web Browser control and then click the empty cell at the bottom of the form. When the Insert Hyperlink dialog pops enter the following expression in the Base URL property:

```
=IIf(len([txtURL])>0, [URL], "http://<server>/Library/AppImages/" & [txtPicture])
```

■ **Tip** If no URL is specified for this item, this code uses the same URL for the Picture control and re-displays the image here. You could change this to display a different web page or display a static picture from the `AppImages` library.

---

www.it-ebooks.info

Adding an InventoryItem Record

In the next step, you'll add the ability to insert a record into the `InventoryItem` table. You'll use this feature when a new copy of an item is received. You'll need a record in the `InventoryItem` table to be able to lend it out. In the client form, you used a command button that called an append query. In the web form you'll invoke a data macro to add the record. First, you'll implement the data macro and then you'll add a button to call it.

Creating the Data Macro

All data macros, including named macros must be associated to a table. The `InventoryItem` table is the logical choice for this one.

1. Open the `InventoryItem` table using the Datasheet View. From the Table tab of the ribbon, click the Named Macro button and then click the *Create Named Macro* link, as shown in Figure 15-43.

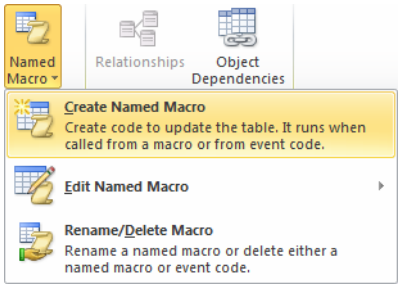


Figure 15-43*. Creating a named macro*

2. The macro will need to know what item is being received into inventory; the other values can be hard-coded. To specify the `ItemID` field, you'll add a parameter that will be supplied when the macro is called. Click the *Create Parameter* link and enter **itemID** for the Name and **The item being received** for the Description, as shown in Figure 15-44.

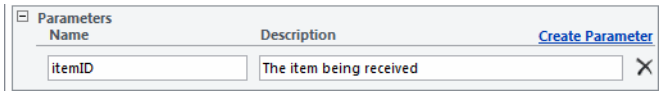


Figure 15-44*. Adding a parameter to the macro*

3. Select the `CreateRecord` action and then select the `InventoryItem` table. Enter **InventoryItem** for the Alias.
4. Add three `SetField` actions to specify the values for the following fields:
   - `ItemID:` Use the `itemID` parameter
   - `Condition:` "New"

www.it-ebooks.info

- Status: “Available”

The final macro design should look like Figure 15-45.

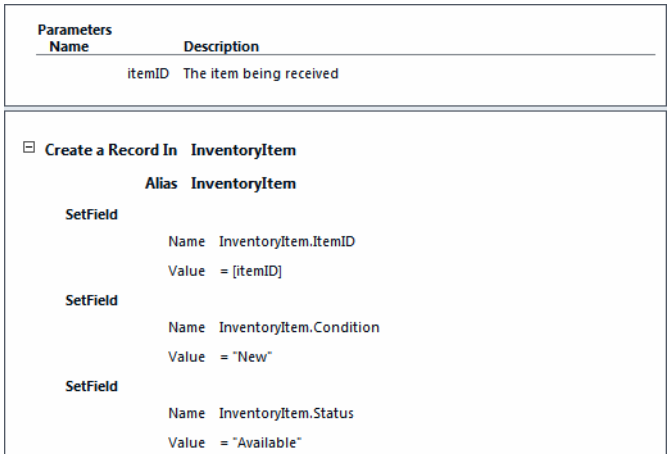


Figure 15-45*. The implementation of* `AddInventoryItem`

Save the macro and enter the name **AddInventoryItem** when prompted. Close the Macro Editor.

Adding a Command Button

Now you’ll add a command button to the `webItem` form. This will call the data macro to insert a record into the `InventoryItem` table. It will also re-query the `InventoryItem` subform so the new item will be displayed.

1. Open the `webItem` form using the Layout View.
2. In the Design tab of the ribbon, click the Button button and then click the empty cell just below the `LostFee` control.
3. In the Format tab of the Property Sheet, enter **Add Inventory** for the Caption property.
4. In the Event tab, click the ellipses next to the On Click event, which will display the Macro Editor.
5. For the first action, select the `RunDataMacro` action and then select the `InventoryItem.AddInventoryItem` data macro.
6. For the `itemID` parameter, enter the following expression:

```
[Forms]![webNavigation]![NavigationSubform].[Form]![ItemID]
```


www.it-ebooks.info

■ **Note** Because the `webItem` form will be a subform of the `webNavigation` form, to access the `txtSearch` control, you have to fully qualify the path to it by starting with the parent form.

7. For the second action, select the `Requery` action. Enter **InventoryItem** for the Control Name parameter.

The complete macro implementation should look like Figure 15-46.

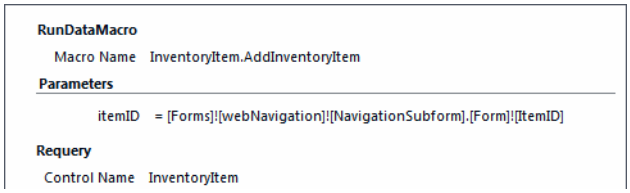


Figure 15-46. *The implementation of the command button*

Providing a Search Feature

For the final step, you'll add a simple search facility. This will allow the user to search by the Author's name. To do this, you'll add an unbound Text Box control to the form where the name can be entered. In the AfterUpdate event, you'll write a macro that will create a filter to limit the items displayed on the form.

1. While in the Layout View, select the `Title` field. In the Arrange tab of the ribbon, click the Insert Above button to add a new row at the top of the form.
2. Add a Text Box control to the second cell, which will also add a label to the first cell. Set the Caption of the label to **Search** and change the Name property of the Text Box control to **txtSearch**.
3. In the Event tab, click the ellipses next to the After Update event, which will display the Macro Editor.
4. In the Add New Action dropdown box, select the `SetFilter` action. Enter the following expression for the Where property:

```
[Author] Like [Forms]![webNavigation]![NavigationSubform].[Form]![txtSearch] & "*"
```

The Macro Editor should look like Figure 15-47.

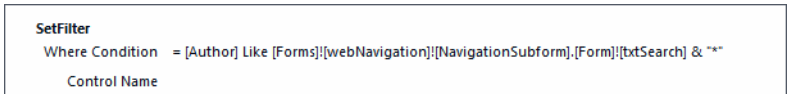


Figure 15-47. *Implementation of the After Update event*

5. Save and close the Macro Editor.

www.it-ebooks.info

Admittedly, this is a very simply search feature. If you want to build a more complex search, you can add multiple fields to the `webItem` form. In the macro, you'll need to add logic to test for the fields that were specified and implement conditional logic to construct the appropriate where clause. You did this in Chapter 9 using VBA code. You can accomplish the same thing in a macro. When you're done, you'll call the same `SetFilter` action passing in the where clause.

Modifying the Navigation Form

In order to use the new `webItem` form, you'll need to add it to the `webNavigation` form so the end-user can find and access it.

1. Open the `webNavigation` form and drag the `webItem` form to the navigation control.
2. Change the label in the navigation control to remove the "web" prefix.
3. Because the `webItem` form is considerable larger than the other forms you'll need to enlarge the `webNavigation` form so the `webItem` form will fit properly.
4. Save the changes.
5. From the Backstage view, click the Sync All button to update the SharePoint with the new and modified web forms.
6. Close the Access application

Now go to the SharePoint site. Test the search feature. Verify the picture and the URL is displayed correctly. Try adding an inventory item. The `Item` web form should look like Figure 15-48.

www.it-ebooks.info

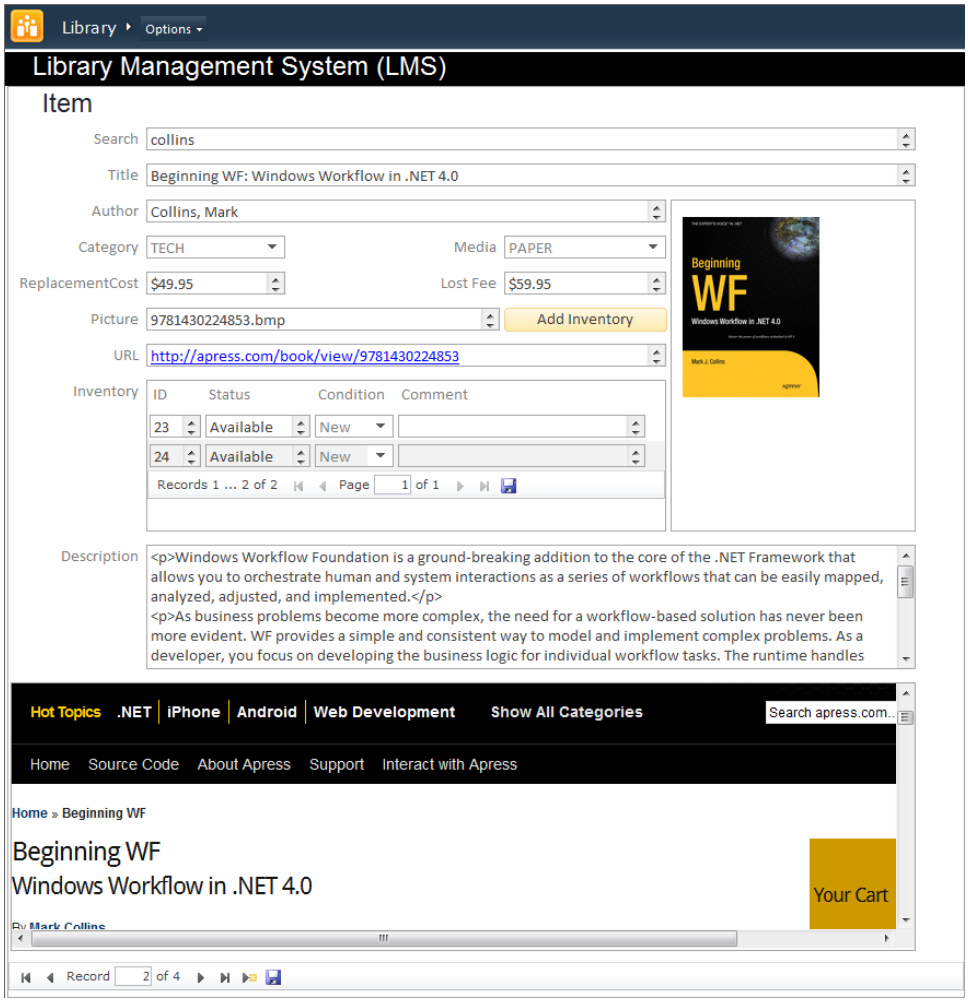


Figure 15-48. The final `Item` *web form implementation*

Summary

In this chapter you published your Access database to a SharePoint server. This moves the data to a central repository (SharePoint) as well provides a platform for creating web forms that SharePoint users can use to view and modify the data.

An additional benefit is that the Access client is also stored in the SharePoint site. The end users can use the web forms and/or download the Access database and use the client forms. This hybrid approach allows you to continue to use the client for complex forms that are difficult (or impossible) to implement on the web. Both the web forms and the Access client are available from SharePoint without needing to distribute anything to the client machines. Note, however, that to use the Access client, a copy of Access or the Access Runtime must be installed on the client (see Chapter 14 for details).

www.it-ebooks.info

The most significant limitation of web forms is that you cannot use VBA code. The Macro Editor for Access 2010 has been much improved to help alleviate the lack of VBA support.

The primary techniques that were presented in this chapter include:

- Publishing an Access database to a SharePoint server
- Resolving the publishing issues
- Creating several web forms
- Using the new Navigation form
- Storing dynamic images in a Document Library

www.it-ebooks.info

PART 5

Advanced Topics

In Part 5, I'll demonstrate some of the advanced features of Access. In Chapter 16, you'll send reminder e-mails from Access using the Outlook application. You'll also use the built-in data collection facility that uses both Access and Outlook to send e-mails and collect and process e-mail responses. In Chapter 17, I'll show you how to use several external data sources including the following:

- Outlook folders
- SharePoint lists
- XML files

Chapter 18 covers several unrelated features, such as the following:

- Using timers
- Calling the Windows API
- Using `TempVars`
- Using the `WebBrowser` control
- Adding charts

Finally, in Chapter 19, I'll explain the security features of Access 2010.

www.it-ebooks.info

CHAPTER 16

Integrating Outlook

Because Access 2010 is part of the Office suite of applications, it has been designed to be easily integrated with the other Office applications. In this chapter, you'll see how you can use Outlook to send e-mails from your Access solution. You can also use Access and Outlook together to create a simple data collection facility.

You'll provide a solution that sends an e-mail to each customer that has an overdue item. I'll show you first how to do this as a data macro. Using a data macro is also a great way to send an e-mail to notify someone when a record is inserted or updated. Using a macro is also the best way to send an e-mail from a web database. I will then show you how to send e-mails using VBA, which gives you a little more flexibility. The primary advantage of using VBA is that you can include attachments to your e-mail.

You'll then implement a simple data collection facility to send out and process a customer survey. The e-mail will go to all the customers in your database and include a form for them to enter a response. The replies are then automatically processed into a table in your Access database.

Sending E-mails

Your Access application will now send an e-mail to each customer that has overdue items checked out. This first version will send a summary e-mail to each customer indicating the number of overdue items that they have. You'll add the item details later.

The first step is to create a query that will return the list of customers that have overdue items. Then you'll create a data macro that will call this query and send an e-mail for each customer record. The last step will be to create a UI macro that you will use to invoke the data macro. You will use this to start the e-mail process.

Creating the OverdueItems Query

The `OverdueItems` query will search for `Loan` records that have not been checked in and are past the due date. It will also need to return the customer details such as name and e-mail address. Because a customer can have multiple overdue items, you'll need to use aggregation to provide a single row per customer.

1. Open the `Library.accdb` file that you modified in Chapter 12. This file contains local tables.
2. From the Create tab in the ribbon, click the Query Design button.
3. In the Show Table dialog box add the following tables to the query and then close the dialog box:

www.it-ebooks.info

- `Customer`
- `Loan`

4. Double-click the following columns to add them to the query:
   - `Customer.CustomerID`
   - `Customer.LastName`
   - `Customer.FirstName`
   - `Customer.Email`
   - `Loan.CheckedIn`
   - `Loan.DueDate`
   - `Loan.LoanID`
5. In the Criteria row add the specified expressions in the following columns:
   - `Customer.Email`: **Is Not Null**
   - `Loan.CheckedIn`: **Is Null**
   - `Loan.DueDate`: **<Now()-1**
6. In the Design tab of the ribbon, click the Total button to make this an aggregate query. This will set the Total property for each of the columns to Group By.
7. Change the Total property of `Loan.CheckedIn` and `Loan.DueDate` to Where and unselect the Show check box for these columns.
8. Change the Total property of `Loan.LoanID` to Count. Change the Field property of this column to **ItemCount: LoanID**.
9. Save the query and enter the name **OverdueItems** when prompted.

The query design should look like Figure 16-1.

www.it-ebooks.info

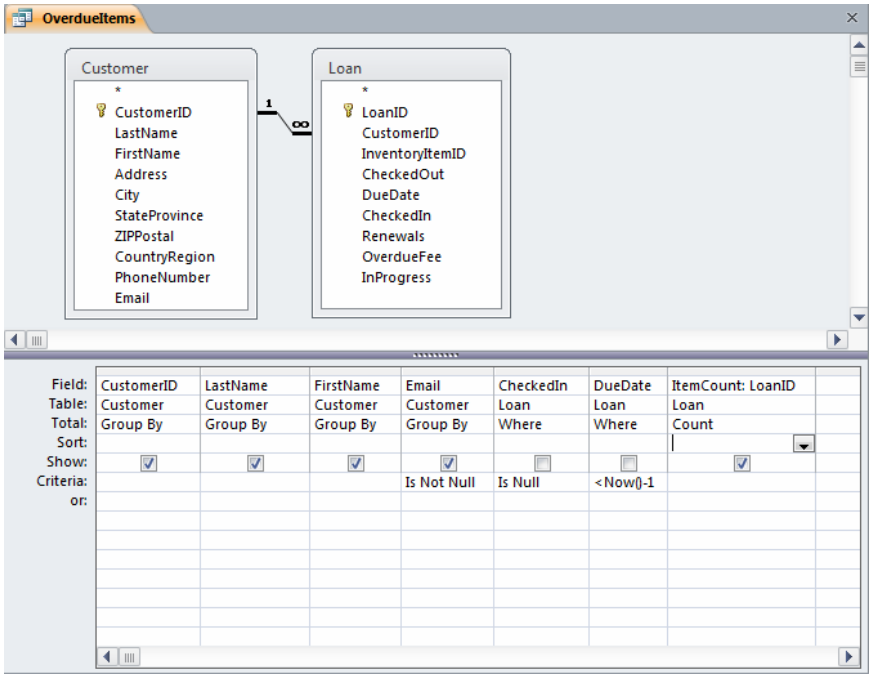


Figure 16-1. The design of the `OverdueItems` *query*

Switch to the Datasheet View to test the query. You should have results similar to Figure 16-2.

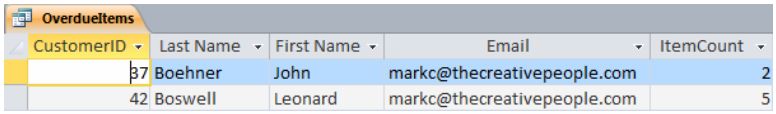


Figure 16-2. Sample results from the `OverdueItems` *query*

■ **Tip** I updated all of the `Customer` records to use my e-mail address so I don't send out any test e-mails to actual customers. If you need to save the existing e-mail addresses, make a copy of the `Customer` table first, before updating the e-mails.

■ **Note** If your query doesn't return at least two records you'll need to generate some test data. If you have `Loan` records with no `CheckedIn` date, you can adjust the `DueDate` column so they will be overdue. To generate additional `Loan` records, use the `CheckOut` form as explained in Chapter 7.

www.it-ebooks.info

Creating a Data Macro

Access 2010 includes a `SendEmail` macro action that provides a simple way to send an e-mail. It can be used only in a data macro, however. For this project, you'll create a named macro that can be invoked from the application. For more information on using named macros see Chapter 3. Named macros must be associated to a table; you'll add this one to the `Loan` table.

1. Open the `Loan` table in the Datasheet View.
2. From the Table tab of the ribbon, click the Named Macro button and then click the *Create Named Macro* link as shown in Figure 16-3. This will display the Macro Editor.

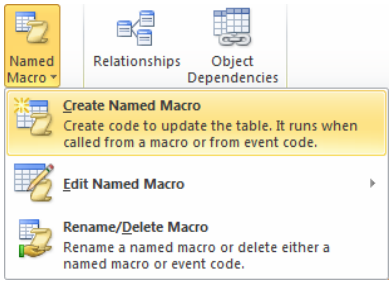


***Figure 16-3**. Creating a named macro*

3. In the Add New Action dropdown list, select the `ForEachRecord` action. For the For Each Record In property, select the `OverdueItems` query. Leave the Where Condition property blank and enter **OverdueItems** as the Alias.
4. There will now be two Add New Action dropdown lists; one inside the `ForEachRecord` data block and one after it. In the one inside the data block, select the `SendEmail` action.
5. Enter the following values for the specified parameters:
   - To: `=([OverdueItems].[Email])`
   - Subject: `Overdue Items`
   - Body: `=([OverdueItems].[FirstName] & " " & [OverdueItems].[LastName] & " has " & [OverdueItems].[ItemCount] & " item(s) that are overdue")`
6. Save the macro and enter the name **SendOverdueEmails** when prompted.

The macro design should look like Figure 16-4.

www.it-ebooks.info

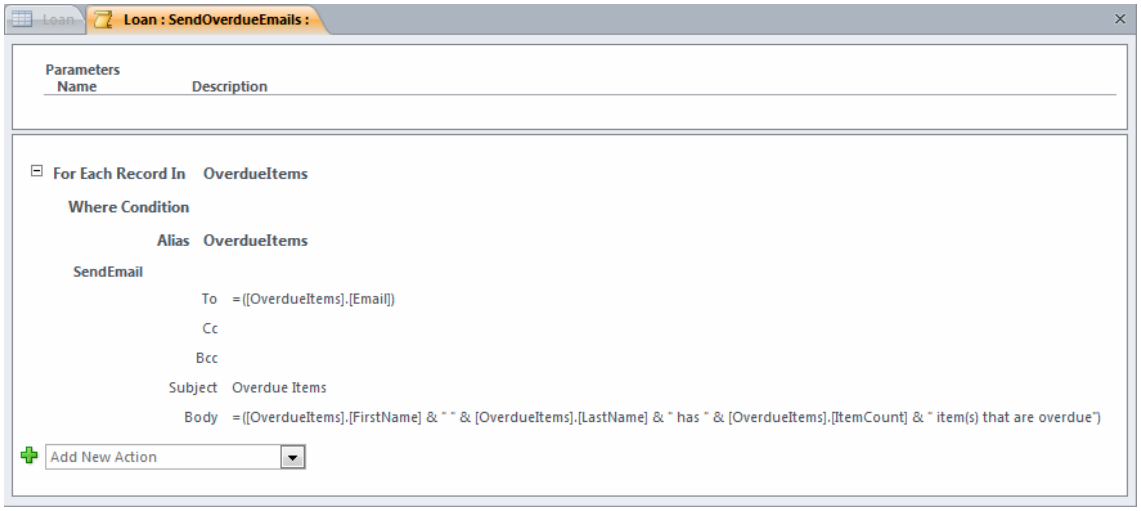


Figure 16-4. *The* `SendOverdueEmails` *data macro*

Adding a UI Macro

In order to run this data macro you'll need to create a UI macro that will invoke it. This will be available in the Navigation pane. You could also add a command button to the `Menu` form to invoke the data macro.

1. From the Create tab of the ribbon, click the Macro button.
2. In the Add New Action button select the `RunDataMacro` action.
3. For the Macro Name parameter select the `Loan.SendOverdueEmails` data macro.
4. Save this macro and enter the name **SendOverdueEmails** when prompted.

The design should look like Figure 16-5.

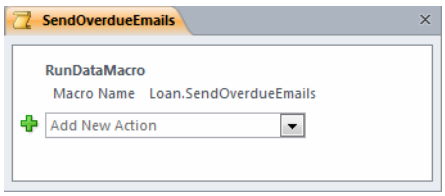


Figure 16-5. *The* `SendOverdueEmails` *UI macro*

www.it-ebooks.info

Sending the E-mails

To send the e-mails, from the Navigation page, right-click the `SendOverdueEmails` macro and click the *Run* link. You will then see the dialog box shown in Figure 16-6. This dialog box is displayed to ask for your permission to allow another program to send an e-mail on your behalf. The Allow button is disabled for a few seconds to avoid any automated attempt to bypass this dialog box. You will need to approve each e-mail that is being sent.

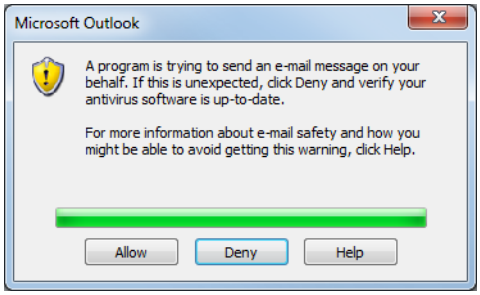


Figure 16-6. *The Outlook Security guard*

■ **Note** Viruses are notorious for infecting a machine and then using the e-mail client to propagate itself to other people. The propagation is often much more effective because the e-mail looks like it came from someone they know. In response, Microsoft developed an update to Outlook 98 and Outlook 2000 known as the Outlook Security Patch. This block, or a version of it, has been in every release of Outlook since then.

■ **Note** If you use a data macro to send e-mails from a web database, the macro is run on the server. This means that the e-mails are being sent by SharePoint instead of Outlook. SharePoint does not have this security issue. However, you will need to configure the outgoing e-mails in SharePoint.

Depending on your e-mail provider and how the Send and Received is configured in Outlook after a while you should receive these e-mails in your Inbox. The e-mail will look similar to Figure 16-7.

www.it-ebooks.info

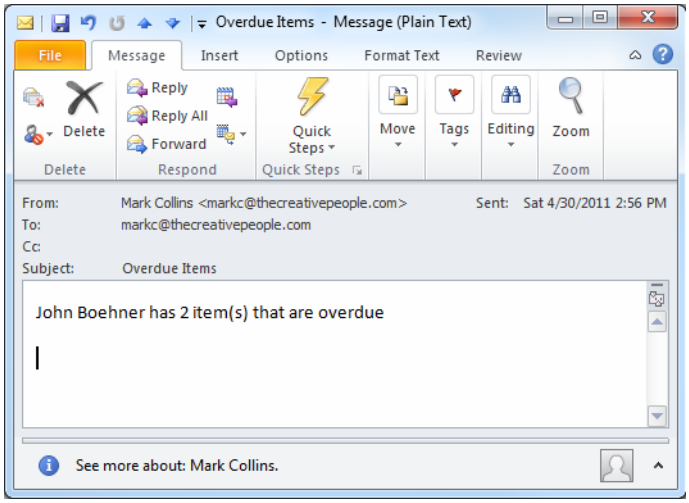


Figure 16-7. *The incoming e-mail message*

■ **Tip** If you don't want to wait to receive the e-mail you can also look in your Sent folder. Because you are also the sender, the e-mail will be in your Sent folder immediately. It will be in your Inbox after it has made the round trip to your e-mail provider and back.

Sending E-mails with VBA

Now I'll show you another way to send an e-mail using VBA. This version will include the details of the items that are overdue. You could have done this in the `SendOverdueEmails` macro with some local variables and a little bit of string manipulation. This version will also include a picture of the items as attachments, which you cannot do in a macro.

The approach with this version will be similar to the previous exercise. You'll need a query to return the details. In this case, however, rather than just returning summary information, you'll need the item details as well. You will then add a function to the `Main` module that will run the query, format the e-mails, and send them. Finally, you'll modify the `SendOverdueEmails` UI macro to call this function instead of the data macro.

Creating the OverdueItemDetails Query

The `OverdueItemDetails` query will return the details of each `Loan` record that is overdue. It will also return the customer details such as `FirstName`, `LastName`, and `Email` as well as the item details including `Title` and `Picture`.

1. From the Create tab in the ribbon, click the Query Design button.

www.it-ebooks.info

2. In the Show Table dialog box add the following tables to the query and then close the dialog box:
   - `Customer`
   - `Loan`
   - `InventoryItem`
   - `Item`
3. Remove the link between the `Loan` and `InventoryItem` tables that connects `Loan.LoanID` to `InventoryItem.CurrentLoanID`.
4. Double-click the following columns to add them to the query:
   - `Customer.CustomerID`
   - `Customer.LastName`
   - `Customer.FirstName`
   - `Customer.Email`
   - `Loan.CheckedIn`
   - `Loan.DueDate`
   - `Item.Title`
   - `Item.Picture`
5. In the Criteria row add the specified expressions in the following columns:
   - `Customer.Email`: **Is Not Null**
   - `Loan.CheckedIn`: **Is Null**
   - `Loan.DueDate`: **<Now()-1**
6. In the Sort row, select Ascending for `CustomerID` and `DueDate`. Make sure the `CustomerID` column is before the `DueDate` column so the query will be sorted by `CustomerID` first.
7. Save the query and enter the name **OverdueItemDetails** when prompted.

The query design should look like Figure 16-8.

www.it-ebooks.info

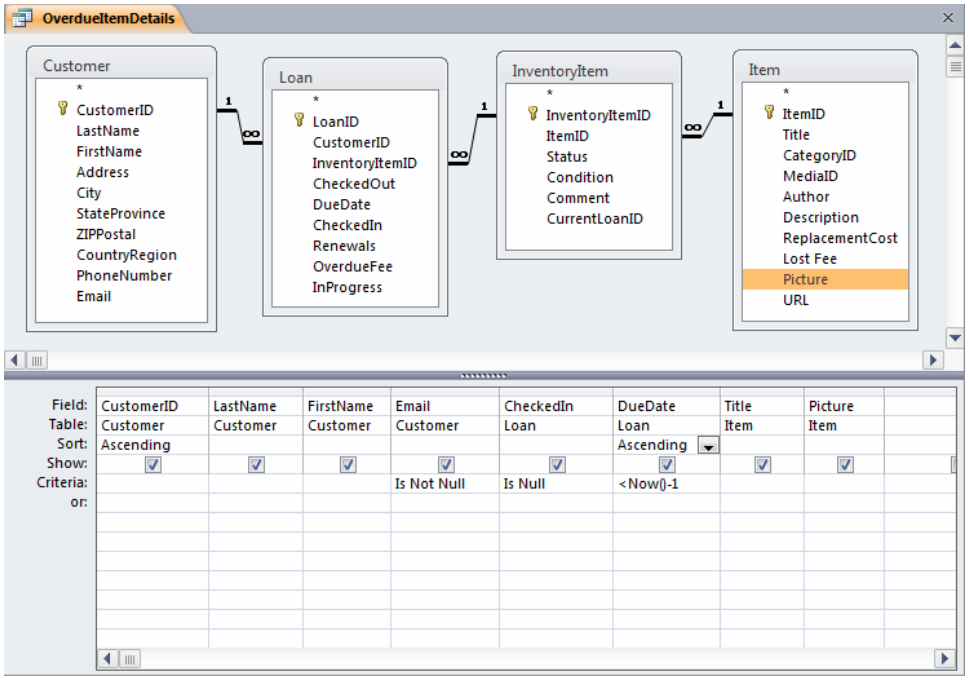
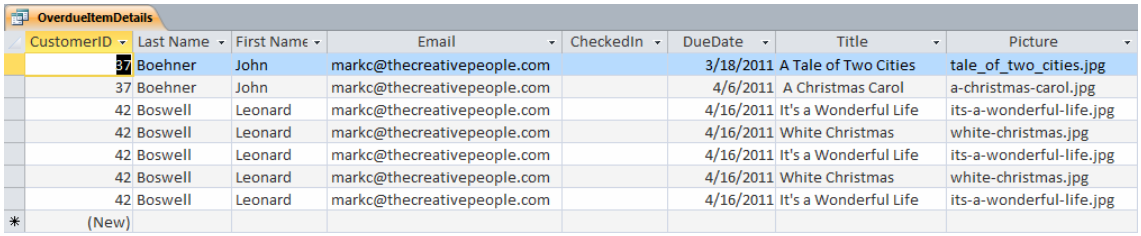


Figure 16-8. *The design of the* `OverdueItemDetails` *query*

Switch to the Datasheet View to test the query. You should have results similar to Figure 16-9.



CustomerID	Last Name	First Name	Email	CheckedIn	DueDate	Title	Picture
37	Boehner	John	markc@thecreativepeople.com		3/18/2011	A Tale of Two Cities	tale_of_two_cities.jpg
37	Boehner	John	markc@thecreativepeople.com		4/6/2011	A Christmas Carol	a-christmas-carol.jpg
42	Boswell	Leonard	markc@thecreativepeople.com		4/16/2011	It's a Wonderful Life	its-a-wonderful-life.jpg
42	Boswell	Leonard	markc@thecreativepeople.com		4/16/2011	White Christmas	white-christmas.jpg
42	Boswell	Leonard	markc@thecreativepeople.com		4/16/2011	It's a Wonderful Life	its-a-wonderful-life.jpg
42	Boswell	Leonard	markc@thecreativepeople.com		4/16/2011	White Christmas	white-christmas.jpg
42	Boswell	Leonard	markc@thecreativepeople.com		4/16/2011	It's a Wonderful Life	its-a-wonderful-life.jpg
(New)							

Figure 16-9. *Sample results from the* `OverdueItemsDetails` *query*

Implementing the VBA Code

You'll now add a function to the `Main` module to send the e-mails. This function will execute the `OverdueItemDetails` query and use the results to format the e-mails. You still need to send a single e-mail to each customer even though the query may return multiple items. So you'll need some logic to group the details into a single e-mail.

www.it-ebooks.info

Double-click the `Main` module in the Navigation pane, which will open the VBA editor. You will need a reference to the Outlook object model. From the Tools menu click the *References* link. Then add the reference to the `Microsoft Outlook 14.0 Object Library` as shown in Figure 16-10.

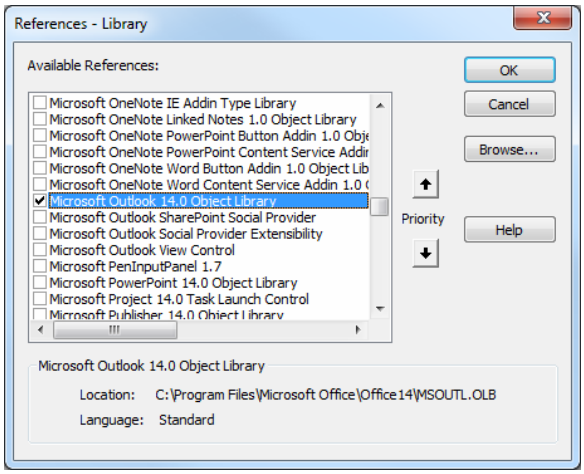


Figure 16-10. *Adding the Outlook reference*

Now add the `SendOverdueEmails` function using the code shown in Listing 16-1.

Listing 16-1. *Implementation of the `SendOverdueEmails` Function*

```
Public Function SendOverdueEmails() As Integer

Dim mailSender As Outlook.Application
Dim mail As Outlook.MailItem
Dim rs As DAO.Recordset
Dim custID As Integer
Dim email As String
Dim itemList As String
Dim customerInfo As String

    custID = 0
    itemList = ""

    Set mailSender = New Outlook.Application

    Set rs = CurrentDb.OpenRecordset("OverdueItemDetails", dbOpenDynaset)
    Do Until rs.EOF
        If (rs("CustomerID") <> custID) Then
            If (custID <> 0) Then
                mail.Body = customerInfo & vbCrLf & _
                    "The following items are overdue:" & vbCrLf & vbCrLf & itemList

                mail.Save
```


www.it-ebooks.info

```
                mail.Send

                itemList = ""
            End If

            Set mail = mailSender.CreateItem(olMailItem)

            mail.To = rs("Email")
            mail.Subject = "Overdue Items"

            customerInfo = rs("FirstName") & " " & rs("LastName")

        End If

        If (Len(rs("Picture")) > 0) Then
            mail.Attachments.Add ("L:\Images\" & rs("Picture"))
        End If

        itemList = itemList & rs("Title") & " was due on " & _
            Format(rs("DueDate")) & vbCrLf
        custID = rs("CustomerID")

        rs.MoveNext
    Loop

    If (custID <> 0) Then
        mail.Body = customerInfo & vbCrLf & _
            "The following items are overdue:" & vbCrLf & vbCrLf & itemList

        mail.Save
        mail.Send
    End If

    ' Clean up
    Set mailSender = Nothing

    rs.Close
    Set rs = Nothing

End Function
```

■ **Note** The query returns a record for each item that is overdue, but you only want to send one e-mail to each customer. As the VBA code loops through each record, it keeps track of current customer and sends the e-mail when all the records for the current customer have been processed. An alternative approach would be to use a query that just returns a list of customers and then call a second query to get the details.

www.it-ebooks.info

Most of this function is performing basic string manipulation to format the content of the e-mails. I'll highlight some of the statements that pertain to sending e-mails. The following statement obtains an instance of the Outlook application. If Outlook is not currently running, the following command will start it:

```
set mailSender = New Outlook.Application
```

This statement creates a new e-mail:

```
set mail = mailSender.CreateItem(olMailItem)
```

Subsequent code then sets the following properties:

- `To`
- `Subject`
- `Body`

This statement is used to add an attachment to the e-mail:

```
mail.Attachments.Add ("L:\Images\" & rs("Picture"))
```

■ **Caution** This code uses the mapped `L:` drive that you setup in Chapter 14. You may need to change this to use a different mapped drive or use a UNC file path in your environment.

Finally, the following code saves the message and then sends it:

```
mail.Save
mail.Send
```

Save the code changes and close the VBA editor.

Sending the E-mails

You'll now modify the `SendOverdueEmails` UI macro to call this function instead of the data macro.

1. From the Navigation pane, right-click the `SendOverdueEmails` macro and click the *Design View* link. This will open the Macro Editor.
2. Select the `RunDataMacro` action and click the delete icon.
3. In the Add New Action dropdown list, select the `RunCode` action.
4. For the Function Name parameter, enter **SendOverDueEmails**(). Make sure you include the parentheses.

The macro should look like Figure 6-11.

www.it-ebooks.info

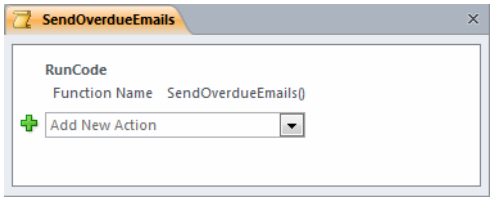


Figure 6-11. *The modified* `SendOverdueEmails` *UI macro*

Save the macro changes and then click the Run button from the Design tab of the ribbon. You won't see anything happen, but if you look in Outlook you'll find that the e-mails have been sent and you'll soon find them in your Inbox.

The e-mails should look like Figure 6-12. Notice that the body of the e-mail includes details of the items that are overdue. Pictures of the items are also attached.

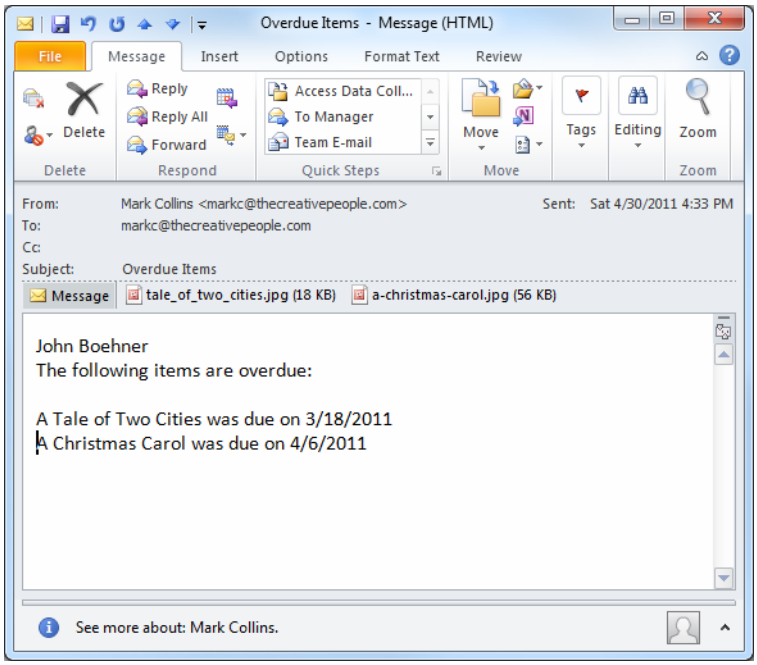


Figure 6-12. *Sample e-mail with details and attachments*

www.it-ebooks.info

■ **Note** The security patch described earlier was improved in Outlook 2010 to allow programmatic access without the annoying pop-up dialog box in some situations. If you have anti-virus software installed and it is up-to-date, the dialog box is suppressed. This change does not apply to e-mails sent from a data macro like you did earlier. E-mails sent using VBA, however, can now be sent without the pop-up as long as you keep the client protected. For more information, check out the article at `http://msdn.microsoft.com/en-us/library/ff864479.aspx`.

Using Data Collection

The collaboration of Access and Outlook provides a really useful feature for sending e-mails and processing the responses. In Access you can define a data entry form and the underlying table that will store the data. The form is then e-mailed by Outlook to a list of recipients that can be defined in another Access table. As the replies are received by Outlook, the data is extracted from the form and stored in the table. The e-mail recipients do not need to use Outlook; any e-mail client will work. Outlook will send the initial message and then receive and process the responses.

You can use this for a lot of purposes such as gathering expense or status reports. You could use this to collect feedback from students at a training event or attendees at a meeting. In this project you'll send a survey to the customers in your database asking for areas of interest.

Understanding Data Collection

The data collection process will send a form to one or more recipients and when responses are received, the data in the form will be stored in a table. So the table behind the form needs to contain all the fields on the form. It's just like the forms you created in Chapter 6; once you select the table that the form will use, you can then select which of the fields should be included on the form.

Some of the fields can be read-only; they are populated in the form when it is sent and the recipient cannot change it. For example, if you're asking for feedback from an event, you might want to include the name and date of the event. You would not want these to be modified. You could also send data for the recipient to review and modify if necessary. A good example of this is sending an e-mail for someone to update their contact information. The form can include the current values and the recipient can fill in missing fields or modify the existing ones.

The first step in designing a data collection solution is to decide if the response will insert a record in the results table or update one. For example, suppose you held meeting that 20 people attended and you want to send them a form to gather feedback. You could populate the results table with 20 records, leaving many of the fields blank. The form can then display the non-blank fields and leaving the recipient to fill in the remaining. When the response is processed, the record will be updated with the data provided in the form. With this approach, you'll need to pre-populate the results table before using the data collection wizard.

The other approach is to not create the records in the results table until the responses are received. In this case, the data collection process will insert the records for you, but only as each response is received. The disadvantage of this approach is that the form cannot be pre-filled with any values. Remember the form simply mirrors the data in the table. Because there is no record when the initial form is generated there is no data to display.

www.it-ebooks.info

■ **Caution** The results table needs to be a table and not a query. The wizard will allow you to use a query and it will send the e-mails. However the pre-populated data will not be included in the e-mails. More importantly, Access will not be able to process the responses.

The next item to decide is who should receive the e-mail. Access allows you the option of simply entering the e-mail address(es) into the To line in Outlook. This is acceptable for a very small list. For internal recipients, you can also use a distribution list if the list is relatively static. However, in many cases you will want to get the e-mail addresses from Access. If you're using the first option where the results table is pre-populated, you can include the e-mail address in that table.

The e-mail address can also be in a table that is related to the results table. For example you can add the `CustomerID` column in your results table and use the Lookup Wizard to create this column so a table relationship is established. Access will then be able to get the e-mail from the `Customer` table for each pre-populated record in the results table.

■ **Note** If there are multiple records with the same e-mail address, Access will only send one e-mail to each address; however, the e-mail will contain multiple copies of the form. This will actually work and if both forms are filled in, both records will be updated. However, it may look somewhat odd to the recipient.

If you're not pre-populating the results table, you can still add a lookup column in the results table to another table that contains e-mail addresses. For example, you can add a `CustomerID` column to the results table. Since there is no data in the results table to limit the list, you will send the e-mail to the entire list of customers. If you want to send the e-mail to a subset of customers, create a table that only contains the subset and link the results table to that instead.

Defining a Data Collection E-mail

This will become clearer as you implement a data collection project. In this scenario, you'll send an e-mail to all customers (that have an e-mail address) and request them to choose a media type and category that they would like to see the library expand its selection of. In this scenario there's no pre-populated data that needs to be included with the initial form, so you'll let the collection process insert records as each response is processed.

The data that needs to be captured is the `MediaID` and `CategoryID` selected by each customer. The `Customer` table contains the e-mail addresses. The data collection wizard will only allow you to get e-mail addresses from the results table or a table related to it. To satisfy that requirement, you'll also add a `CustomerID` field to results table.

www.it-ebooks.info

Creating the Results Table

The results table for this project will be called Survey and will include a primary key and the MediaID and CategoryID fields, which contain the collected data. It will also have a CustomerID column that is never populated, but is needed so the wizard can link the Customer table.

1. From the Create tab of the ribbon, click the Table Design button.
2. Add the following fields:
   - **SurveyID:** AutoNumber, primary key
   - **MediaID:** Lookup (use the Media table)
   - **CategoryID:** Lookup (use the Category table)
   - **CustomerID:** Lookup (use the Customer table)

■ **Note** Refer to Chapter 2 is you have questions about using the Lookup Wizard.

The table design should look like Figure 16-13. Close the Survey table.

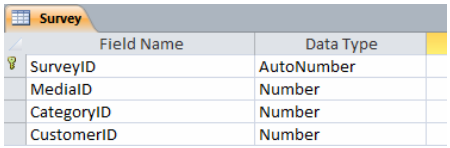


Figure 16-13. *The design of the Survey table*

■ **Note** To simplify the testing I have cleared out the Email field in the Customer table for all customers except for George Washington. I also updated the e-mail address for John Adams to use my webmaster e-mail address. You should make similar changes to your data to minimize the number of e-mails that need to be sent.

Installing the Data Collection Add-In

If you get an error suggesting that Outlook is not installed, you may have the necessary add-in disabled. To check, open Outlook, go to the Backstage view and click the Options button. Then select the Add-Ins tab. The add-in needed is called the Microsoft Access Outlook Add-in for Data Collection and Publishing. You might see this listed as an inactive add-in as shown in Figure 16-14.

www.it-ebooks.info

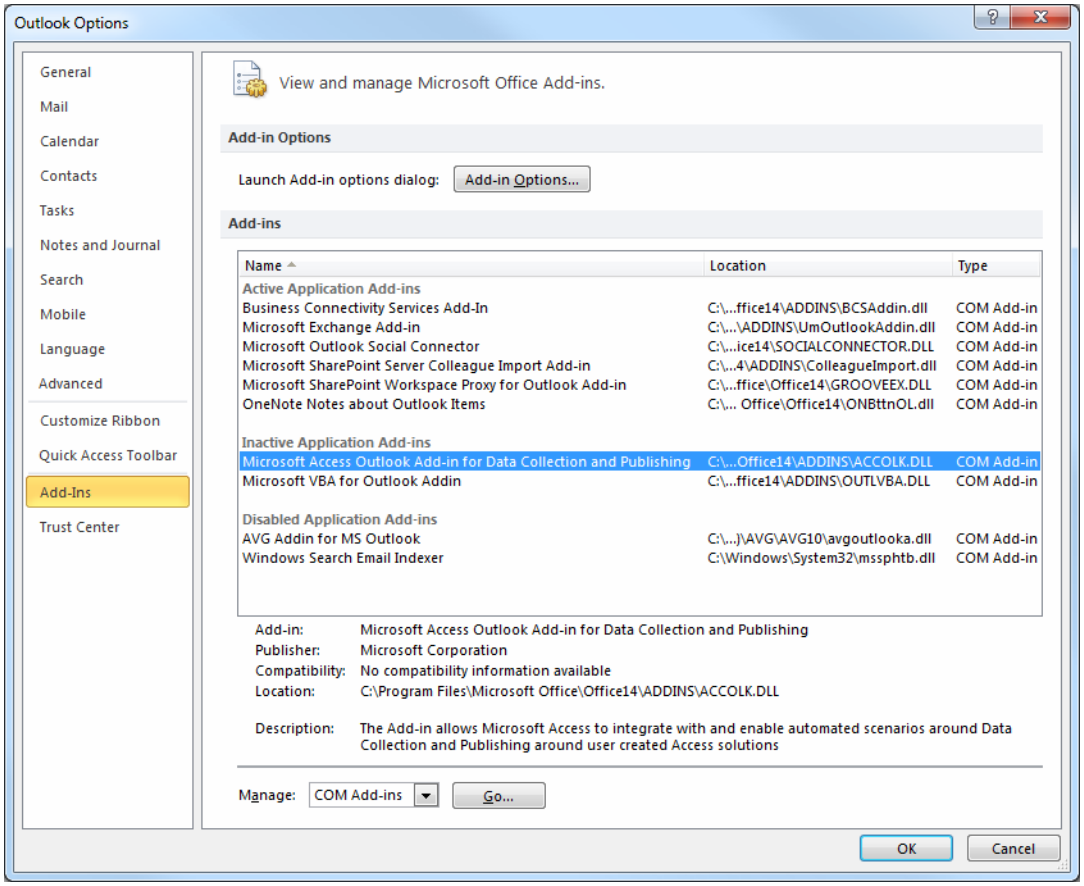


Figure 16-14. *Displaying the existing Add-Ins*

To activate it, select the COM Add-ins dropdown at the bottom of the dialog box and click the Go button. In the COM Add-Ins dialog box, select this add-in as shown in Figure 16-15.

www.it-ebooks.info

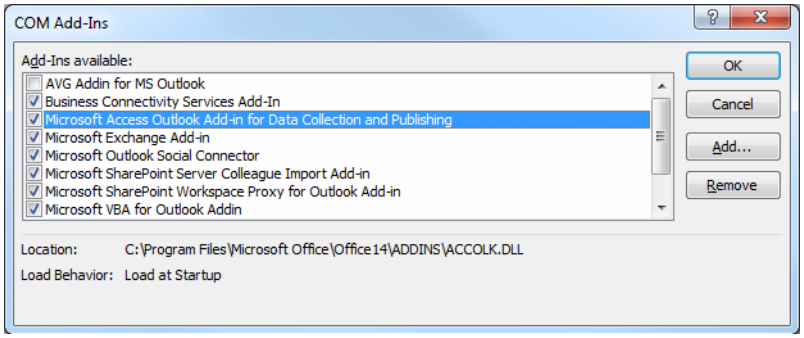


Figure 16-15. *Enabling the data collection add-in*

Setting up Data Collection

Now you're ready to run the data collection wizard. This will guide you through a series of dialogs and allows you to configure the details of the data collection process.

1. Select the Survey table from the Navigation pane. Then, in the External Data tab, click the Create E-mail button shown in Figure 16-16.

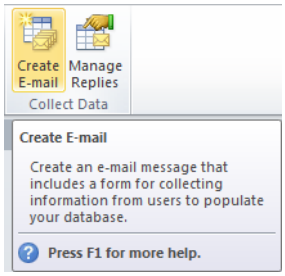


Figure 16-16. *Starting the data collection wizard*

2. This will launch the Data Collection wizard. The first dialog box, shown in Figure 16-17, provides an overview of the process. Click the Next button to continue.

www.it-ebooks.info

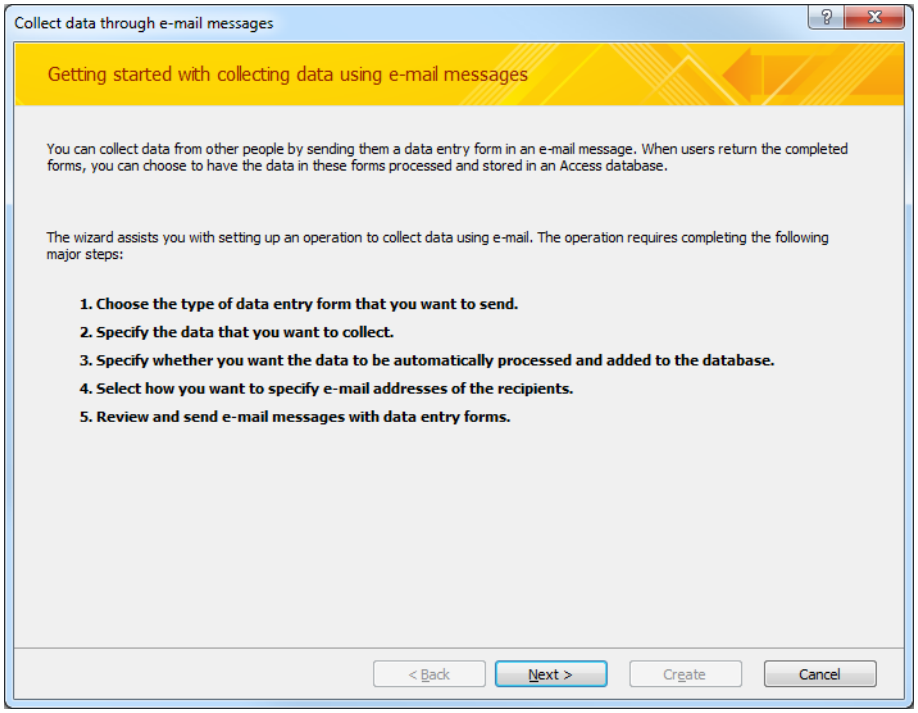


Figure 16-17*. The initial dialog box of the Data Collection wizard*

3. The second dialog box allows you to choose the type of form that will be included in the e-mail. You can either use standard HTML or you can use InfoPath to design the form. Using InfoPath to design the form will give you a lot more flexibility and you can provide a much better user experience. This will require that each recipient have Outlook and InfoPath installed so this option is only viable for internal e-mails. Select the HTML option as shown in Figure 16-18.

www.it-ebooks.info

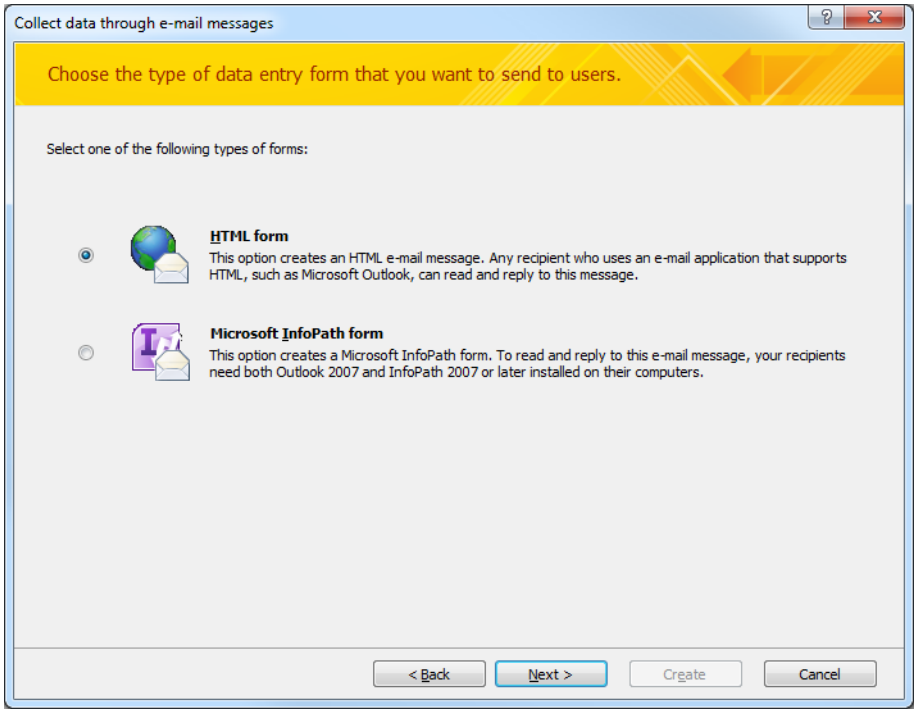


Figure 16-18. *Selecting the HTML form type*

4. In the third dialog box, select the `MediaID` and `CategoryID` fields to be included in the form, as shown in Figure 16-19.

5. For each field, enter some text in the Label property. This text will be included in the form. None of these fields will be read-only, so leave the Read-Only checkbox unselected.

www.it-ebooks.info

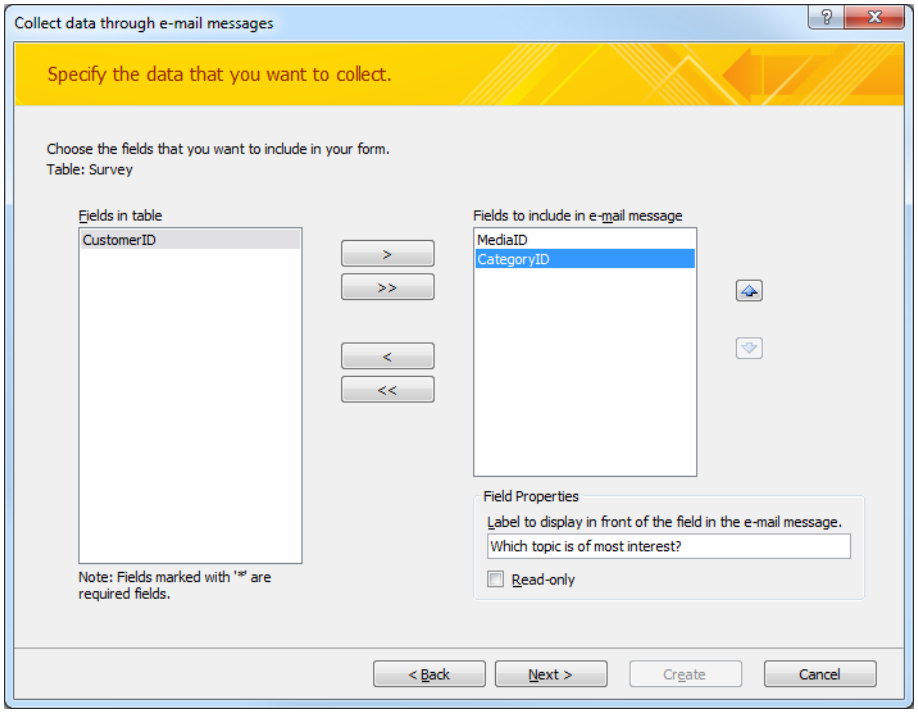


Figure 16-19*. Selecting the fields to inlcude*

6. In the fourth dialog box, select the "Automatically process replies and add data to Survey" check box, as shown in Figure 16-20.

www.it-ebooks.info

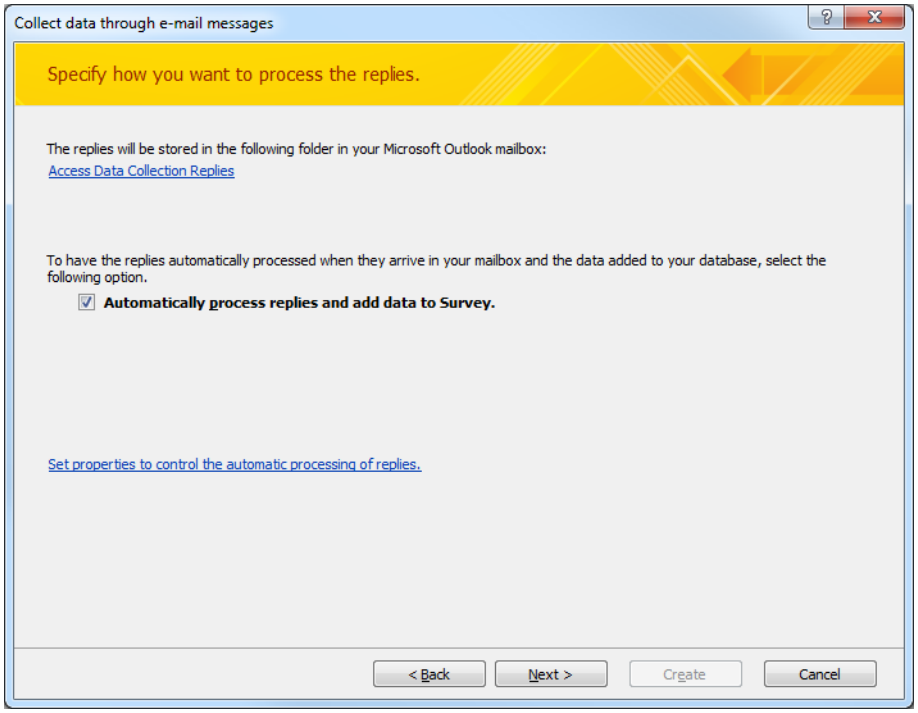


Figure 16-20*. Select the option to automaticaly process replies*

7. In the fifth dialog box, shown in Figure 16-21, you will indicate that e-mail addresses will come from your Access database. As I mentioned, for a small number of recipients you could choose the Outlook option. In this case Outlook would then prompt you to enter the addresses in the To field.

www.it-ebooks.info

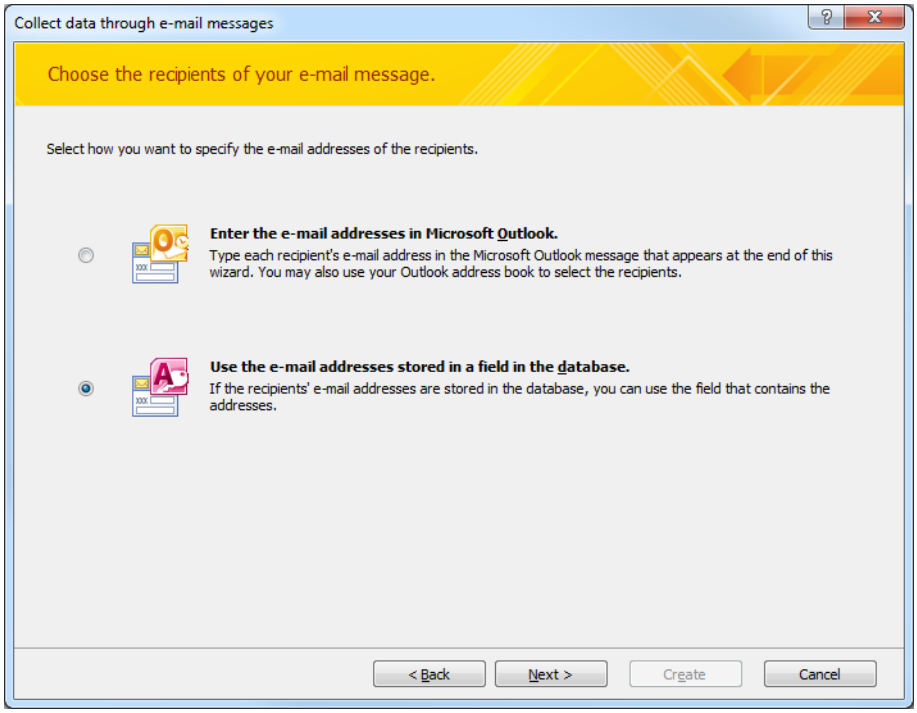


Figure 16-21*. Choosing to get the e-mail addresses from Access*

8. Because you chose to get the e-mail addresses from Access, the sixth dialog box allows you to select where these will come from. You can either select the current table, which is the results table (`Survey` in this case) or a table associated with this table. Select the "An associated table" option.

9. Then select the field in the result table (`Survey`) that links the associated table, which is the `CustomerID` field.

10. The dropdown list with then show the possible fields from the `Customer` table. Select the `Email` field as shown in Figure 16-22.

www.it-ebooks.info

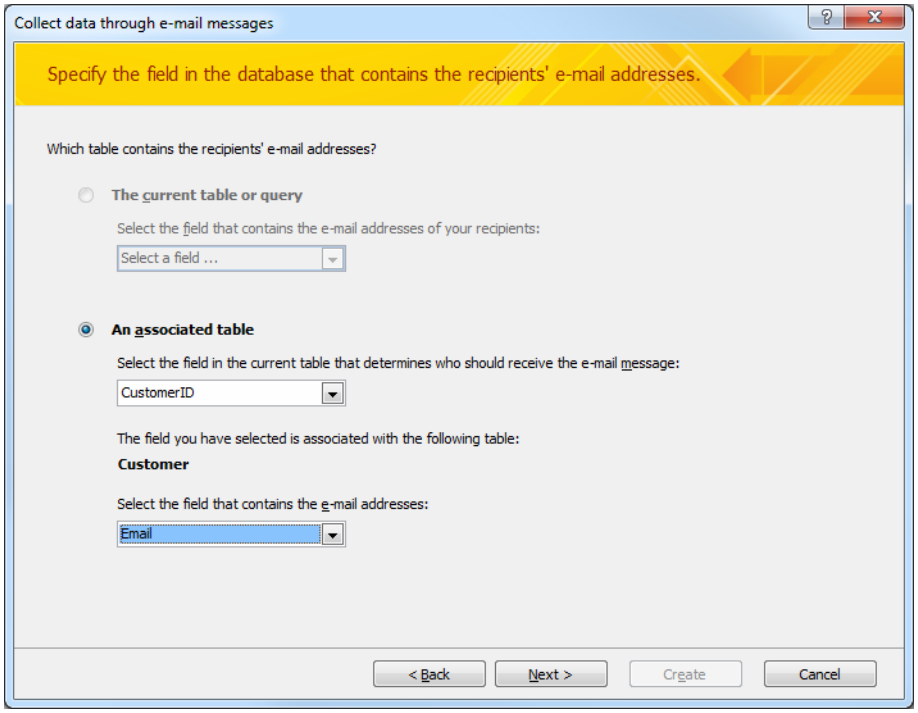


Figure 16-22. *Selecting the source of the e-mail addresses*

11. In the seventh dialog box you'll enter the subject and introductory text that will be included with the e-mail. Enter **Product Survey** for the Subject field and some appropriate text for the Introduction field. You can use the text shown in Figure 16-23 or make up something yourself.

www.it-ebooks.info

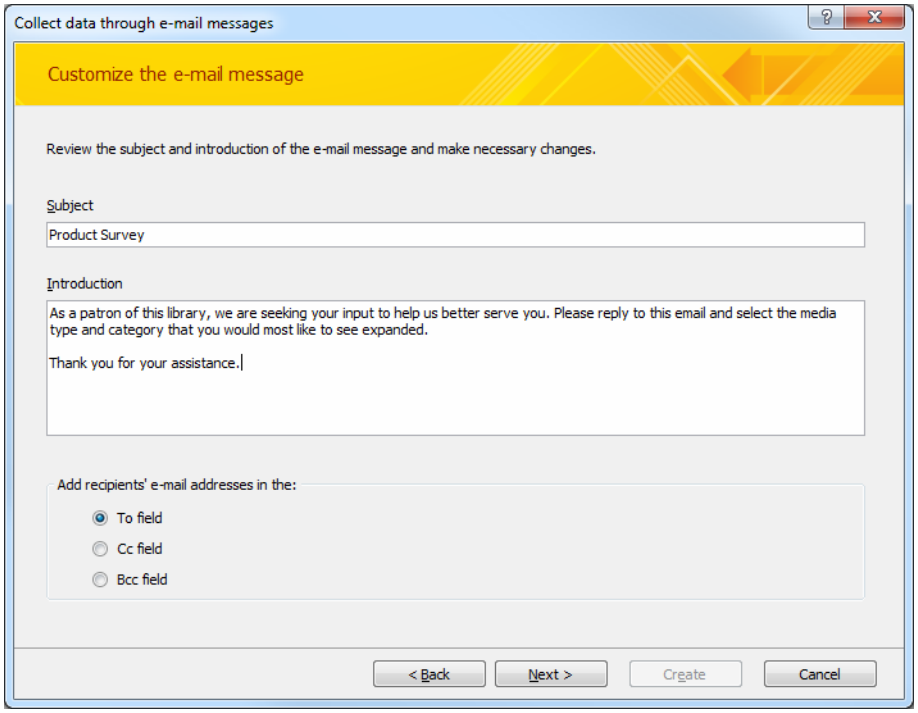


Figure 16-23*. Configuring the e-mail mesage*

12. The eighth dialog box, shown in Figure 16-24, provides an explanation of how the replies will be processed. This dialog can also display warning messages. In this case, only two of the records in the `Customer` table have an e-mail address. The e-mail is configured to go to all customers so a warning is displayed letting you know that some intended recipients will not receive the e-mail because the Access database does not have a valid e-mail defined for them. Click Next to continue.

www.it-ebooks.info

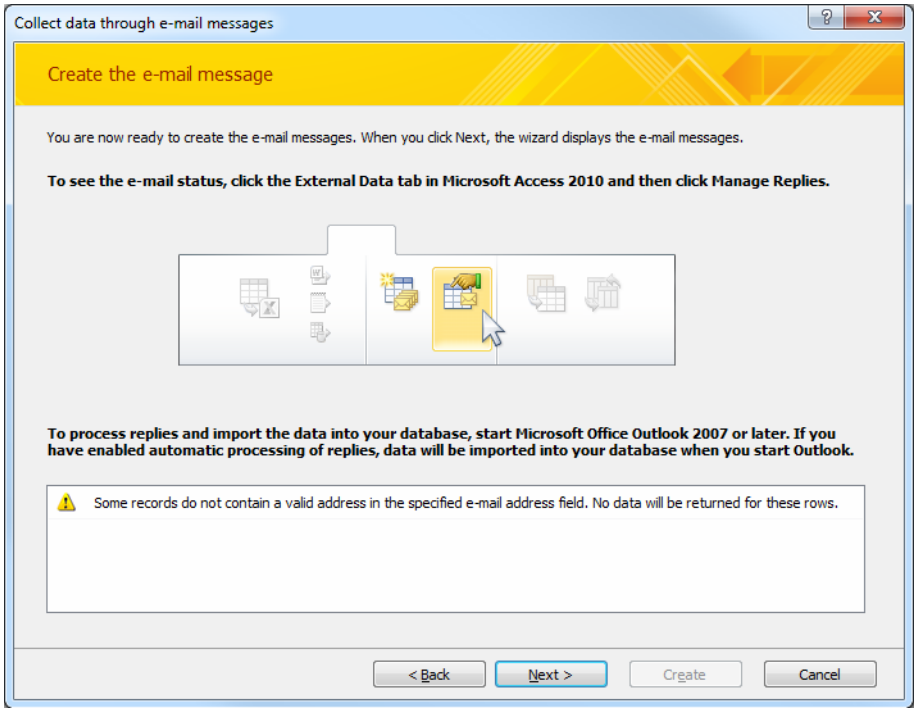


Figure 16-24. *Reply processing information*

13. The final dialog box, shown in Figure 16-25, shows the e-mail addresses that will be mailed. You have the option here to unselect one or more addresses if you chose to not send an e-mail to them. Click the Send button to send the e-mails and start the collection process.

www.it-ebooks.info

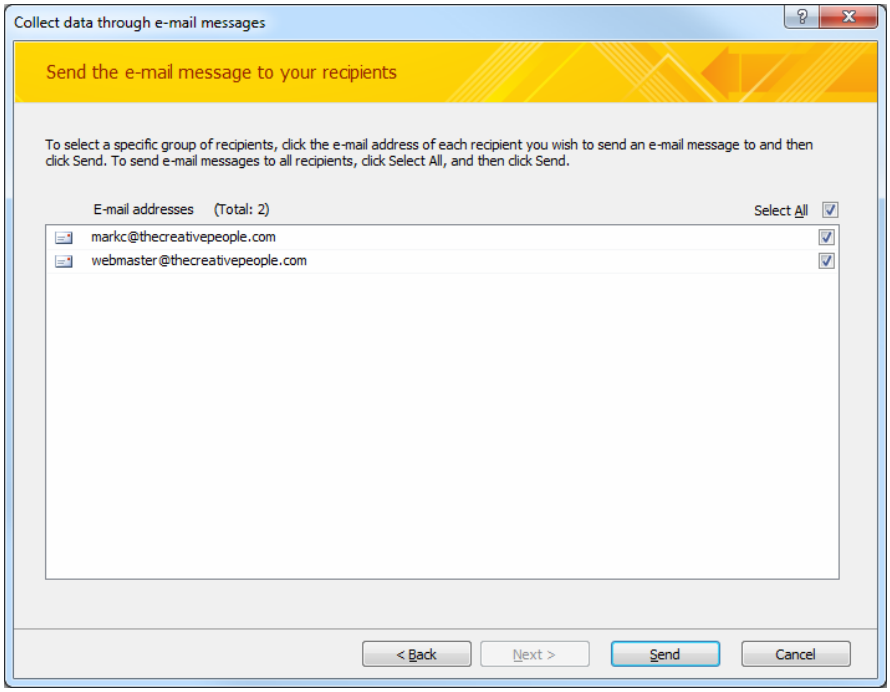


Figure 16-25. *Sending the e-mails*

Processing the Replies

At this point, the e-mails have been sent to the desired recipients. You can check your Sent folder to verify that they were sent. When the message arrives in your Inbox, each recipient should click the reply button, fill in the form and send the response. Outlook will receive the replies and store the data in the `Survey` table automatically.

■ **Note** I'm testing this using a single e-mail box, as you probably are as well. With all the different e-mails going back and forth, I want to make sure you understand what will happen in a "live" scenario. You, as the originator, will have the original outgoing e-mails in your Sent folder. The replies that come back to you will be placed in a special Outlook folder that I will explain later. (You must run Outlook on the originator's machine to process the replies.) Each recipient will receive an e-mail in their Inbox. When they reply, they will also have the response e-mail in their Sent folder.

When you receive an e-mail in your Inbox, click the Reply button and enter a `MediaID` and `CategoryID`. The e-mail will look like Figure 16-26.

www.it-ebooks.info

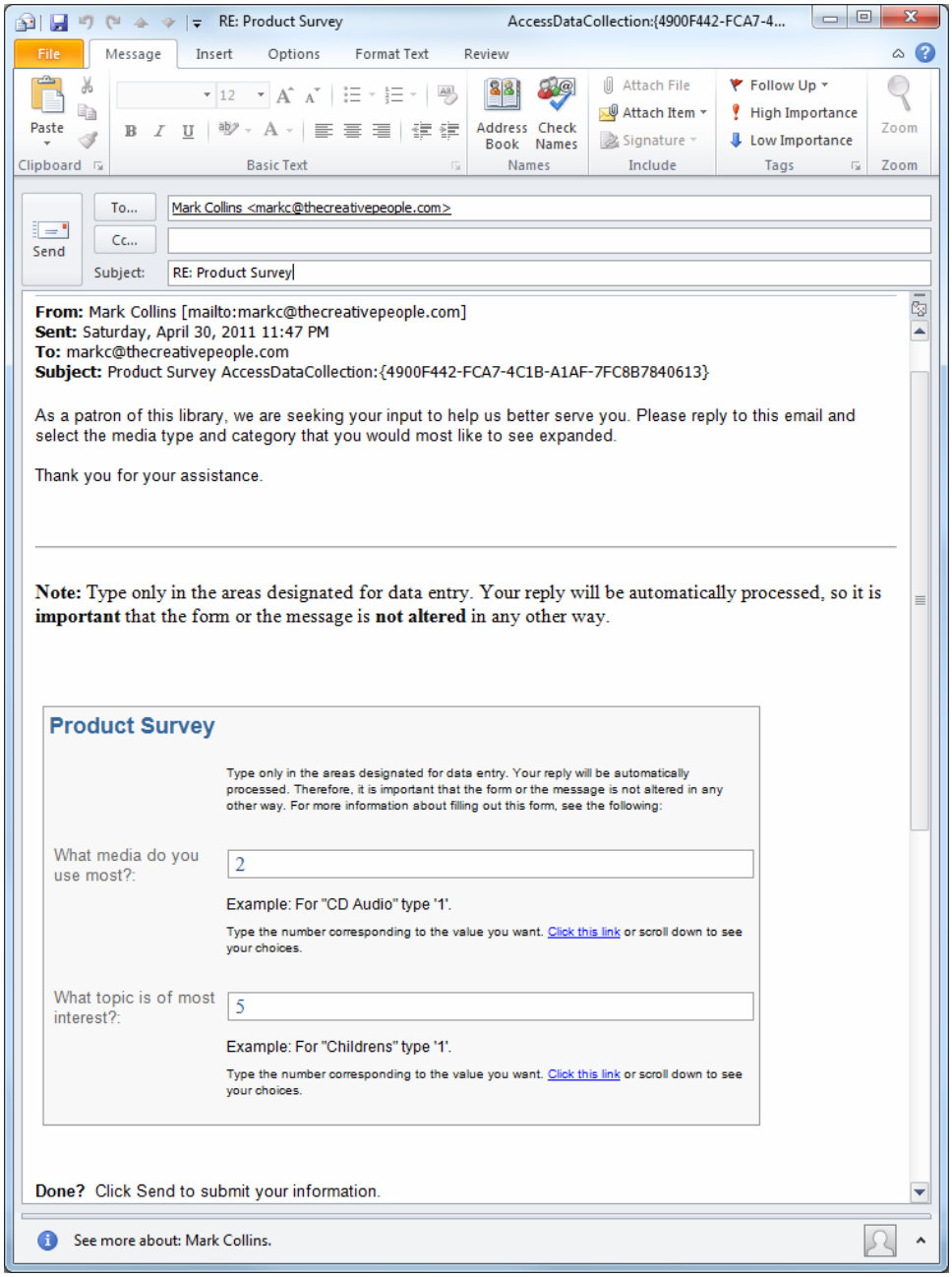


Figure 16-26. A sample response e-mail

www.it-ebooks.info

The fields used for data collection were ID fields (`MediaID` and `CategoryID`) so the form is expecting the recipient to enter a numeric value for these IDs. However, the HTML form does not have the ability to include a dropdown list. The form tries to help the user by displaying the allowed values and their meanings. If you scroll down to the bottom of the e-mail you'll see the legend that is provided as demonstrated in Figure 16-27.

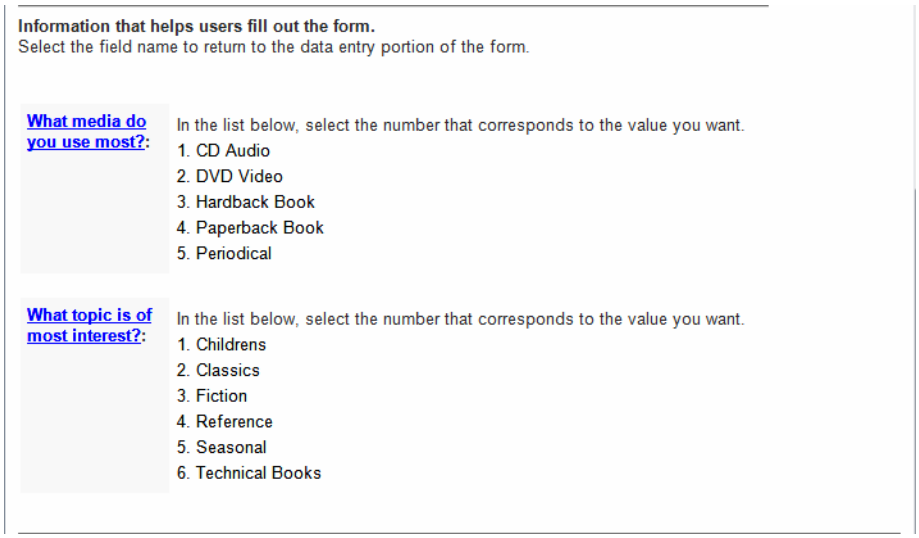


Figure 16-27. *The legend for the* `MediaID` *and* `CategoryID` *fields*

Admittedly, this is a bit awkward. The InfoPath form will provide a much better user experience, but is not practical for many scenarios, such as this one. The average user is not likely to have InfoPath installed and many will not use Outlook either.

As each recipient sends the form back, the replies will be placed in a folder called Access Data Collection Replies. This folder has a special column called Data Collection Status. These should all have the value "Collecting data using e-mail was successful" as shown in Figure 16-28.

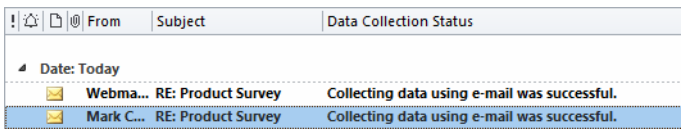


Figure 16-28. *The Access Data Collection Replies folder*

■ **Tip** If the status is "Collecting data not processed" you can right-click the item then click the *Export data to Microsoft Access* link. They should be processed automatically but I've seen instances where the synchronization has not occurred yet and this is how you can process them manually if you don't want to wait.

www.it-ebooks.info

Once the replies have been processed, go back to Access and open the Survey table. You should see a record for each reply as shown in Figure 16-29.

Survey

SurveyID	MediaID	CategoryID	CustomerID
1	DVD Video	Seasonal	
2	Hardback Book	Reference	

Figure 16-29*. The Survey table populated with e-mail data.*

Resending the E-mail

Once you have set up a data collection e-mail, you can easily repeat this process to send it again to the same list of addresses or to a different list. From the External Data tab of the ribbon, click the Manage Replies button. This will display the dialog box shown in Figure 16-30.

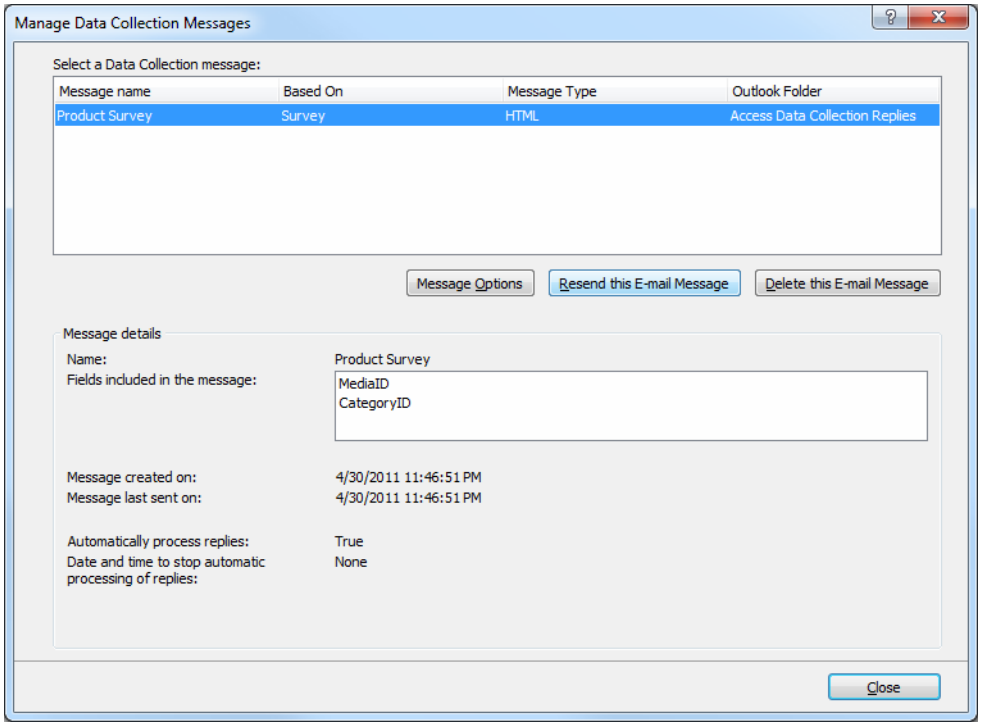


Figure 16-30*. The Manage Data Collection Messages dialog box*

This shows the data collection e-mails that have been previously configured. Select the Product Survey message and click the "Resend this E-mail Message" button. This will display the dialog box shown in Figure 16-20. Follow this and the next five dialog boxes to resend the e-mails. These are the exact same dialog boxes that you used earlier to send the original e-mails.

www.it-ebooks.info

These dialog boxes allow you to define the recipients and to specify the introduction message. You can send the e-mails to the same people or specify a different list. You can also modify the text, if appropriate. As the responses are processed, they will populate the same `Survey` table.

Summary

In this chapter you looked at two uses of Outlook in conjunction with your Access solution. First, you can use Outlook to send e-mails from the Access application. I showed you two ways to accomplish this. The first, using a data macro and the second, using VBA.

Using a data macro is a pretty simple way to send an e-mail. You used it from a named macro that is called from the UI. However, you can easily add the `SendEmail` into an After Insert, After Update, or After Delete macro to send an instant notification when an event occurs. Also, since web databases cannot use VBA, the macro approach is the best solution for web databases. The VBA approach provides more flexibility, the most significant being the ability to include attachments.

I also showed you how to use the data collection facility to send an e-mail and automatically process the responses. This can be a little tricky to setup and has a few oddities but can be a useful feature in many instances.

www.it-ebooks.info

CHAPTER 17

Using External Data

One of the most prominent features of Access is its ability link or import external data sources. In addition to its own internal tables, it can link to many other types of databases. In the early days of Access, this was accomplished primarily through Open Database Connectivity (ODBC) data sources. Access can still link to any data source that you can obtain an ODBC driver for. This includes SQL Server, Oracle, MySQL, DB2 and more. Access 2010 can also link to Outlook and SharePoint.

In previous chapters, I have already demonstrated linking to other Access and SQL Server databases. In this chapter, I will demonstrate how to link to Outlook and SharePoint. I will also show you how to import an XML file. This is not an exhaustive list of the capabilities of Access 2010, but represents the more common and useful applications of external data sources.

Linking and Importing

To be clear, I will first define the terms *linking* and *importing*, because they are very different but are sometimes used interchangeably. When linking to a data source, the data does not actually reside in the Access database. Instead, it is still in the original data store. When linking to a SQL Server table, for example, the data is in the SQL Server database. Access merely stores the link to that database. When the linked table is queried, Access will, in turn, query the SQL Server database.

The Access forms will treat a linked table as if it were a local table. Access provides this transparency so you can have tables from different sources and even different types of sources, and the application is unaware of the details of where the data actually resides. Another characteristic of linked tables is that the external source may be modified outside of the Access application. If you link to an existing SQL table, for example, there are likely other applications using that table. Your Access database becomes just another user all sharing the same data repository.

When importing a table, however, you are simply making a copy of the existing data and storing it into a local table. Subsequent changes to the external data are not reflected in the Access table. Likewise, changes made in Access are not propagated to the external source. While this may seem like a significant limitation, importing an external data source has a practical application in many scenarios. I will demonstrate one of these later in the chapter.

Linking a SharePoint List

In Chapter 15, you published your `Library` application to a SharePoint site. The process of publishing the database created new a SharePoint list for each table, copied the data to SharePoint and converted your Access file to a web database with links to the SharePoint lists. In this chapter, however, you'll use a client database (not published to the web) and link to a single SharePoint list. This is a handy technique

www.it-ebooks.info

for accessing SharePoint data. You won't be able to publish web forms, but you will be able integrate the existing SharePoint data into your client application.

Creating a Team SharePoint Site

For this exercise, I created a new SharePoint site using the standard Team Site template, which creates some basic lists such as a `Tasks` list, a `Shared Documents` library, and a `Calendar`. You will link to the `Tasks` list in the Access database. If you don't already a SharePoint site with the standard `Tasks` list you can easily create one.

Go to the SharePoint site and from the Site Actions menu, click the *New Site* link. In the Create dialog box, select the Team Site template and enter **Team** for the name and URL as shown in Figure 17-1.

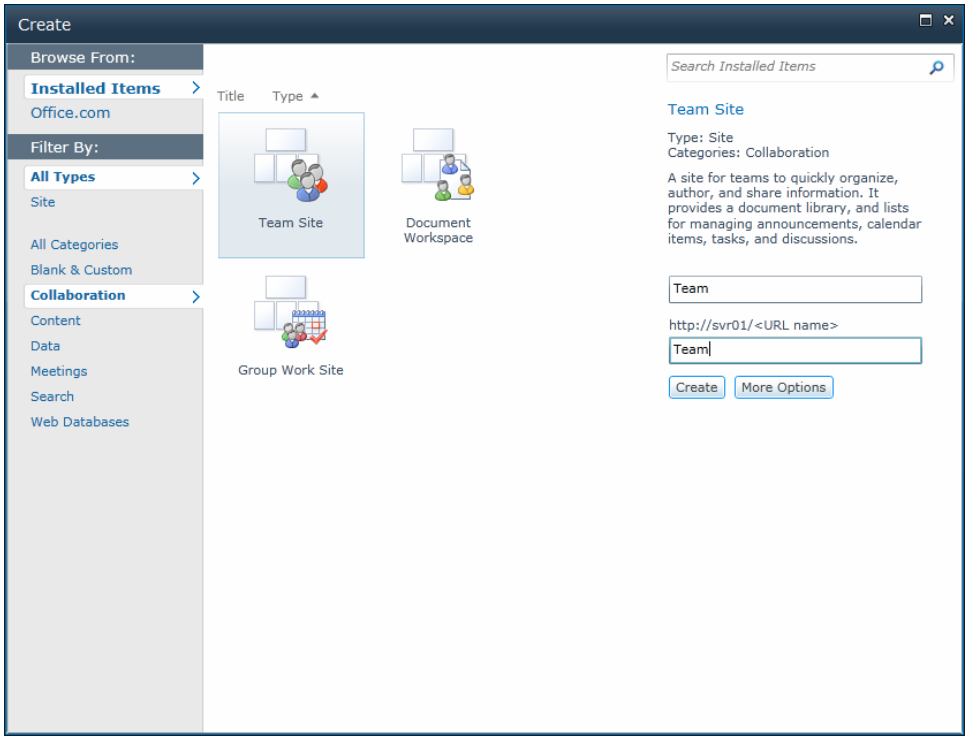


Figure 17-1*. Creating a SharePoint site using the Team Site template*

Go to the All Site Content page, shown in Figure 17-2, to see the lists and libraries that were created for you. (From the Site Actions menu, click the *View All Site Content* link.)

www.it-ebooks.info

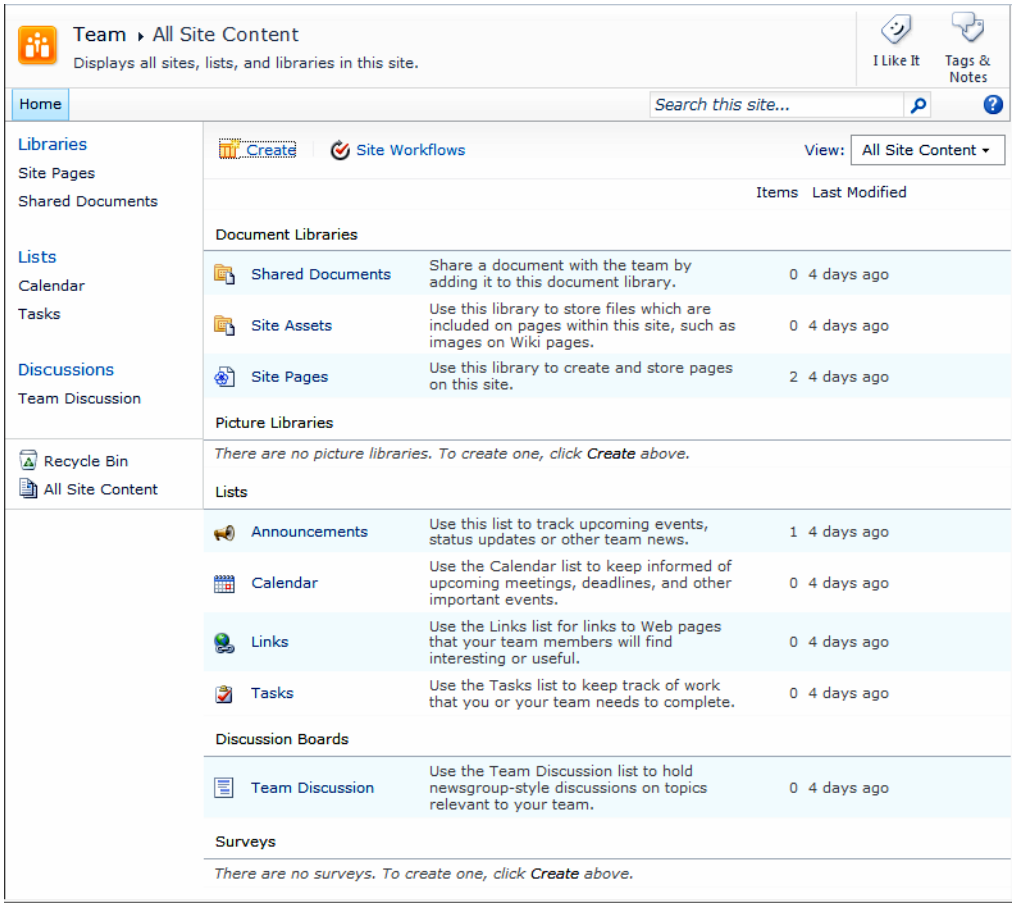


Figure 17-2. *The All Site Content page*

Go to the **Tasks** list and add a few items so you'll have some data to work with. The **Tasks** list will look similar to Figure 17-3.

Type	Title	Assigned To	Status	Priority	Due Date	% Complete	Predecessors
	Finish coding the data entry form NEW	INTERNAL\mark	In Progress	(1) High	5/3/2011		
	Complete the documentation project NEW	INTERNAL\mark	Not Started	(2) Normal	5/6/2011		
	Review the design documents NEW	INTERNAL\mark	Completed	(2) Normal	4/26/2011	100 %	

Figure 17-3. *Displaying the content of the* **Tasks** *list*

www.it-ebooks.info

Creating a Linked Table

In this chapter, you'll start with a blank Access database as these exercises are not related to the existing `Library` application.

1. Open Access 2010 and select the Blank Database template. Enter the File Name **Chapter17.accdb**, as shown in Figure 17-4, then click the Create button.

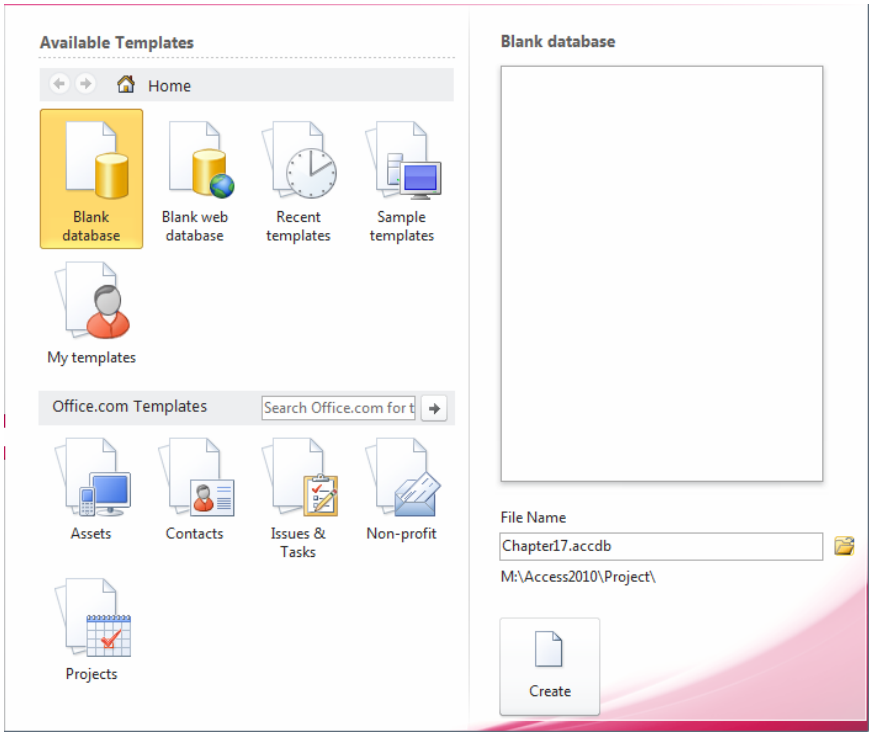


Figure 17-4. *Creating a blank Access database*

2. The template will create a `Table1` table. Close this table without saving it.
3. From the External Data tab of the ribbon, click the More button in the Import & Link group and then click the *SharePoint List* link, as shown in Figure 17-5.

www.it-ebooks.info

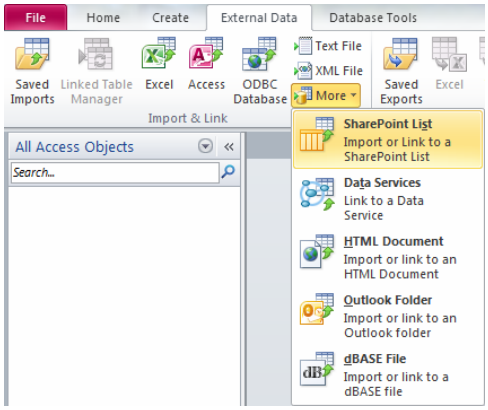


Figure 17-5. Selecting the SharePoint List link

4. In the first dialog you'll need to specify the SharePoint site that will be used. Enter the URL of the `Team` site that you just created (or an existing site that you want to use). Select the Link option as shown in Figure 17-6.

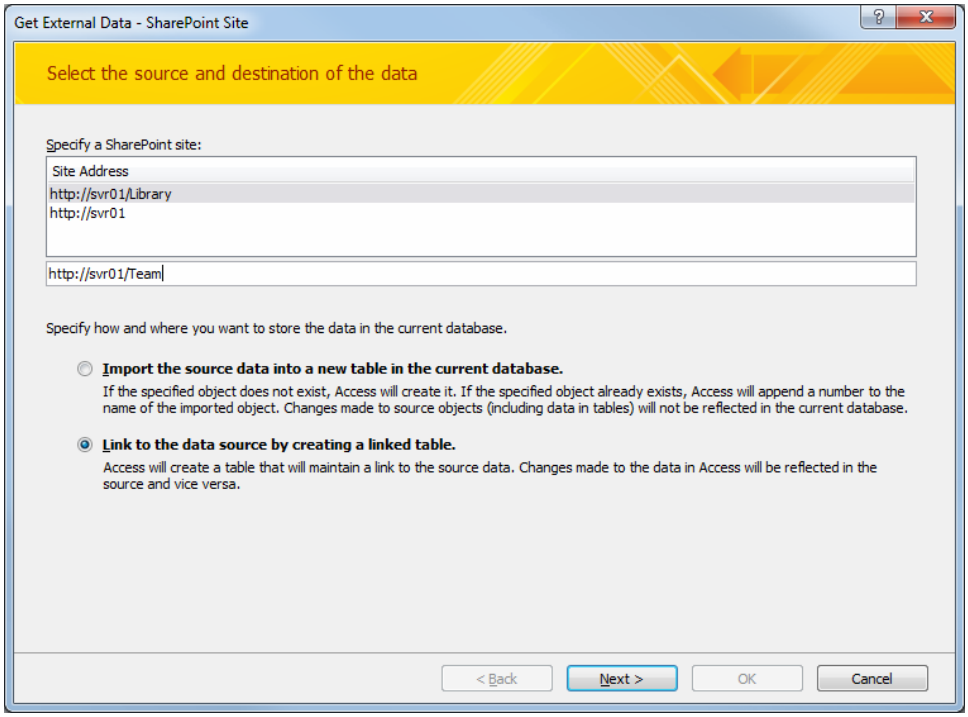


Figure 17-6. Specifying the SharePoint site to which you want to link

www.it-ebooks.info

5. In the second dialog box, you can select one or more lists that you want to link to. Just select the `Tasks` list as shown in Figure 17-7.

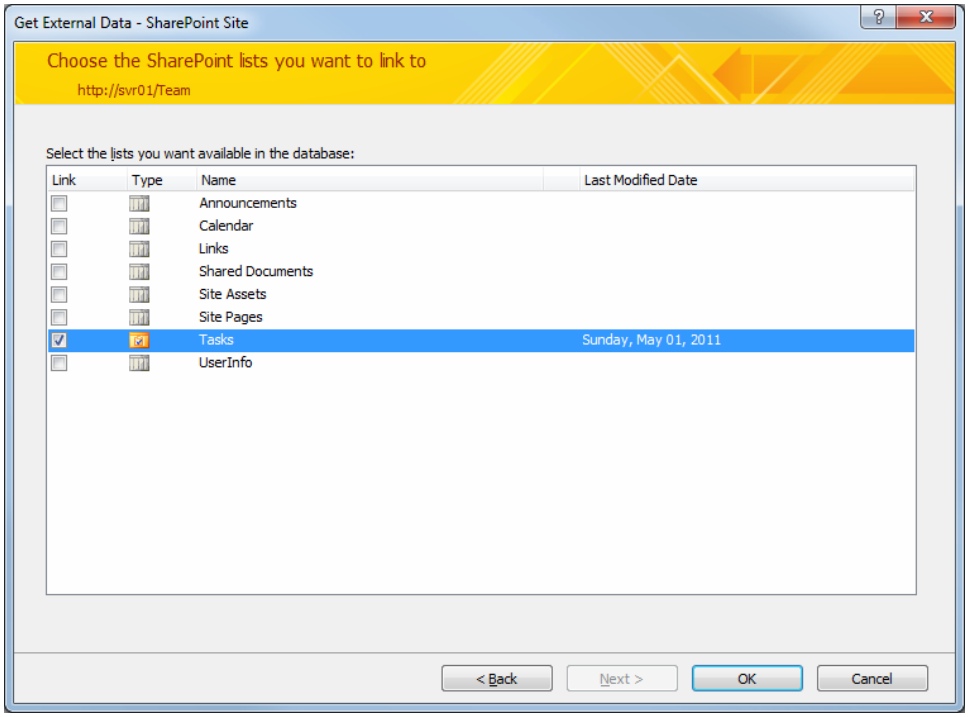


Figure 17-7. *Selecting the list that will be linked to your Access database*

6. Click the OK button to create the linked table.

When the linking has finished you should notice that there are two linked tables: `Tasks` and `UserInfo`. The `UserInfo` list was also linked even though you didn't select it. This was necessary, because the `Tasks` list has a dependency on the `UserInfo` list.

Using the Linked Tables

The contents of these linked tables should look like Figure 17-8 and 17-9.

www.it-ebooks.info

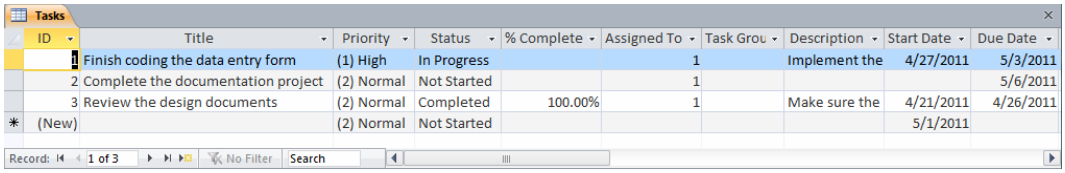


Figure 17-8. The `Tasks` *list*

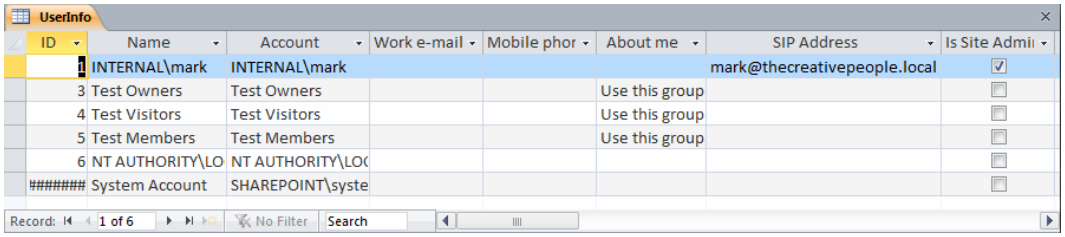


Figure 17-9. The `UserInfo` *lists*

Notice that the `Assigned To` field in the `Tasks` list is a number. This corresponds to the `ID` field of the `UserInfo` table. To make this easier to view, you'll create a query that joins both tables and displays the user's name instead of the ID.

7. From the create tab of the ribbon, click the Query Design button.
8. In the Show Table dialog box add the `Tasks` and `UserInfo` tables.
9. Access should automatically create the join between the tables because of the relationship that was defined when the linked tables were created. However, there is a second join from the `Tasks.Task Group` column that is not needed. Select this join and delete it.
10. Double-click the following fields to add them to the query:
    - `Tasks.ID`
    - `Tasks.Title`
    - `Tasks.Priority`
    - `Tasks.Status`
    - `Tasks.% Complete`
    - `UserInfo.Name`
    - `Tasks.StartDate`
    - `Tasks.DueDate`
11. In the `Status` field, enter **<>"Completed"** for the Criteria property.
12. In the `DueDate` field, select Ascending for the Sort property.

www.it-ebooks.info

13. Save the query and enter the name **OpenTasks** when prompted.

The final design should look like Figure 17-10.

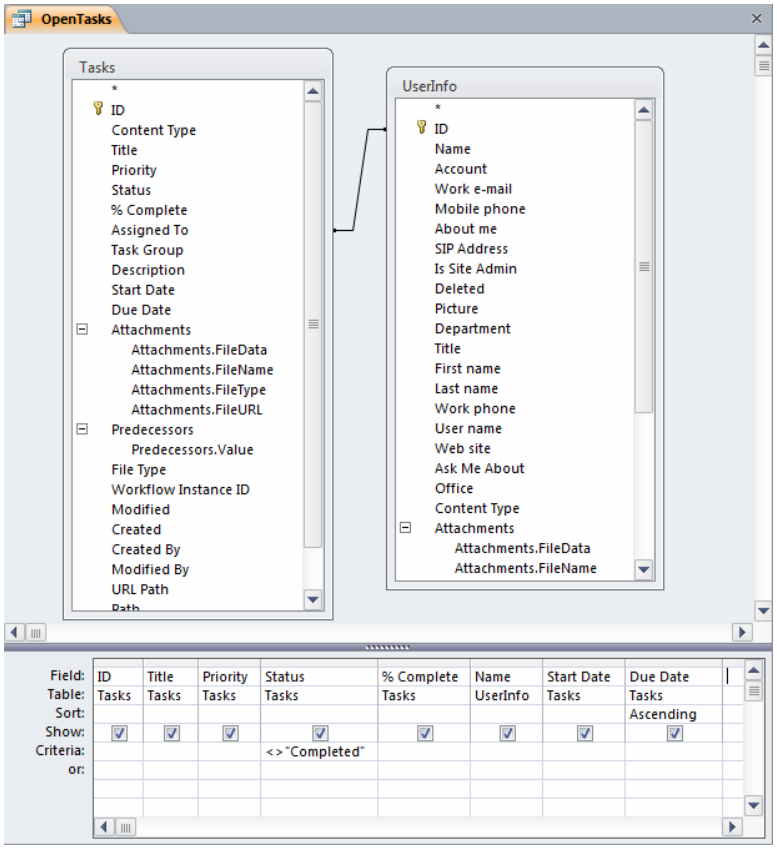


Figure 17-10. *The design of the* `OpenTasks` *query*

Click the Run button in the ribbon, the results should look similar to Figure 17-11.

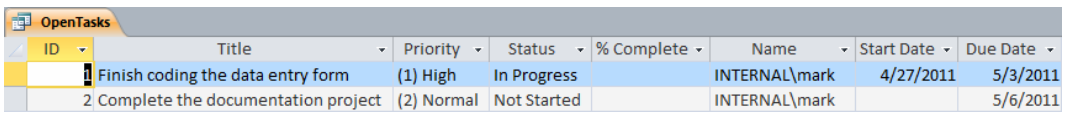


Figure 17-11. *The results of the* `OpenTasks` *query*

www.it-ebooks.info

Linking an Outlook Folder

For the next exercise you'll create a linked table to an Outlook folder. This will allow you to view these e-mails from within your Access application.

Modifying the Data File Name

At the time of this writing, there is a bug in the current release of Office that occurs when attempting to link to an Outlook folder that is stored in a `.pst` file. If the name of the data file is "Outlook," which is the default name, Access is not able to open it. If you try to create the linked table and get an error saying the Outlook is not installed or configured then you are probably affected by this bug. Another symptom of this bug occurs when trying to select the folder and Access cannot expand the folder list.

1. In either case, the work around is fairly simple. In Outlook, go to the backstage view and click the Account Settings button.
2. Select the Data Files tab, which is shown in Figure 17-12.

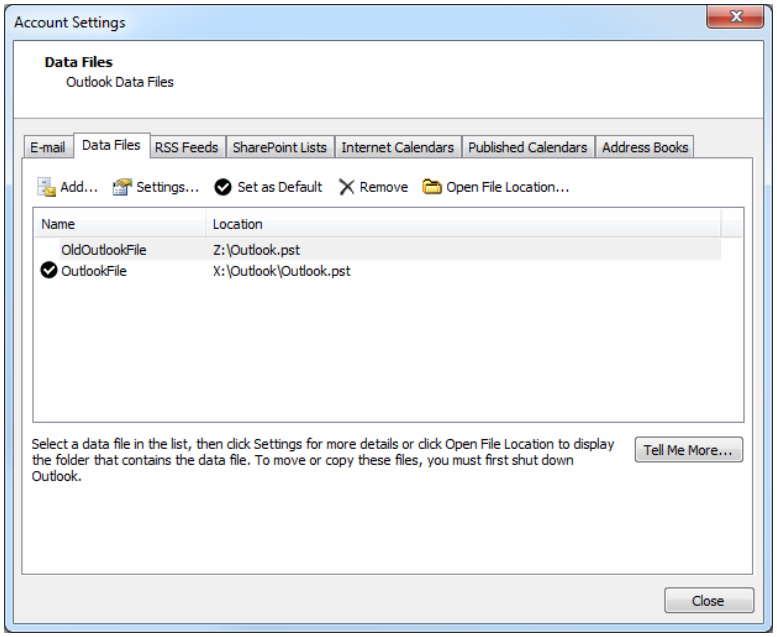


Figure 17-12. *The Data Files tab*

3. Next select the data file and click the Settings link. (I have two data files; if you have multiple files, you should repeat this step for each one.)
4. In the Outlook Data File dialog box, shown in Figure 17-13, change the Name property to something other than "Outlook."

www.it-ebooks.info

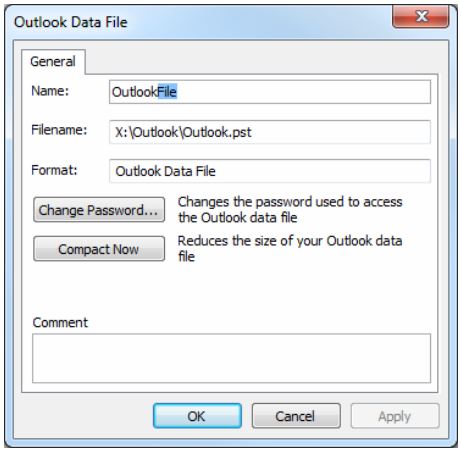


Figure 17-13. *Modifying the Name property of the data file*

You don't need to change the actual filename or location of the `.pst` file; just change the Name property.

Close the dialog boxes.

Creating the Linked Table

With this bug resolved, you're ready to create a linked table that references a folder in Outlook.

1. From the same More button you used to link the SharePoint list, click the *Outlook Folder* link, as shown in Figure 17-14.

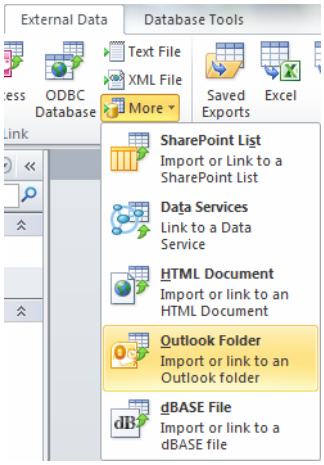


Figure 17-14. *Selecting the Outlook Folder link*

www.it-ebooks.info

2. In the first dialog box you'll need to choose if you want to import the data or link to it. Select the Link option as shown in Figure 17-15.

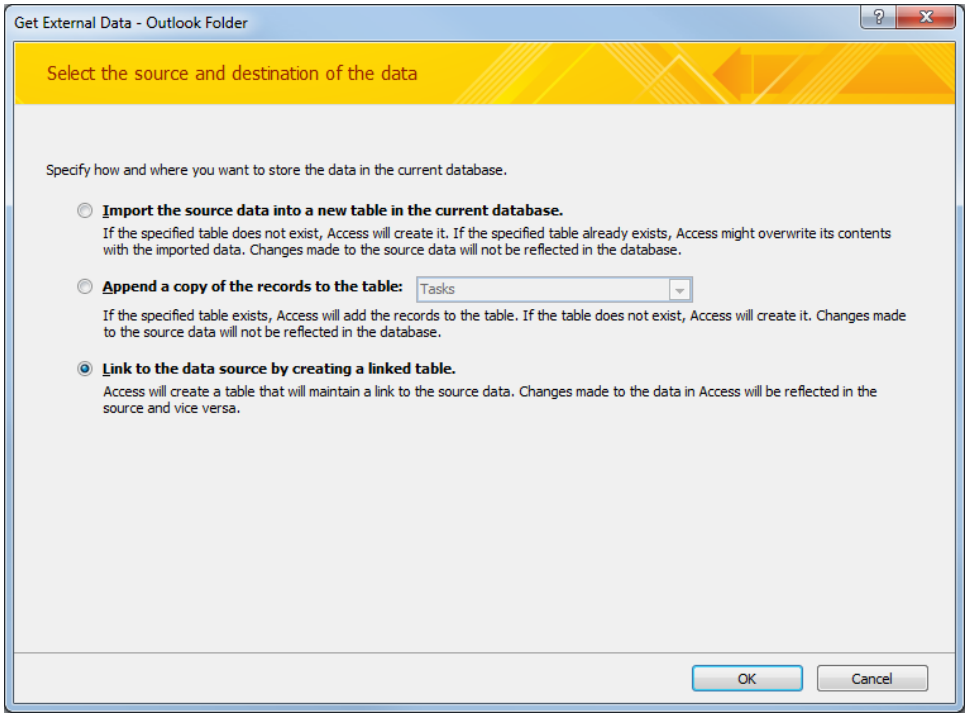


Figure 17-15. Selecting to link the Outlook folder

3. In the second dialog box, expand the folder list, as shown in Figure 17-16, and select a folder to link to.

www.it-ebooks.info

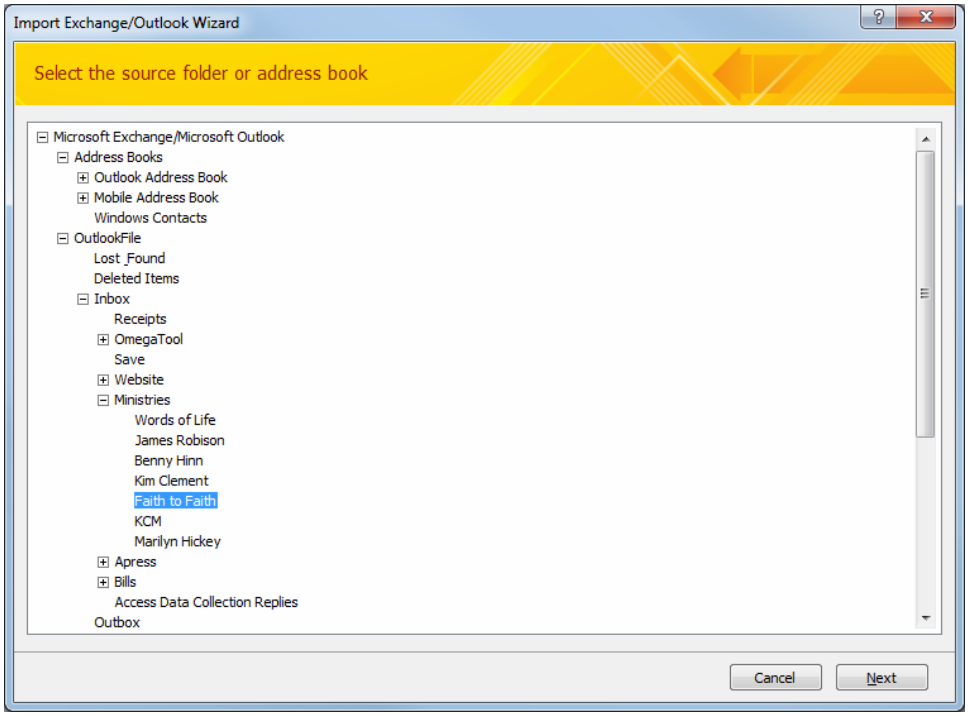


Figure 17-16*. Selecting the folder to create a link for*

■ **Note** Your list of folders will likely be very different from mine. For this exercise, it doesn't matter which folder you choose as long as there are some items in that folder.

---

4. After you select the folder, the final dialog shown in Figure 17-17, prompts you for the name of the linked table. This will default to the name of the selected folder but you can enter a different name for the Access table name.

www.it-ebooks.info

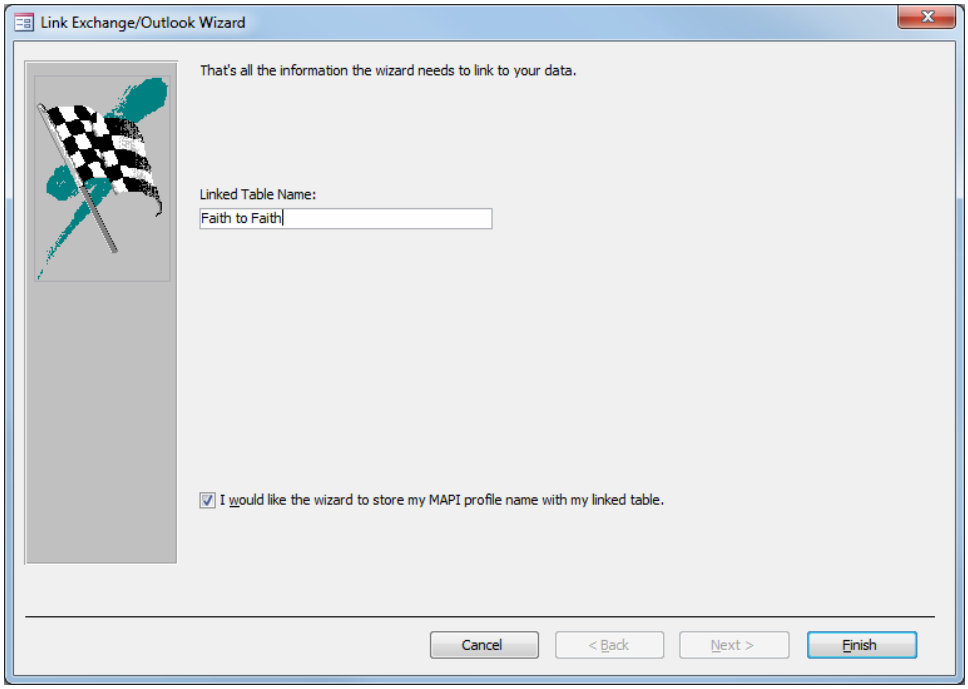


Figure 17-17. *Specifying the name of the linked table*

5. Click the Finish button to setup the linked table. When this is complete, the dialog box shown in Figure 17-18 confirms that the linked table was created successfully.

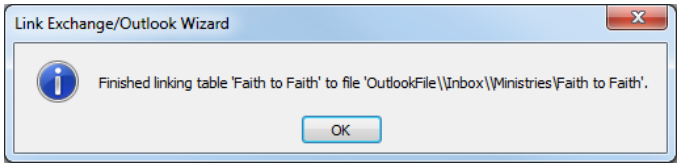


Figure 17-18. *The confirmation dialog box*

Using the Linked Table

Open the linked table using the Design View so you can see the fields that are included. You will see a pop-up window warning you that you will not be able to change the table. Just click the Yes button to continue. The table design should look like Figure 17-19.

www.it-ebooks.info

Faith to Faith

Field Name	Data Type
Importance	Number
Icon	Text
Priority	Number
Subject	Text
From	Text
Message To Me	Yes/No
Message CC to Me	Yes/No
Sender Name	Text
CC	Text
To	Text
Received	Date/Time
Message Size	Number
Contents	Memo
Created	Date/Time
Modified	Date/Time
Subject Prefix	Text
Has Attachments	Yes/No
Normalized Subject	Text
Object Type	Number
Content Unread	Number

Figure 17-19*. The linked table in the Design View*

Close the table. Now you'll create a query to return just the fields that you're interested in. This query will also return only the most recent five items.

1. From the Create tab of the ribbon, click the Query Design button.
2. In the Show Table dialog box, add the linked table that you just created and close the dialog box.
3. Double-click the following fields to add them to your query:
   - `Received`
   - `Subject`
   - `From`
   - `Contents`
4. In the `Received` field, set Sort as Descending.
5. Save the query and enter an appropriate name.

The query design should look like Figure 17-20.

www.it-ebooks.info

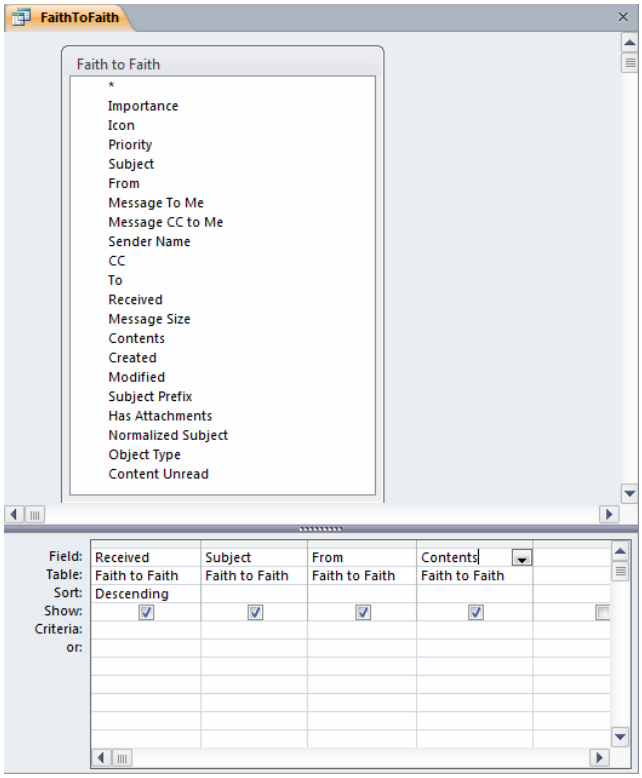


Figure 17-20. *The* `FaithToFaith` *query design*

Switch to the SQL View and enter **top 5** after the SELECT keyword, as shown in Figure 17-21.

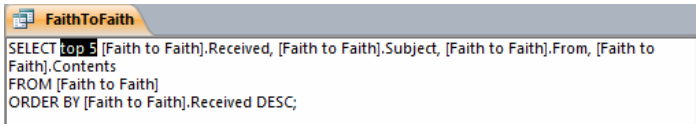


Figure 17-21. *Adding the "*`top 5`*" constraint*

Save the query changes and switch to the Datasheet View. You should see only the last five records in the selected folder.

Importing an XML File

For the next external data source, you will import an XML file. Access does not support linking to an XML file, however, being able to import XML is still a useful feature as I will demonstrate.

www.it-ebooks.info

In this exercise, you will import a file that defines various currency exchange rates. This is a typical example of data that you might receive from an external service. You can setup a subscription to receive the current exchanges rates each day. You can then import this into Access so you'll always have the latest rates in your database.

Understanding the XML Import Process

XML supports a free form data structure. The entity and attribute names are not predefined so they can be expanded as required. As long as the sender and receiver agree on the interpretation of these elements everything works great. The definition of the XML content is called a *schema*. XML also allows for hierarchical data structures. Consider, for example, the simple XML file shown in Listing 17-1.

Listing 17-1. *Sample* `regions.xml` *File*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<regions>
  <Country><Name>United States</Name>
    <States>
      <State><Name>Alaska</Name><Code>AK</Code>
        <Cities>
          <City><Name>Juneau</Name></City>
          <City><Name>Fairbanks</Name></City>
        </Cities>
      </State>
      <State><Name>Alabama</Name><Code>AL</Code>
        <Cities>
          <City><Name>Montgomery</Name></City>
          <City><Name>Mobile</Name></City>
        </Cities>
      </State>
    </States>
  </Country>
  <Country><Name>Canada</Name>
  </Country>
</regions>
```

This file defines two countries (United States and Canada). The United States element also defines two states (Alaska and Alabama). Each of these states also define two cites (Juneau and Fairbanks) and (Montgomery and Mobile), respectively.

If you were to import this XML file in to Access, the import would create three tables (`Country`, `State`, and `City`). As you would expect, the `Country` table would have two records (United States and Canada); the `State` table would have two records (Alaska and Alabama) and the `City` table would have four records. What you may not expect, however, is that the import process does not retain any of the contextual information. For example, there is no indication as to which country each state belongs to.

■ **Caution** In my opinion, this is a pretty serious limitation. If your XML data is hierarchical and that hierarchy is important to you, then you should not use the XML import feature.

---

www.it-ebooks.info

Importing the Exchange Rates

Fortunately for this project, the exchange rate data does not have hierarchical data. The content of the XML file is shown in Listing 17-2.

Listing 17-2. *The Contents of* ***currency.xml***

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ExchangeRates>
<currency><csymbol>AUD</csymbol>
  <cname>Australia Dollars</cname><crate>0.9116185788</crate>
</currency>
<currency><csymbol>CAD</csymbol>
  <cname>Canada Dollars</cname><crate>0.9452000000</crate>
</currency>
<currency><csymbol>EUR</csymbol>
  <cname>Euro</cname><crate>0.6753106429</crate>
</currency>
<currency><csymbol>GBP</csymbol>
  <cname>United Kingdom Pounds</cname><crate>0.5986589954</crate>
</currency>
<currency><csymbol>GHS</csymbol>
  <cname>Ghana Cedis</cname><crate>1.5039999485</crate>
</currency>
<currency><csymbol>ILS</csymbol>
  <cname>Israel New Shekels</cname><crate>3.3960000000</crate>
</currency>
<currency><csymbol>INR</csymbol>
  <cname>India Rupees</cname><crate>44.2400016785</crate>
</currency>
<currency><csymbol>JPY</csymbol>
  <cname>Japan Yen</cname><crate>81.1700000000</crate>
</currency>
<currency><csymbol>MXN</csymbol>
  <cname>Mexico Pesos</cname><crate>11.4841003418</crate>
</currency>
<currency><csymbol>NZD</csymbol>
  <cname>New Zealand Dollars</cname><crate>1.2347203575</crate>
</currency>
<currency><csymbol>PHP</csymbol>
  <cname>Philippines Pesos</cname><crate>42.7900009155</crate>
</currency>
<currency><csymbol>RUB</csymbol>
  <cname>Russia Rubles</cname><crate>27.4055000000</crate>
</currency>
<currency><csymbol>TTD</csymbol>
  <cname>Trinidad and Tobago Dollars</cname><crate>6.3299999237</crate>
</currency>
<currency><csymbol>USD</csymbol>
  <cname>United States Dollars</cname><crate>1.0000000000</crate>
</currency>
```


www.it-ebooks.info

```
<currency><csymbol>ZAR</csymbol>
  <cname>South Africa Rand</cname><crate>6.5669350000</crate>
</currency>
</ExchangeRates>
```

1. You can download this file from `www.apress.com`. Save this on your local machine.

2. From the External Data tab of the ribbon, click the XML File button, as shown in Figure 17-22.

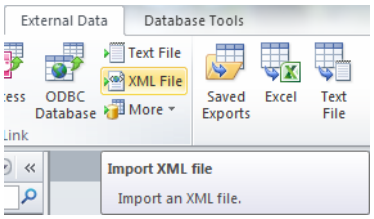


Figure 17-22. *Selecting to import an XML file*

3. In the first dialog box specify the `currency.xml` file as the source file to be imported, as shown in Figure 17-23.

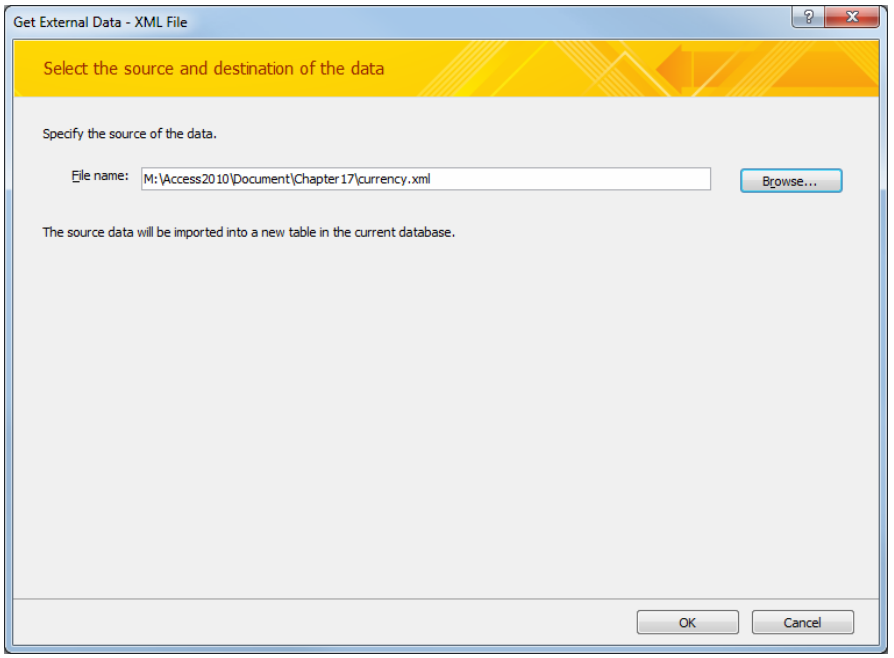


Figure 17-23. *Specifying the source file*

www.it-ebooks.info

4. The second dialog box, shown in Figure 17-24, lists the tables that will be created. For this file there is only a single table, `currency`. Click OK.

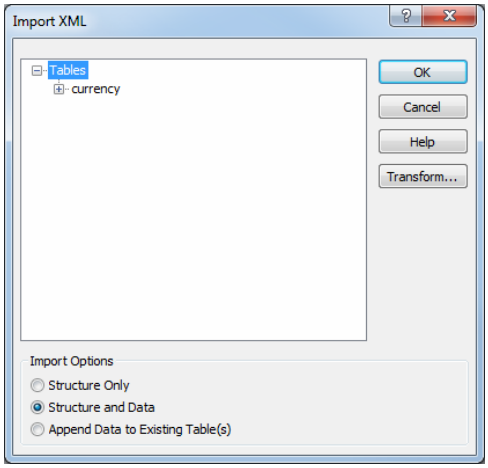


Figure 17-24. *Listing the tables that will be created*

■ **Note** The table and column names are based on the element names in the source file.

---

5. The third dialog box provides the option to save the details of this import so it can be easily re-imported later. You can also set up a task in Outlook to remind you to import this file. Select the "Save import steps" check box and click the Save Import button as shown in Figure 17-25.

www.it-ebooks.info

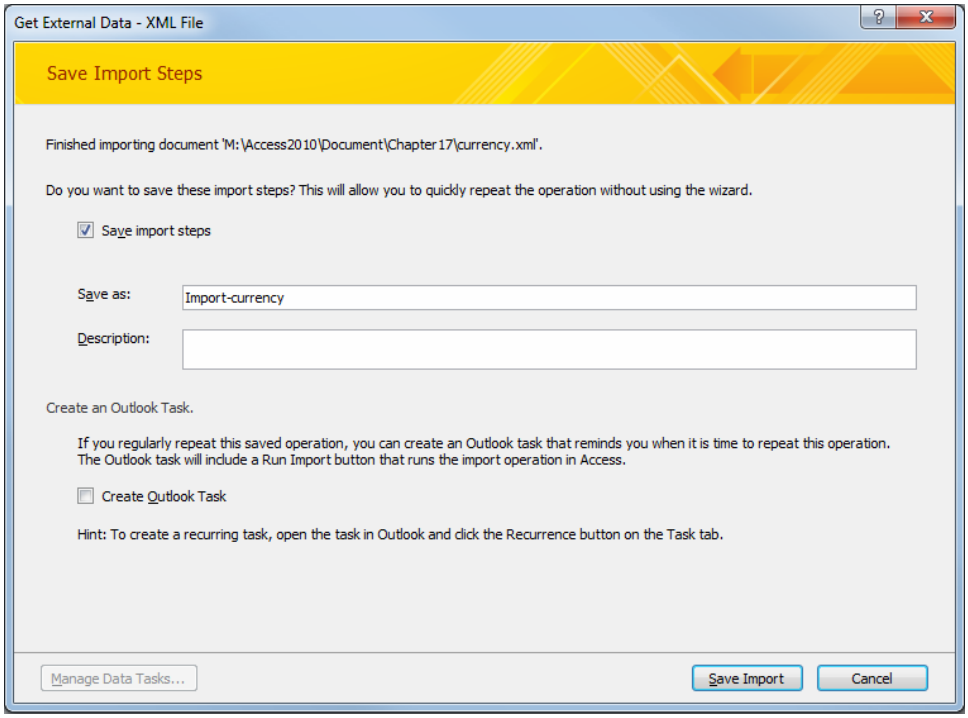


Figure 17-25. *Saving the import details*

Now open the currency table using the Datasheet View. The data should look like Figure 17-26.

currency

csymbol	cname	crate
AUD	Australia Dollars	0.9116185788
CAD	Canada Dollars	0.9452000000
EUR	Euro	0.6753106429
GBP	United Kingdom Pounds	0.5986589954
GHS	Ghana Cedis	1.5039999485
ILS	Israel New Shekels	3.3960000000
INR	India Rupees	44.2400016785
JPY	Japan Yen	81.1700000000
MXN	Mexico Pesos	11.4841003418
NZD	New Zealand Dollars	1.2347203575
PHP	Philippines Pesos	42.7900009155
RUB	Russia Rubles	27.4055000000
TTD	Trinidad and Tobago Dollars	6.3299999237
USD	United States Dollars	1.0000000000
ZAR	South Africa Rand	6.5669350000

Figure 17-26. *The contents of the* `currency` *table*

www.it-ebooks.info

Summary

In this chapter you created linked or imported tables in Access that provide external data from different sources. You added:

- A query that shows all of the open tasks in your team's SharePoint Tasks list
- A query that returns the last five emails in the selected Outlook folder
- A table that contains the most recent snapshot of currency exchange rates

The first two queries use a linked table so the data is coming directly from the external source and therefore always has the most recent information. This third uses an imported table and so will require a manual re-import to update the data.

These exercises should give you an idea of how external data can be used by Access and handled just like local data. Providing data from various external sources can significantly improve the usefulness of your Access solution. In the next chapter you'll look at some other techniques to add handy features to your Access application.

www.it-ebooks.info

CHAPTER 18

Miscellaneous Features

In this chapter, I will demonstrate several unrelated techniques that you will undoubtedly find quite useful in one of your future projects. These are the kind of things that you'll file away in your mind and later find them invaluable. I will explain

- Using timers for building a digital clock and for simple animation
- Calling the Windows API to get the current user
- Using `TempVars` to share data between VBA, macros, queries, and reports
- Using the `WebBrowser` control to display HTML content
- Adding charts to a form or report

Using Timers

Timers are very easy to use and have lots of useful applications. In this chapter, I'll demonstrate two of these. The first is to display the current date and time on the Menu form. The time will include the hour, minute, and second, so you'll need to update this at least once a second to keep the display current. The second example will implement some simple animation.

A timer is an object that raises a `Timer` event at a specified interval. In Access, client forms have a built-in timer object. All you need to do is specify the `TimerInterval` property. The `TimerInterval` property determines the delay (in milliseconds) between each event. For example, if you set this to 60,000, an event will fire in 1 minute intervals.

■ **Caution** Timers are only supported on client forms. They do not work in web forms.

■ **Tip** Set the `TimerInterval` to **0** to disable the timer. This will prevent the `Timer` event from firing.

Once you have the timer configured, you just need to implement the event handler. The code in your event handler will be executed at regular intervals and independent of any other activity on the form.

www.it-ebooks.info

■ **Note** The timer will continue to raise events even when the form is inactive. If it is hidden or minimized, for example, or not the currently selected form if using tabbed documents, your event handler will continue to run at the specified interval. If this code only needs to run while the form is active, you should disable the timer while the form is inactive. You can do this with the `Activate` and `Deactivate` event handlers, which I will demonstrate here.

Adding a Digital Clock

For the first exercise, you'll add a simple clock to the Menu form. This will use a Label control to display the date and a second control to display the time. You'll set the timer interval to 1 second. The event handler will get the current date and time and update the corresponding labels.

1. Open the `Menu` form using the Design View.
2. Select the Apress banner graphic and from the Arrange tab of the ribbon, click the Insert Below button. This will create a new cell that is the entire width of the form.
3. Select the new cell and click the Split Horizontally button, which will divide this into two cells.
4. Select the left cell and click the Split Horizontally button again.
5. From the Design tab of the ribbon, click the Label control and then draw a rectangle on the form. Enter **lblDate**. Drag this control to the first cell of the new row.

■ **Note** You don't need a caption for these labels, because they will be set through code later. However, if you don't enter a caption as soon as you create it, Access will remove it from the form.

6. Repeat this step, entering **lblTime** for the Caption and dragging the control to the second cell.
7. In the Other tab of the Property Sheet, set the Name property as **lblDate** and **lblTime**, respectively.
8. Click the View Code button to create a code file for the `Menu` form and open the VBA Editor. Enter the code shown in Listing 18-1 for the implementation of this class.

www.it-ebooks.info

Listing 18-1. *Implementation of the* `Menu` *Form's Event Handlers*

```
Private Sub Form_Load()
    Me.lblDate.Caption = Format(Now(), "dddd dd, mmm-yyyy")
    Me.lblDate.Caption = Format(Now(), "hh:nn:ss")

    Me.TimerInterval = 1000
End Sub

Private Sub Form_Activate()
    Me.TimerInterval = 1000
End Sub

Private Sub Form_Deactivate()
    Me.TimerInterval = 0
End Sub

Private Sub Form_Timer()
    Me.lblDate.Caption = Format(Now(), "dddd dd, mmm-yyyy")
    Me.lblTime.Caption = Format(Now(), "hh:nn:ss")
End Sub
```

The `Form_Load` handler sets the current value of the label controls and then enables the timer for 1 second intervals (1000 milliseconds). The `Form_Activate` handler also enables the timer, while the `Form_Deactivate` handler disables the timer. The combination of these two events allows you to *pause* the clock while the form is not visible to save on processing resources. The `Form_Timer` method is called every second, when the timer expires, and simply displays the current date and time.

Save the code and form changes. To test this, open the `Menu` form using the Form View, which should look like Figure 18-1.

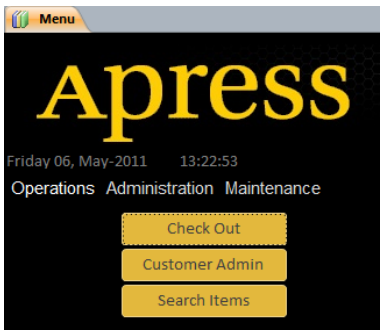


Figure 18-1*. The* `Menu` *form with the current date and time*

Creating Simple Animation

Access does not provide native animation support, like PowerPoint does for example, but you can implement some simple tricks to draw attention to specific UI elements. These techniques rely on using

www.it-ebooks.info

a timer and placing code in the timer event handler to modify one or more control properties. A good example of this is an animated icon. These are implemented by using several images that are displayed in rapid succession giving the appearance that the image is moving.

I will show you how to toggle the `FontBold` property of a Text Box control to simulate a blinking effect. For this demonstration, you'll use the `CustomerLoans` form that you created in Chapter 8. When displaying the due date, if this item is overdue, the form will cause this date to blink, drawing the user's attention to the overdue item.

Open the `CustomerLoans` form using the Design View. In the Design tab of the ribbon, click the View Code button to display the VBA Editor, then add the methods shown in Listing 18-2.

Listing 18-2. *Additional Methods for the* `CustomerLoans` *Form*

```
Private Sub Form_Load()
    Me.TimerInterval = 500
End Sub

Private Sub Form_Activate()
    Me.TimerInterval = 500
End Sub

Private Sub Form_Deactivate()
    Me.TimerInterval = 500
End Sub

Private Sub Form_Timer()
    If (IsNull(CheckedIn) And Not IsNull(DueDate)) Then
        If (DueDate < Now() - 1) Then
            If (DueDate.FontBold) Then
                DueDate.FontBold = False
            Else
                DueDate.FontBold = True
            End If
        Else
            DueDate.FontBold = False
        End If
    End If
End Sub
```

This code is very similar to the clock example you implemented previously, except the interval is set to ½ second (500 milliseconds). The `Form_Timer` event handler is then called twice per second. It first checks for a null `CheckedIn` date and a not null `DueDate`. This serves two purposes:

- If the `CheckedIn` date is not null, the item has already been checked in and we don't need to be concerned about the item being overdue.
- A null `DueDate` indicates that the form is empty (no records to display) and this is a valid scenario. In this case, we need to skip the rest of the logic. Otherwise you'll get an error trying to evaluate the `DueDate`.

So if the item is not checked in yet and there is a due date, the logic determines if the item is overdue. If it is, the `FontBold` property is toggled. Otherwise it is set to `false`.

Let's see this in action.

www.it-ebooks.info

1. Save the code changes and open the `CustomerAdmin` from using the Form View.
2. Select a customer and go to the Items On Loan tab.
3. Select an item that is overdue and you should see the `DueDate` field blinking. The `CustomerLoans` form is also used on this same tab using the Datasheet View. Notice that this change has no effect on the form when displayed using the Datasheet View.
4. Select an item that is not overdue and verify that the `DueDate` field does not blink.

■ **Tip** To find data to test with, open the `Loan` table and look for records that have a null `CheckedIn` field. If there aren't any, use the `CheckOut` form and check out a few items. You can adjust the `DueDate` field, if necessary, so you'll have some that are overdue and some that are not.

Calling the Windows API

Windows provides an API that allows you to access many of its features from your application code. There are literally hundreds of API methods available. You can get a list of the available methods from this site: `http://msdn.microsoft.com/en-us/library/aa383688(v=VS.85).aspx`. Some of the more common methods include `GetComputerName`, `OpenFile`, and `PrintDlgEx`. In this chapter you'll use the `GetUserName` method to get the name of the person that is currently logged in when running the `Library` application. This will be displayed in the `Menu` form.

To use these APIs you'll need to know the function prototype. For example, the `GetUserName` method has the following prototype:

```
BOOL WINAPI GetUserName(
  __out    LPTSTR lpBuffer,
  __inout  LPDWORD lpnSize);
```

To be able to call the method you must first declare it, which tells Access the method name, parameters, return type, and the DLL where the method resides (all of which the MSDN page above can tell you). Alternatively, you can add a reference to the type library as long as one exists and it is correct. If you're just calling one method, as we are here, it's fairly easy to simply declare it in your code.

■ **Tip** For more information about using the Windows API visit this site: `http://visualbasic.about.com/od/usevb6/l/aa103002a.htm`. This is a great article with lots of good advice and helpful links.

---

1. Open the `Menu` form using the Design View. There should be an empty cell to the right of the `lblDate` and `lblTime` labels that you added earlier.

www.it-ebooks.info

2. From the Design tab of the ribbon, click the Label control and then draw a rectangle on the form. Enter **lblUser**. Drag this control to the empty cell. In the Property Sheet set the Name property to **lblUser**.

3. In the Design tab of the ribbon, click the View Code button to display the VBA Editor. Open the `Main` code file and add the following code at the top of the file (just beneath the `Option Compare Database` statement). This code declares the `GetUserName` function. Notice that the declaration also specifies the `advapi32.dll`, which is where the `GetUserName` method is found. Also, `#If Win64` is used to allow different declarations for the 32- and 64-bit versions of Access. The 64-bit version requires the `PtrSafe` attribute.

```
#If Win64 Then
    Private Declare PtrSafe Function GetUserName Lib "advapi32.dll" Alias _
        "GetUserNameA" (ByVal lpBuffer As String, nSize As Long) _
        As Long
#Else
    Private Declare Function GetUserName Lib "advapi32.dll" Alias _
        "GetUserNameA" (ByVal lpBuffer As String, nSize As Long) _
        As Long
#End If
```

4. Add the method shown in Listing 18-3 to the `Main` code file. This code simply calls the API method, `GetUserName`, and trims off any trailing characters.

Listing 18-3. *Implementation of the* `GetUser` *Method*

```
Function GetUser()

Dim lpBuff As String * 500
Dim ret As Long

    ' Get the user name minus any trailing characters
    ret = GetUserName(lpBuff, 50)
    If (ret <> 0) Then
        GetUser = Left(lpBuff, InStr(lpBuff, Chr(0)) - 1)
    Else
        GetUser = ""
    End If

End Function
```

5. Open the `Menu` code file and add the following line to the `Form_Load` method. This code sets the Caption of the `lblUser` control, using the `GetUser` function:

```
Me.lblUser.Caption = GetUser()
```

6. Save the code and form changes and switch to the Form View.

You should now see the user name displayed next to the clock as shown in Figure 18-2.

www.it-ebooks.info

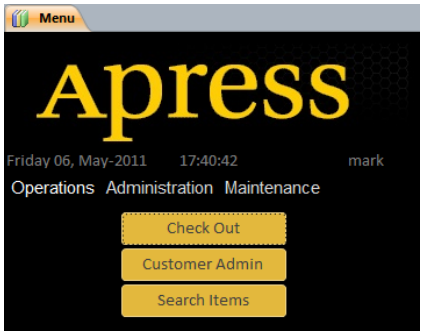


Figure 18-2. *The current user displayed on the* `Menu` *form*

Using TempVars

TempVars are used to share variables across the application. If you're familiar with web applications, they are similar to session variables both in scope and use. You can create a `TempVar` in one form and then access (both read and write) from another. You can access them from within a macro, VBA and queries; just about any place within your application. `TempVars` also retain their values, even if you restart the application.

■ **Caution** The `TempVar` scope is defined by the `Application` object, which includes any of its children such as forms, reports, and queries. The database engine, however, is outside of the application scope. Unfortunately, because data macros are executed by the database engine, they are also outside the application scope. Also if you use the `CurrentDb` object to execute a query, that query will run outside of the application scope. Instead, use the `DoCmd.OpenQuery` to execute the query and it will have access to your `TempVars`.

The application maintains a `TempVars` collection. You can add a `TempVar` to this collection using a macro or VBA code and then access it from other places. For this project, you'll use a `TempVar` to store the name of the current user. You'll modify the `autoexec` macro to set the initial value and then use it in the `Menu` form. To show the power of `TempVars`, you'll also modify the `CheckOut` form to store the user's name when creating new `Loan` records. Now you can tell which user checked out each item.

Creating the UserName TempVar

The first step in using a `TempVar` is to create one and add it to the `TempVars` collection. The perfect place to do that is in the `autoexec` macro, which is called when the Access database is first loaded.

1. From the Navigation pane, right-click the `autoexec` macro and click the *Design View* link. This will start the Macro Editor. Currently, this macro has a `RunCode` action, which is used to load the custom ribbon.

www.it-ebooks.info

2. In the Add New Action dropdown list, select the `SetTempVar` action. For the Name property enter **UserName** and for the Expression enter **GetUser**(). This will call the `GetUser` method that you implemented earlier to call the Windows API. The modified macro should look like Figure 18-3.

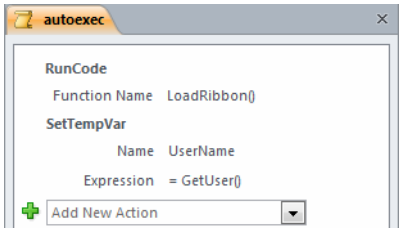


Figure 18-3. *The modified* `autoexec` *macro*

3. Save and close the `autoexec` macro.
4. Open the `Menu` form using the Design View and click the View Code button in the ribbon to display the VBA Editor.
5. Modify the `Form_Load` and `Form_Timer` methods of the `Menu` code file using the code shown in Listing 18-4. The new code is in bold.

Listing 18-4. *Modified* `Form_Load` *and* `Form_Timer` *Methods*

```
Private Sub Form_Load()
    Me.lblDate.Caption = Format(Now(), "dddd dd, mmm-yyyy")
    Me.lblDate.Caption = Format(Now(), "hh:nn:ss")

    Me.TimerInterval = 1000

    If (Not IsNull([TempVars]![UserName])) Then
        Me.lblUser.Caption = [TempVars]![UserName]
    End If

End Sub

Private Sub Form_Timer()
    Me.lblDate.Caption = Format(Now(), "dddd dd, mmm-yyyy")
    Me.lblTime.Caption = Format(Now(), "hh:nn:ss")

    If (Me.lblUser.Caption = "lblUser" And Not IsNull([TempVars]![UserName])) Then
        Me.lblUser.Caption = [TempVars]![UserName]
    End If
End Sub
```

The `autoexec` macro runs as soon as the application is loaded. However, the `Menu` form is also loaded on start-up, because it is defined as the default form. You cannot guarantee that the macro has set the `TempVar` before the form is loaded. To handle this, you also added code to the `Form_Timer` method to set

www.it-ebooks.info

the label. To prevent updating it every second, the code only updates the label if it still has its default value, `lblUser`. In both cases, to avoid a runtime error, the code makes sure the `TempVar` has been set before trying to use it.

To test this you must close the `Library` database and re-open it. The `autoexec` macro has already run before you added the second action so the `TempVar` has not been set. If you try to run the `autoexec` macro again, it will fail because the ribbon has already been loaded.

Close the application and reload it. Verify that the user name is displayed on the `Menu` form.

Storing the CheckOut User

Now that you have a `TempVar` that stores the current user's name, you'll take advantage of this and store this when creating a new loan. You'll first modify the `Loan` record to add a new field called `UserName`. Then you'll modify the `CheckOut` form to store the user's name in the new field.

1. Open the `Loan` table using the Design View. Add a new column named **UserName** with the `Text` Data Type as shown in Figure 18-4.

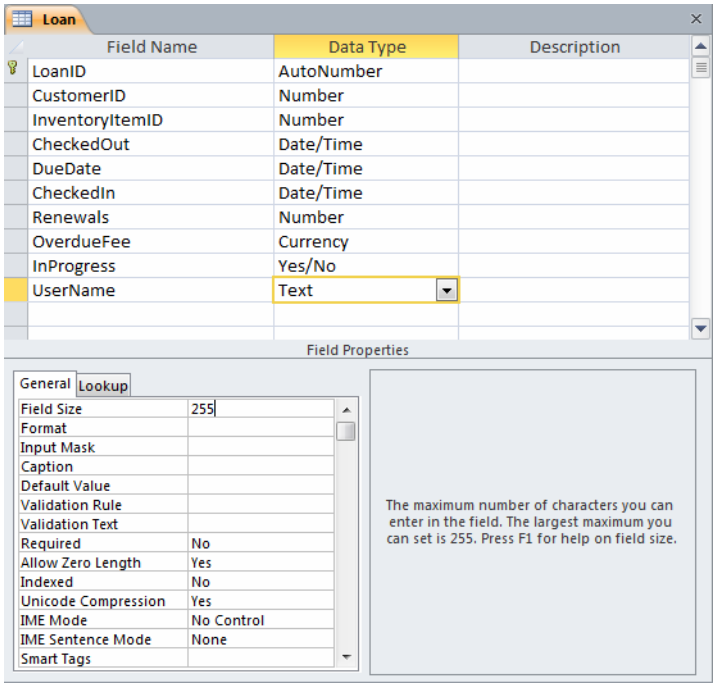


Figure 18-4. *Adding the* `UserName` *field*

2. Save and close the `Loan` form.
3. Ideally, you would simply modify the Before Change data macro to update the `UserName` field on the `Loan` record. However, as I mentioned, this will not work

www.it-ebooks.info

because data macros do not have access to your `TempVars`. Instead, you'll need to update the record and store the `UserName` manually. Fortunately you already have a query that updates the `Loan` record; you'll just need to add the `UserName` field to that query. Open the `CheckOut` form using the Design View. Click the View Code button to display the VBA Editor.

4. In the `Complete_Click` method, change the code that creates the SQL statement using the following code. The original SQL statement only set the `InProgress` column. The modified code also sets the `UserName` field using the `TempVar`:

```
sSQL = "UPDATE Loan SET Loan.InProgress = False, " & _
            " Loan.UserName = '" & [TempVars]![UserName] & "'" & _
       "WHERE Loan.InProgress=True " & _
       "AND Loan.CustomerID=" & txtCustomerID & ";"
```

■ **Note** This code uses the `CurrentDb` object to execute the SQL, which is outside of the application scope. I told you earlier that this will cause an error. In this case, the entire SQL statement is passed in to the `CurrentDb` object so it doesn't need to access the `TempVar`. If you use the `CurrentDb` to execute a query and the query uses a `TempVar` then you'll get an error because the query will not know how to evaluate the `TempVar`.

---

5. Save the code changes.
6. Open the `CheckOut` form using the Form View and check out an item to a customer.
7. Then open the `Loan` table using the Datasheet View. The record you just added should have the `UserName` field populated, as shown in Figure 18-5.

Loan

LoanID	CustomerID	InventoryIte	CheckedOut	DueDate	Checked	How	OverdueFee	InProgress	UserName
27	42	7	4/9/2011 3:40:20 PM	4/16/2011		0	$0.00	☐	
28	42	9	4/9/2011 3:41:26 PM	4/16/2011		0	$0.00	☐	
29	42	10	4/9/2011 3:41:29 PM	4/16/2011		0	$0.00	☐	
30	42	11	4/9/2011 3:41:40 PM	5/30/2011		1	$0.00	☐	
31	37	13	5/6/2011 7:53:25 PM	5/13/2011		0	$0.00	☐	mark
(New)			5/6/2011 7:55:33 PM			0	$0.00	☑	

Figure 18-5. *The* `Loan` *table with the* `UserName` *field populated*

Displaying HTML Content

You may have noticed that the item descriptions for the items that were imported from Apress have HTML tags in them. The standard Text Box control simply displays the tags without formatting them. The Web Browser control can display HTML correctly, but you can only pass a URL to it, not the actual HTML content. I'll show you a simple trick that will allow you to display HTML content that is stored in the database using the Web Browser control.

www.it-ebooks.info

To accomplish this, you'll first get the HTML content from the database and save it to a file on the local machine. Then you'll pass the filename to a Web Browser control, which will load the file and render the content. To make this change you'll modify the `Item` form and hide the Text Box control and add an unbound Web Browser control. You'll then add code to the `OnCurrent` event, to create the file and pass it to the Web Browser control.

■ **Note** When you make this change, you will not be able to modify the `Description` field. For this demonstration, you will simply hide the existing `Description` field. You could, however, display both controls and then use the `Description` field for editing and the unbound control for display purposes. Because these fields take up a lot of space, that's not an ideal solution. As an alternative, you could create a separate form that is used for editing and use this one for display only.

Modifying the Item Form

You will need to keep the bound `Description` field to get the content from the `Item` record. However, you will move it to a smaller cell and set its `Visible` property to No. In its place you'll add an unbound Web Browser control.

1. Open the `Item` form using the Design View.
2. Delete the associated label for the `Description` field. Drag the `Description` field in its place and then set the Visible property to No.
3. In the Design tab of the ribbon, click the Web Browser control and the click in the empty cell where the `Description` control used to be. When the Insert Hyperlink dialog appears, just click the Cancel button.
4. Drag this control to the empty cell. Change its Name property to **DescriptionHTML**. Enlarge the field to be about 2-inch high to give you more room to display the contents.

The design should look like Figure 18-6.

www.it-ebooks.info

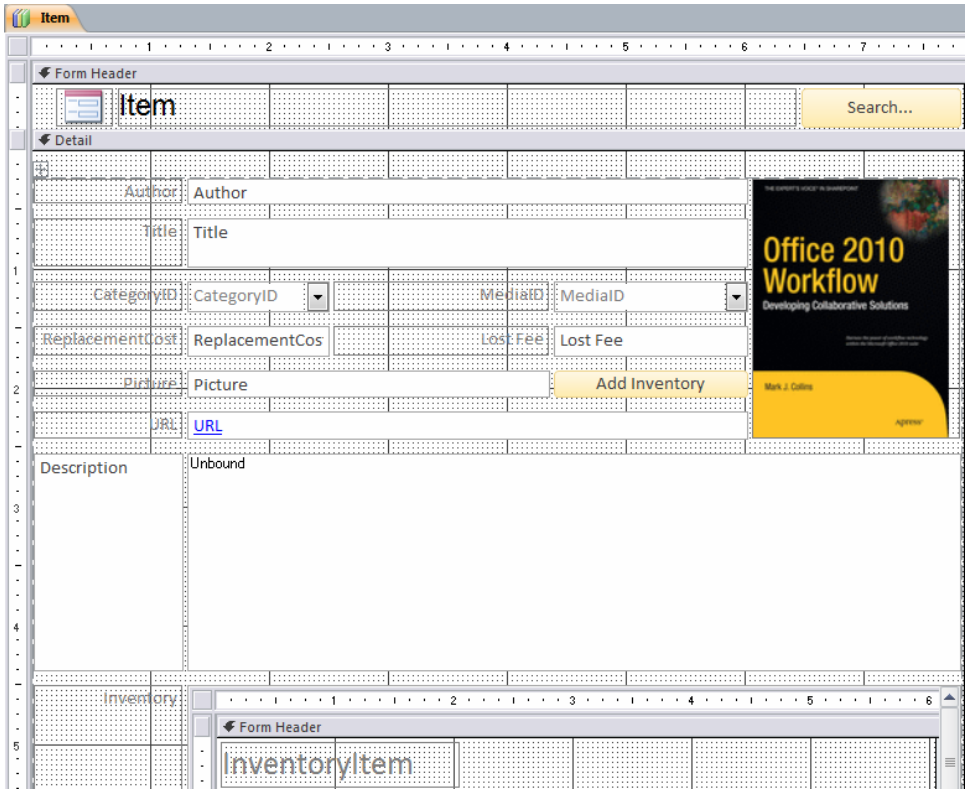


Figure 18-6. *The modified design of the* `Item` *form*

Passing HTML Content to a Web Browser Control

Now you're ready to configure the Web Browser control to display the HTML content that is stored in the `Description` field. You'll add code to the `OnCurrent` event which is raised whenever a new record is being displayed in the form.

1. From the design tab of the ribbon, click the View Code button to display the VBA code.
2. From the Tools menu, select the *References* link. Then add the `Microsoft Scripting Runtime` reference as shown in Figure 18-7.

www.it-ebooks.info

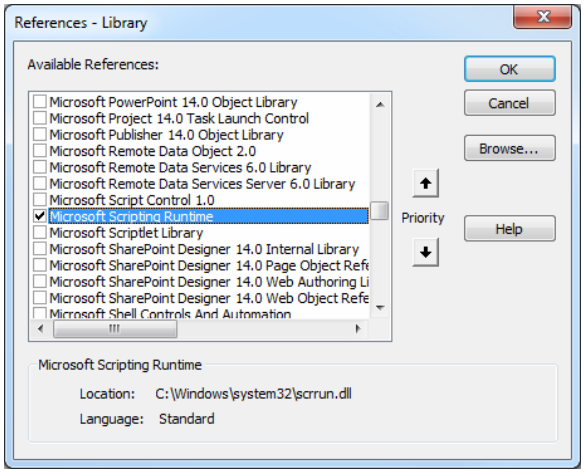


Figure 18-7. *Adding the Microsoft Scripting Runtime reference*

3. Find the existing `Form_Current` method from the `Item` code file.
4. Modify the implementation of this event handler using the code shown in Listing 18-5 (the new code is shown in bold).

Listing 18-5. *The Modified Implementation of the* `Form_Current` *Event Handler*

```
Private Sub Form_Current()

Dim fsys As New FileSystemObject
Dim outstream As TextStream
Dim filename As String

    If (Len(Me.Description) > 1) Then
        filename = GetFullImagePath("Library.htm")
        Set outstream = fsys.CreateTextFile(filename, True, False)
        outstream.WriteLine (Me.Description)
        Set outstream = Nothing
        DescriptionHTML.Object.Navigate (filename)
        DescriptionHTML.Visible = True
    Else
        DescriptionHTML.Visible = False
    End If

    DisplayImage

    If (Len(URL) > 0) Then
        webBrowser.Object.Navigate URL
        webBrowser.Visible = True
    Else
        webBrowser.Visible = False
```


www.it-ebooks.info

```
    End If

End Sub
```

The first part of this code checks to see if there is any content in the `Description` field. If there is, the `GetFullImagePath` function that you created in Chapter 9 is used to return the path where the product images are stored. (Refer to Chapter 9 for more information.) The file used to generate the HTML content will be the `Library.htm` file in this folder.

This new code then writes the `Description` text to this file. Finally, the `Navigate` method of the Web Browser object, `DescriptionHTML`, is called to tell it to load this file.

To test this, switch to the Form View. Click the Search button and select an item from the `ItemSearch` dialog box. The form should look like Figure 18-8.

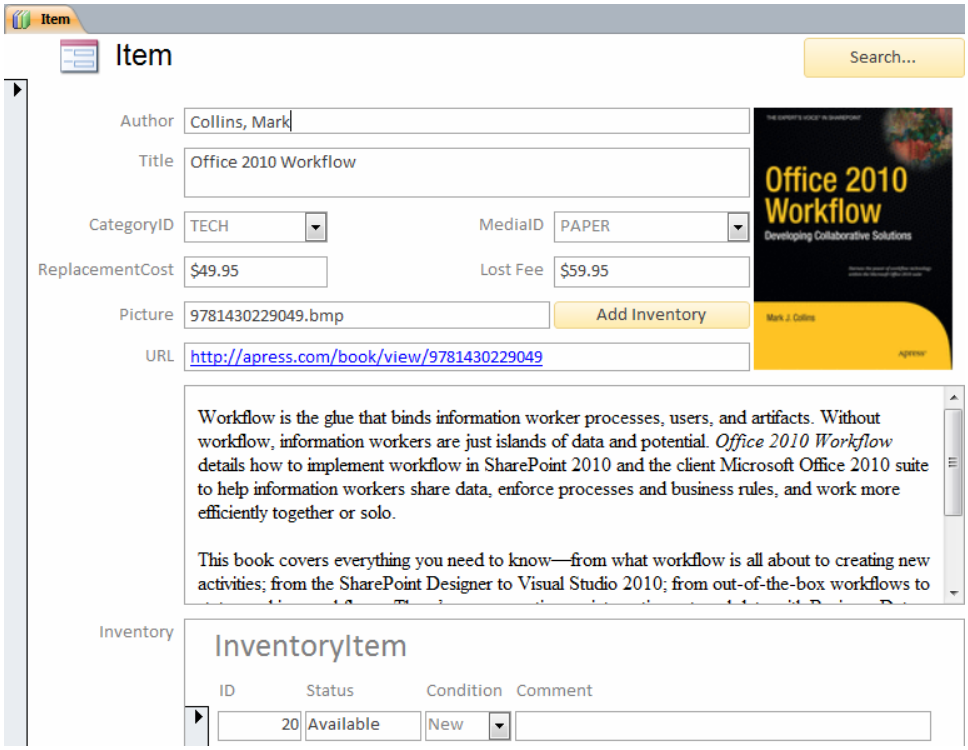


Figure 18-8. *The final* `Item` *form*

Notice the `Description` text is much more readable.

www.it-ebooks.info

Adding a Chart

Access 2010 provides a Chart control that you can drop onto a form or report. While somewhat simplistic compared to the charting available in Excel, it's still a useful feature to be able to add a quick chart. For this exercise, you'll create a blank form and add the Chart control to display a pie chart that illustrates the media types that are most popular.

1. From the Create tab of the ribbon, click the Blank Form button. This will create a blank form and open it using the Layout View. Switch to the Design View.

■ **Note** The Chart control is not available while using the Layout View. When you switch to the Design View, you'll be able to add this control.

---

2. From the Design tab of the ribbon, select the Chart control and then draw a rectangle on the form, taking up the entire area of the form. This will start the Chart Wizard that will take you through a series of dialogs to help you configure your chart.
3. The first dialog box, shown in Figure 18-9, is used to specify the source table or query. Select the Queries radio button and then select the `AllLoans` query.

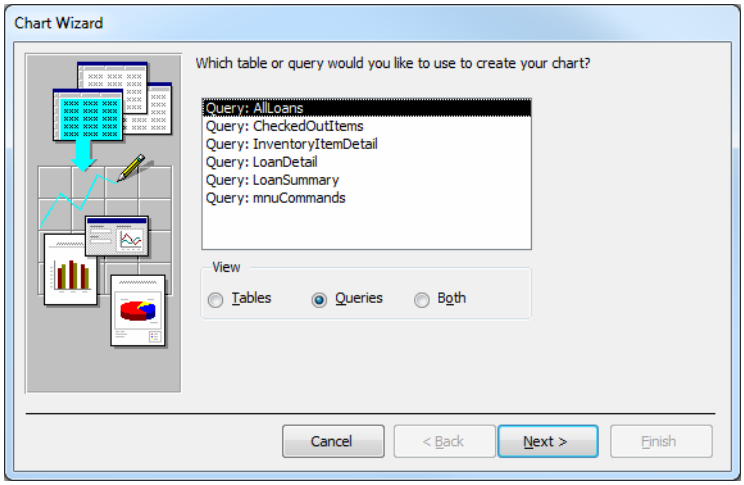


Figure 18-9. Selecting the source table or query.

4. This will be a pie chart that shows the relative portion of all items that have been loaned out from each media type. So the only detail you'll need from the `AllLoans` query is the `MediaDescription` field. In the second dialog box, add the `MediaDescription` field as shown in Figure 18-10.

www.it-ebooks.info

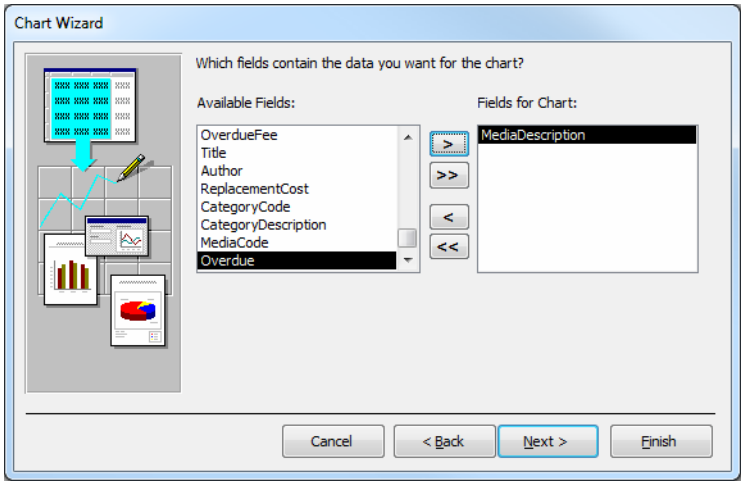


Figure 18-10. *Selecting the columns to be included*

5. In the third dialog box, select the Pie Chart as shown in Figure 18-11.

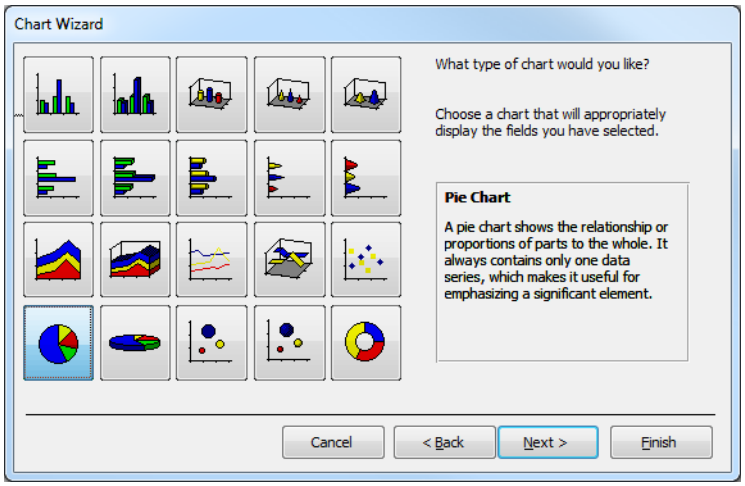


Figure 18-11. *Selecingt the Pie Chart template.*

6. In the fourth dialog box, shown in Figure 18-12, you can configure how you want each of the fields displayed. In this case, because there is only one data column, there's nothing to change. Click the Next button to select the default layout.

www.it-ebooks.info

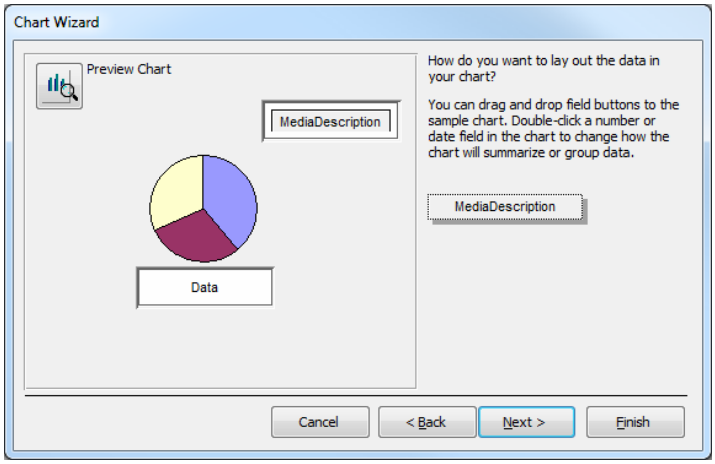


Figure 18-12. *Configuring the data fields*

■ **Note** The chart that is displayed on this dialog box uses canned data. It uses your field names like `MediaDescription` so you might be tempted to think it is also displaying a chart based on your data. However, the Preview Chart button can be used to see how the selected options will look with your actual data.

---

7. In the final dialog box, enter the chart title as **Media Types** and select the legend option as shown in Figure 8-13.

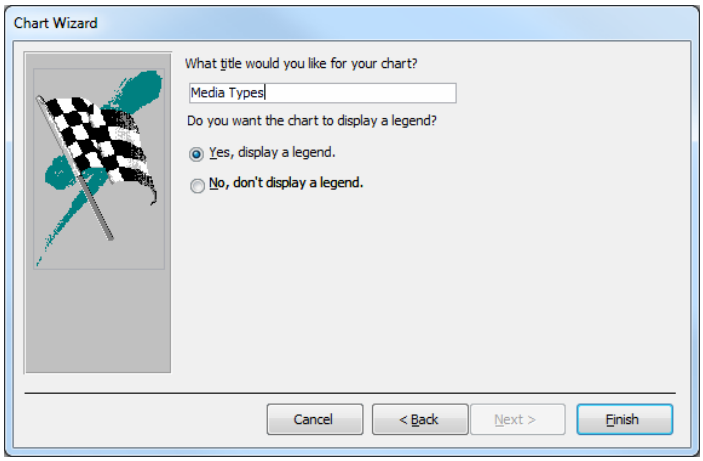


Figure 18-13. *Specifying the chart title*

www.it-ebooks.info

8. When the wizard has completed, the modified form will be displayed using the Design View. In Design View, the chart is also displayed with canned data. Right-click the chart and click the *Chart Options* *Edit* links, which will open the Chart Editor.

9. From the Chart menu, click the *Chart Options* link, which will display the Chart Options dialog box. Select the third tab, called Data Labels, and then select the Percentage check box as shown in Figure 18-14.

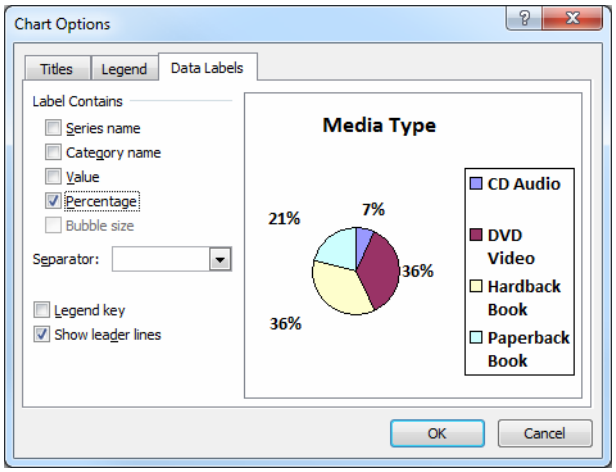


Figure 18-14*. Including the percentages in the chart*

10. From the File menu, select the Save option. This will save the changes to the chart and close the Chart Editor. Enter the name as **MediaTypesChart** when prompted.

11. Switch to the Form View and the chart should look like Figure 18-15.

www.it-ebooks.info

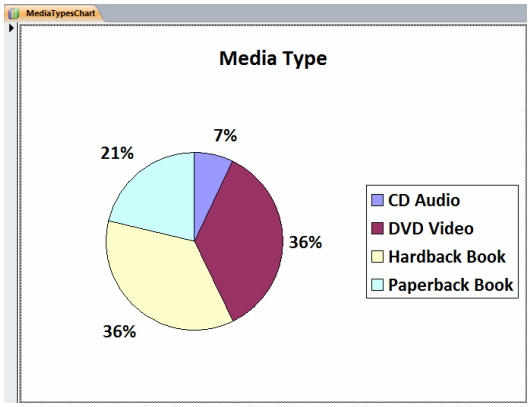


Figure 18-15. *The final* `Media Types` *chart*

Summary

In this chapter, I demonstrated some simple techniques you can use to enhance your Access applications. Keep these in mind as you design and implement your future projects.

- Use timers to "wake-up" a form so you can refresh the data automatically or provide animation by manipulating the control properties.
- Tap in to the enormous potential by calling the Windows API methods.
- Use `TempVars` to maintain variables that are needed by multiple forms or reports.
- Use a Web Browser control to display HTML content.
- Add a chart to forms and reports to provide a graphical presentation of the data.

In the next chapter I'll explain some of available techniques to deal with security issues when using an Access database.

www.it-ebooks.info

CHAPTER 19

Security

As developers, when we talk about security, we tend to think in terms of preventing unauthorized access to data and applications. As a consumer, however, the concern is more often about installing something that will infect our computer, install malware, or perform other such damaging actions. Both threats are real and need to be considered when providing an Access solution. Which steps you take to address them will depend heavily on your environment, the intended users of your database, and how your software will be distributed.

In this chapter, I will explain the facilities that are available to you along with a discussion of when these are normally used. Each security feature is provided to mitigate one or more potential threats. As you assess the threats in your environment you can make logical decisions about the safeguards that are appropriate.

Using Trusted Documents

As you have learned over the last 18 chapters, an Access database is not just a database. It is a powerful application platform. In the last chapter, for example, I showed you how to call the Windows API, which includes methods for creating and deleting files and folders, just to name a few. This allows you to do some amazing things. Unfortunately, unscrupulous people have exploited these capabilities to do much harm. In fact, an Access database has become a favorite place to embed viruses and other malware.

To allow the use of these features for legitimate purposes while preventing unwanted misuse, Access distinguishes between trusted and non-trusted documents. Trusted documents can run unrestricted while non-trusted documents have much of this *active* content disabled. I will first explain how Access handles a non-trusted document and the ways that you can configure this behavior. Then I'll show you how to control which documents should be trusted.

Understanding Disabled Content

Access allows you to open a non-trusted document; however the following features are blocked and will not run:

- VBA code
- "Unsafe" macro actions, which modify the database or use resources outside the database (such as the file system)
- Action queries, which insert, update, or delete records in the database

www.it-ebooks.info

- ActiveX controls
- Queries that modify the data schema

This allows you to open a database and view its content without exposing your computer to potentially harmful code. For example, if you try to open the `Library.accdb` file when it is not trusted, the first thing you'll see is the dialog box shown in Figure 19-1.

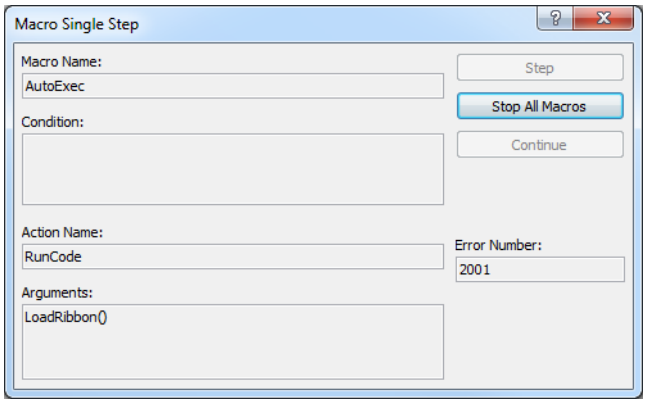


Figure 19-1. *The* `autoexec` *macro is blocked*

The `autoexec` macro calls a VBA function, `LoadRibbon` in this case. Because VBA code is disabled, the macro fails with the 2001 error. After you click the Stop All Macros button, you'll then see the reason for this error. The message bar at the top of the application, shown in Figure 19-2, explains that some content has been disabled.



Figure 19-2. *The untrusted document message bar*

You can configure what Access will do with a non-trusted document. This is configured through the Trust Center settings. From the Backstage view, click the Options button, which will display the Access Options dialog box. Then select the Trust Center tab, which is shown in Figure 19-3.

www.it-ebooks.info

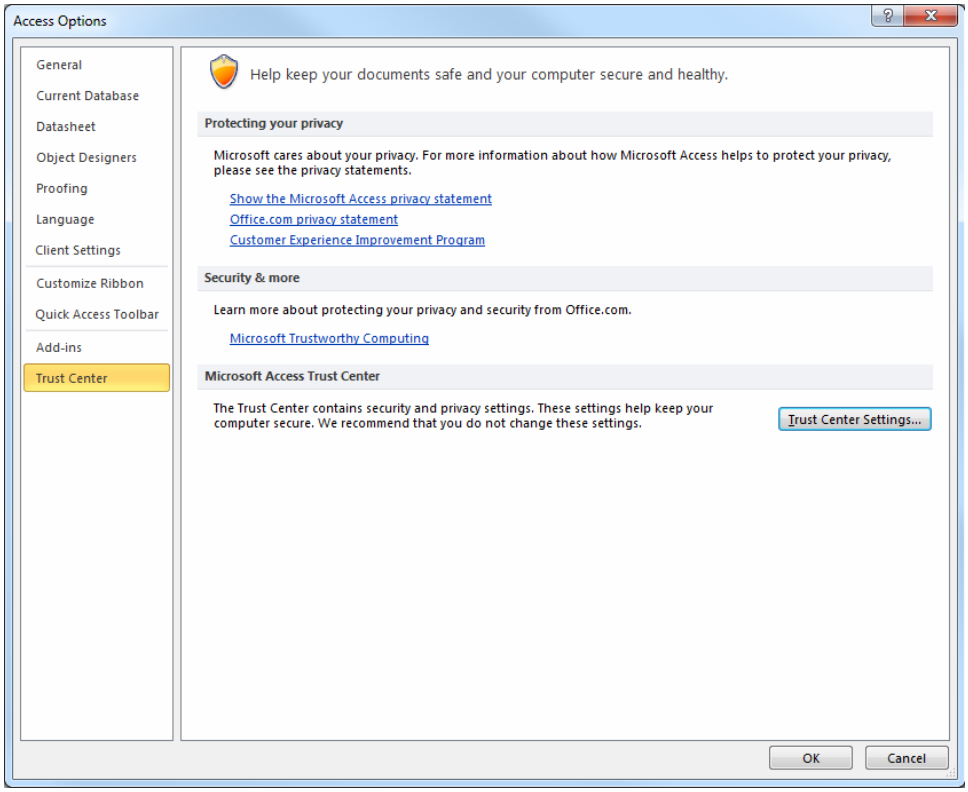


Figure 19-3. *The Trust Center tab of the Access Options dialog box*

Click the Trust Center Settings button to display the Trust Center dialog box. There are several tabs in this dialog box, which are shown in Figure 19-4.

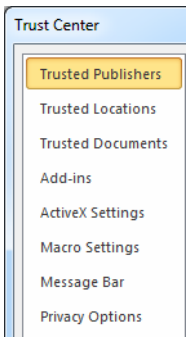


Figure 19-4. *The tabs of the Trust Center dialog box*

www.it-ebooks.info

The first three tabs provide options for determining which documents should be trusted, and I'll explain these later in the chapter. The next three are used to configure how Access handles a non-trusted document.

■ **Caution** All of the Trust Center settings apply to all Office products. If you make any changes, they will also apply to Word, Excel, and any other Office applications that are used on this machine.

For more information about the configuration options on these tabs, check out the following articles:

- Add-ins: `http://office.microsoft.com/en-us/access-help/view-manage-and-install-add-ins-in-office-programs-HA010354315.aspx?CTT=1`
- ActiveX settings: `http://office.microsoft.com/en-us/access-help/enable-or-disable-activex-settings-in-office-files-HA010354314.aspx?CTT=1`
- Macro settings: `http://office.microsoft.com/en-us/infopath-help/enable-or-disable-macros-in-office-files-HA010354316.aspx?CTT=1`

■ **Tip** In my opinion, I would not spend too much time trying to figure out exactly what content is blocked for non-trusted documents. The simplest and probably most effective approach is to allow all active content on trusted documents and block all unsafe content on non-trusted documents. The default settings do that. The main area that you'll need to focus on is determining which documents should be trusted. To say this another way, don't modify these settings to allow a non-trusted document to work. Instead, if it's from a reliable source make it a trusted document. By modifying these settings you also open up these options to all other non-trusted documents.

In the Message Bar tab, shown in Figure 19-5, you can turn off the Message Bar that warns about disabled content. This does not enable the content, but simply suppresses the message that tells you the content is disabled.

www.it-ebooks.info

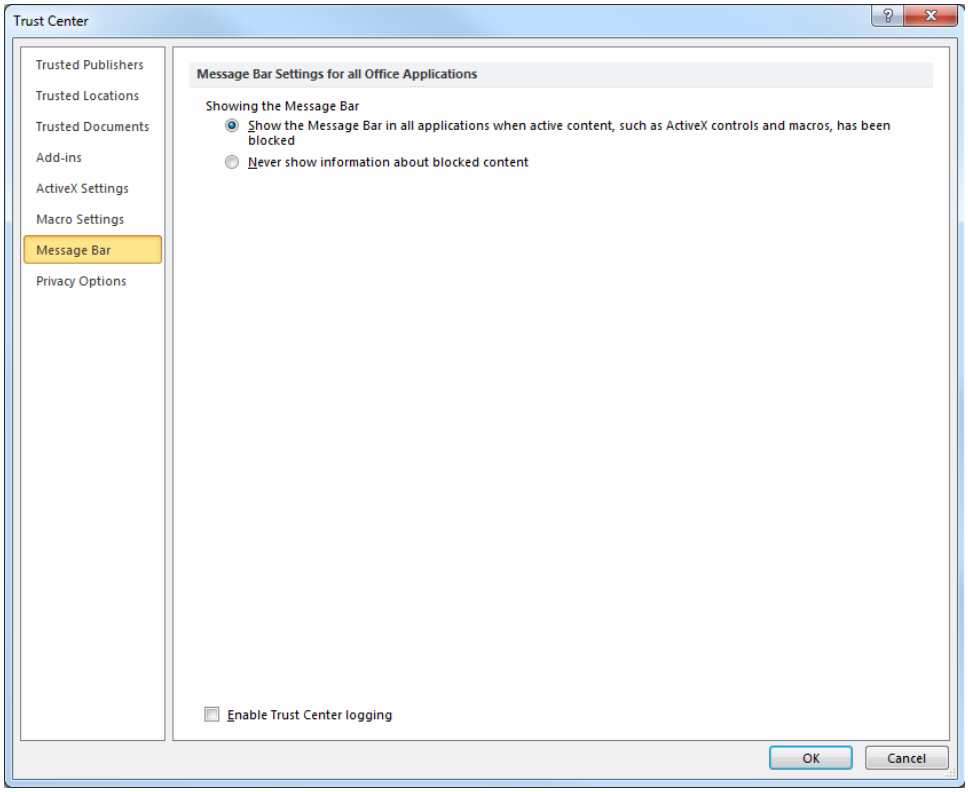


Figure 19-5. *The Message Bar tab*

Trusting a Document

Because trusted documents are not restricted in any way, deciding to trust a document becomes paramount. Deciding what to trust is a two-part question. The first part is rather obvious: does the document come from a reliable source where you're confident that no malicious content has been added?

The second part is subtler: is the document in a secure location that is unlikely to be tampered with? You may have an Access database that was developed internally and that you're quite certain has no malicious code. To allow users easy access to it, it is placed on a network file system. If this file system is not protected, someone could later modify the database and add malicious code.

Trusting a single document is easy. When the Message Bar displays the warning about content being disabled (shown in Figure 19-2), just click the Enable Content button. That will make the current document trusted. You can also trust a document from the Backstage view as shown in Figure 19-6.

www.it-ebooks.info



Figure 19-6. *Trusting a document from the Backstage view*

If you choose to trust a document that is on a network drive, you will see the warning shown in Figure 19-7.

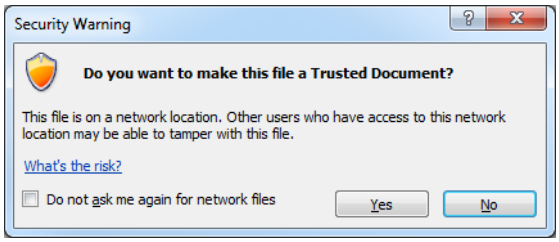


Figure 19-7. *Warning about trusting a document on a network drive*

Using Trusted Locations

There are some Trust Center settings that control both the ability and the need to trust an individual document. Access allows you to designate locations that are considered trusted. You can copy or move a non-trusted document to a trusted location and the file is now trusted. If you have a lot of databases, moving them to a trusted location is an efficient way of making them all trusted.

The Trusted Locations tab of the Trust Center dialog box, shown in Figure 19-8, allows you to add one or more trusted locations.

www.it-ebooks.info

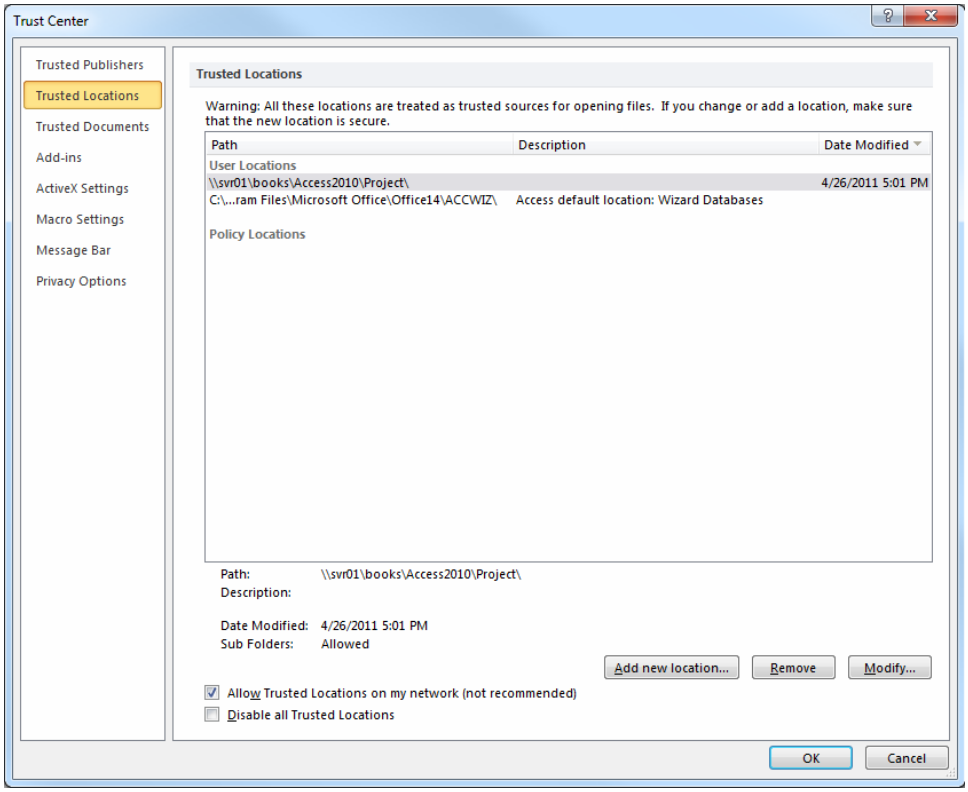


Figure 19-8. *The Trusted Location tab*

To add a trusted location, click the "Add new location" button, which will display the Microsoft Office Trusted Location dialog box, shown in Figure 19-9. Enter or browse to the path that should be added. You can also use the Remove and Modify buttons to manage the existing trusted locations.

www.it-ebooks.info

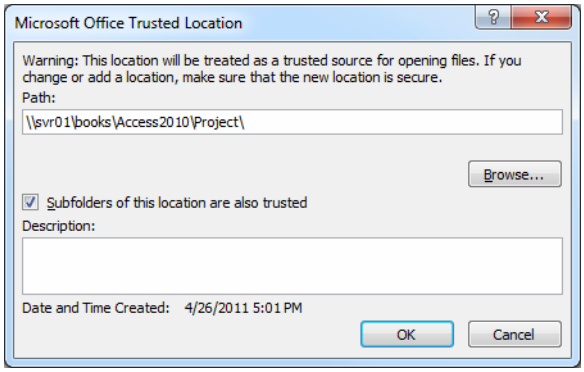


Figure 19-9. *Adding a new trusted location*

There are two options at the bottom of the Trusted Location tab that I need to explain. If the "Allow Trusted Locations on my network (not recommended)" check box is unselected, you will not be able to add a trusted location that is on a network drive; only local drives are allowed. Furthermore, if there is an existing trusted network location is will be disabled.

If the "Disable all Trusted Locations" check box is selected, you will not be able to add a trusted location and all existing trusted locations will be disabled.

■ **Caution** Do not add any default folders such as My Documents or My Downloads as a trusted location. If these default locations are trusted, you will essentially default all new files to be trusted, including files downloaded from the web. Instead, create a trusted subfolder where you can move the file to once you have decided to trust it.

Configuring Trusted Documents

The Trusted Documents tab, shown in Figure 19-10, has a few options that affect your ability to trust documents.

www.it-ebooks.info

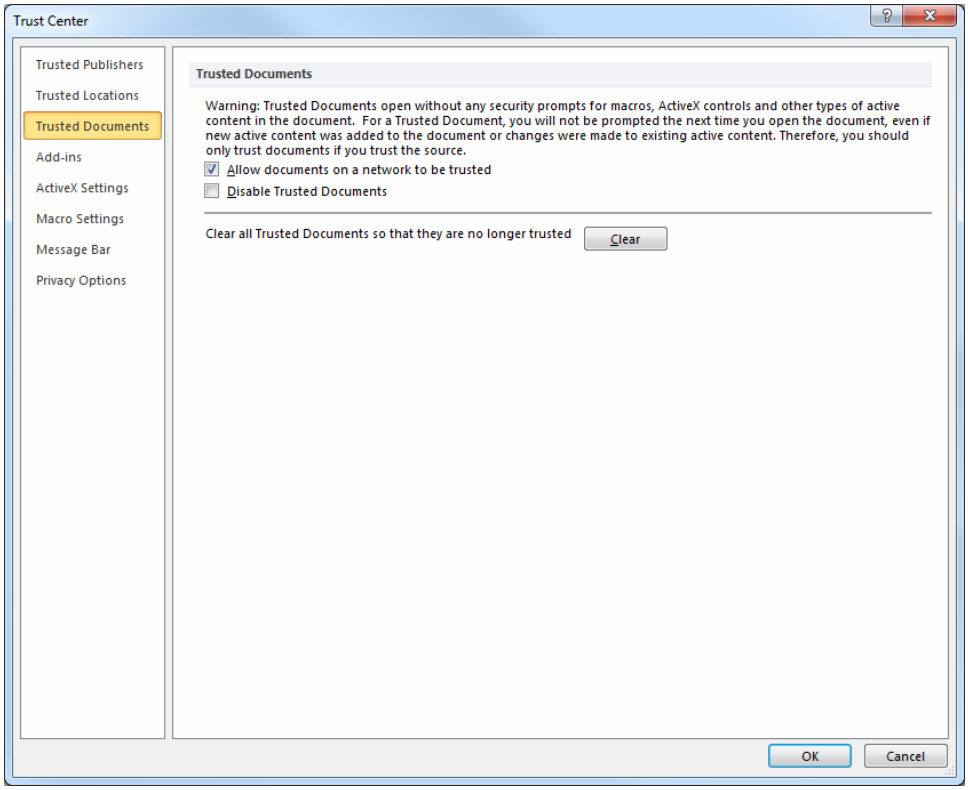


Figure 19-10. *The Trusted Documents tab*

If the "Allow documents on a network to be trusted" is not selected and you open a non-trusted document that is on a network location, you will see the warning about disabled content. You can click Enable Content, which will enable the content for this session only. The file will not be marked as trusted so the next time you open it, the security warning will appear again.

Likewise, selecting the Disable Trusted Document check box works the same way except it applies to all folders, even local files. This option will prevent you from making any file trusted. You can still enable the content for the current session only.

The Clear button is used to make all documents that you may have trusted previously to become untrusted. This does not apply to documents in a trusted location.

■ **Note** You should keep in mind that the Trusted Locations settings have precedence over the settings in the Trusted Document tab. For example, if you have a trusted network location, all files in that folder are trusted. Unselecting the "Allow documents on a network to be trusted" or selecting the Disable Trusted Document check boxes will have no effect on documents in a trusted location.

---

www.it-ebooks.info

Signing a Database

As I said, determining which databases should be trusted is a key consideration in safeguarding your system. One of the primary factors in this determination is where the database came from. Obviously those that are developed internally are likely to be trusted. If you download a database from the Web, you are probably more likely to trust one from Microsoft than you are one from Hackers Anonymous.

But how do you know that the file really came from Microsoft. And, if it did, how do you know that it has not been modified in some way. The answer to both questions is *digital signatures*. Access 2010 allows you to package your database with a digital signature. This is used to both verify the identity of the person who created the database as well as to ensure it has not been modified since it was originally packaged.

Signing an Access database is analogous to having a document notarized. You sign a document in front of a Notary Public who also checks one or more forms of identification such as a driver's license. The notary then puts their seal on the document, confirming that it was really you who signed the document.

Digital signatures, however, take this a step further and also ensure the document has not been tampered with. In earlier times, an official document would be rolled up and sealed with wax. Before the wax hardened, a signet ring would be used to impress an image into the wax. No one could open the document without breaking the seal and you could not re-seal it without the signet ring. Along these same lines, a digital signature can be used to determine if the database was altered in any way since it was originally signed.

Creating a Certificate

Before you can sign a document, you must have a certificate that has been issued by a third party certificate authority (CA). When someone applies for a certificate, the CA must do diligence to verify that the applicant is who they claim to be. Once a certificate has been issued, all other parties can now trust who the originator is. For example, when you download a software update that is signed by the Microsoft Corporation, you can have confidence that this really did come from Microsoft because the CA that issued the certificate has taken the proper steps to confirm this.

If you need to obtain a certificate, there are quite a few CAs to choose from. Here is a link that provides some reviews and comparisons: `www.sslshopper.com/certificate-authority-reviews.html`. You can use this resource to help choose a CA.

Microsoft Office also provides a tool that allows you to generate your own certificate, which will allow you to package and sign your database without having to buy a certificate. This is referred to as a *self-signed* certificate. A self-signed certificate cannot be used to verify the originator's identity but it can be used to generate a digital signature. These signatures are useful for sharing documents internally or between trusted organizations. They still provide protection from someone tampering with your document.

You will use this facility to generate a self-signed certificate and you will then use it to package your Access database.

1. From the Start menu, select the *All Programs* ➤ *Microsoft Office* ➤ *Microsoft Office 2010 Tools* ➤ *Digital Certificate for VBA Projects* links.
2. The Create Digital Certificate dialog box will appear. Enter the name **Library** as shown in Figure 19-11.
3. Click OK to generate the certificate.

www.it-ebooks.info

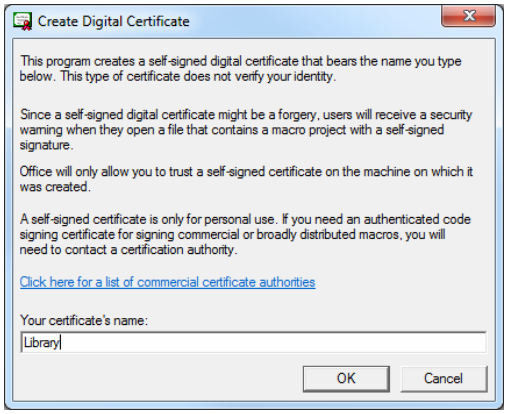


Figure 19-11. *Creating a new certificate*

When the certificate has been generated, you will see the confirmation shown in Figure 19-12.

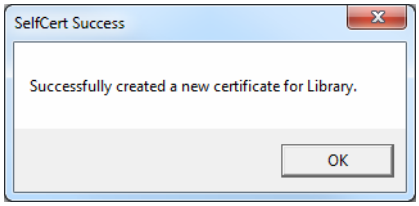


Figure 19-12. *Certificate successfully generated message*

Package an Access Database

Now that you have a certificate you can use it to package your database and include a digital signature. Packaging is a simple process of copying the database into a compressed file and including the certificate and electronic signature.

1. Open the `Library.accdb` file that you used in Chapter 18 and go to the Backstage view.
2. Select the Save & Publish tab. Then select the Save Database As tab and select the Package and Sign option, as shown in Figure 19-13. Then click the Save As button.

www.it-ebooks.info

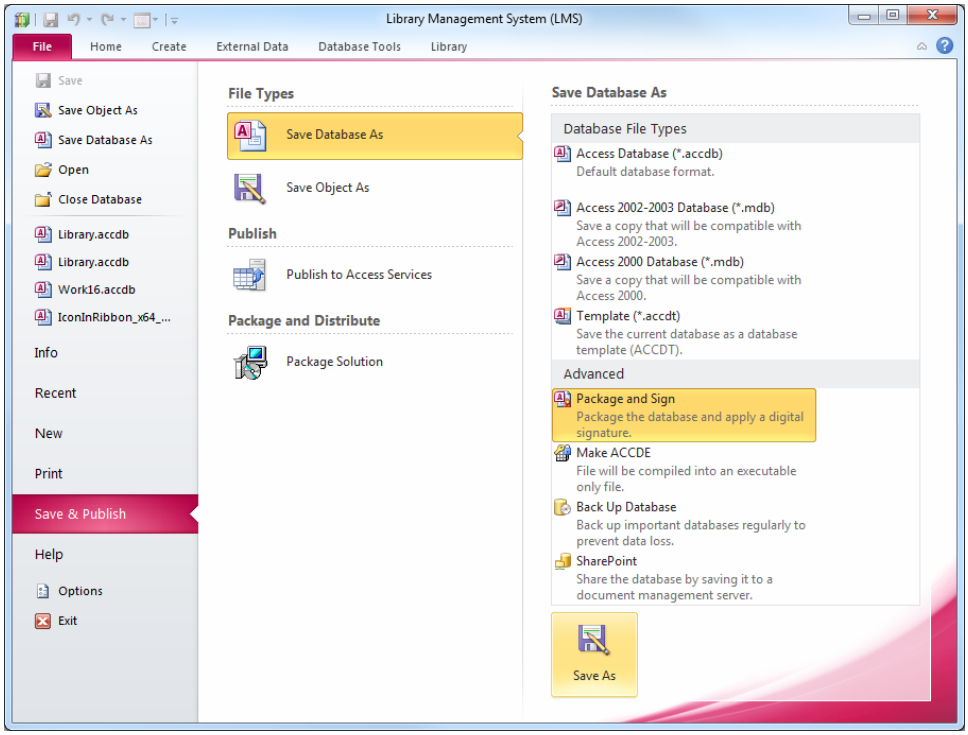


Figure 19-13. *Selecting to package and sign the current database*

3. The Windows Security dialog will open and ask you to select a certificate to include with the package. Select the `Library` certificate that you just created, as shown in Figure 19-14.

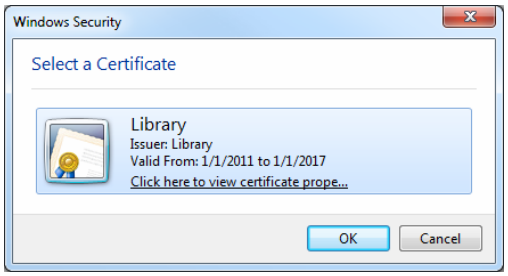


Figure 19-14. *Selecting the certificate to use*

After you click the OK button, you will be prompted for a location to store the packaged version of the `Library` database, which will have an `.accdc` extension.

www.it-ebooks.info

Installing a Signed Database

Your database is now saved as a signed document named `Library.accdc`. This is the file that you can put on the web to be downloaded or distributed to your end-users in some other fashion.

1. Open the `Library.accdc` file that you just created. When you open it, the dialog shown in Figure 19-15 will be displayed warning you that the publisher is not trusted.

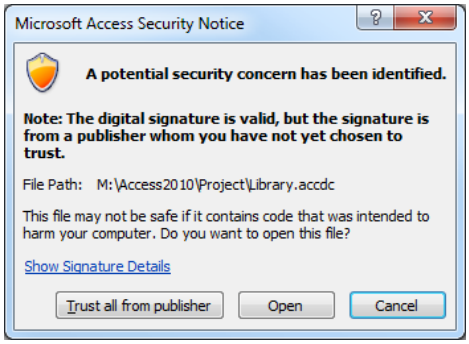


Figure 19-15. *Warning about an untrusted publisher*

■ **Note** If you purchase a certificate from a CA and use that to sign your database, the end user may still see this warning. A certificate merely verifies the publisher. The user still has to decide to trust the publisher. If this is a trusted publisher they can click on the "Trust all from publisher" button. In addition to opening this document, any future files from that publisher will also be opened without displaying this warning.

■ **Tip** If you click the "Trust all from publisher" button, this publisher will be added to the Trust Center. The first tab of Trust Center dialog box lists all of the trusted publishers as shown in Figure 19-16.

---

www.it-ebooks.info

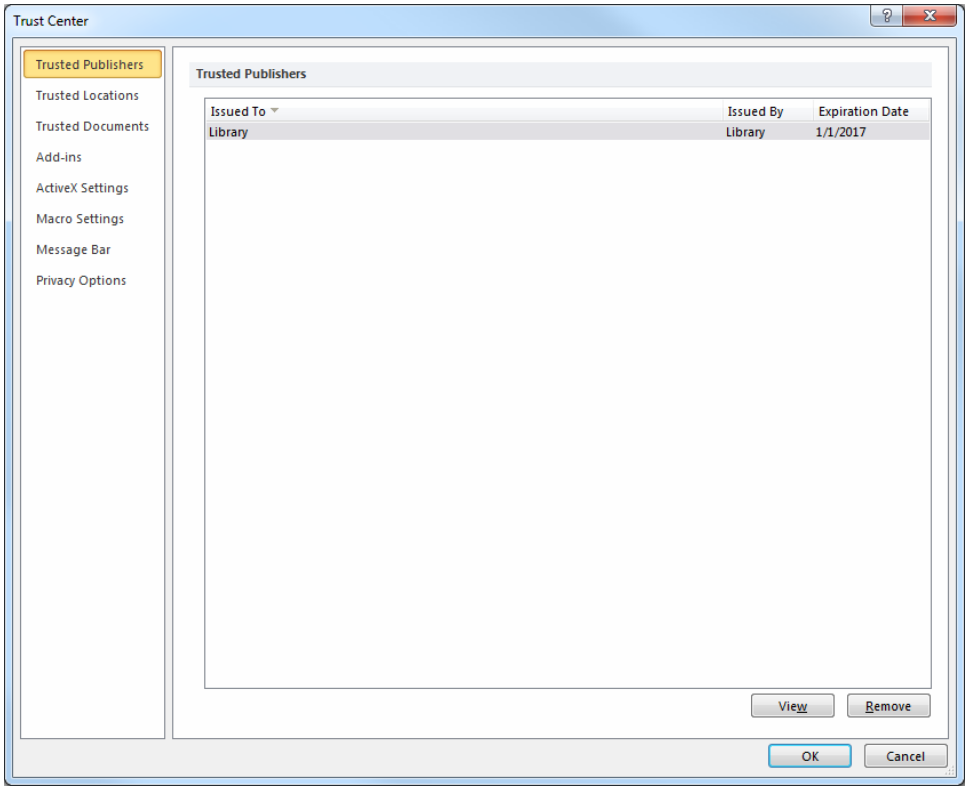


Figure 19-16. *Viewing the trusted publishers*

2. Click the *Show Signature Detail* link, which will display the dialog shown in Figure 19-17.

www.it-ebooks.info

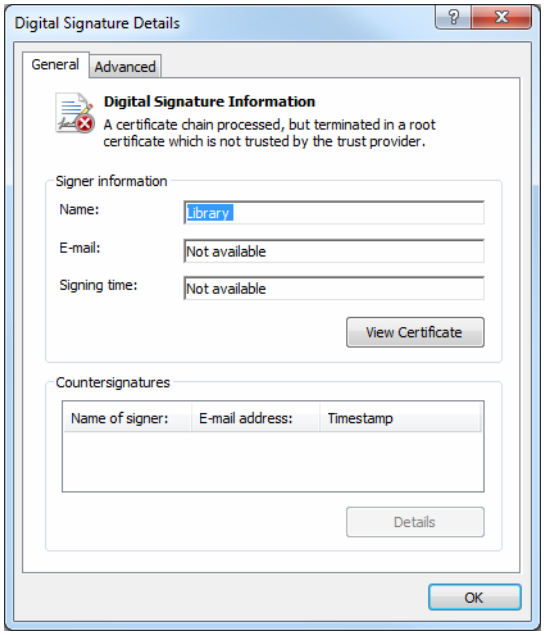


Figure 19-17*. The Digital Signature Details dialog box.*

3. Because you used a self-signed certificate there are no details about the signer (publisher). Click the View Certificate button, which will display the Certificate dialog box shown in Figure 19-18.

www.it-ebooks.info

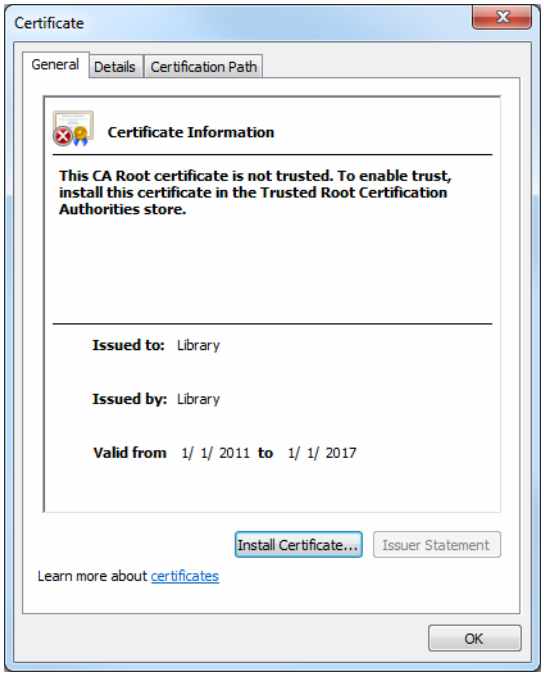


Figure 19-18. *Displaying the certificate details*

4. The last two dialog boxes are intended to provide details about the publisher and certificate so you can decide if the document should be trusted or not. Click the OK button on both of these dialog boxes to close them.

5. Click the Open button on the Microsoft Access Security Notice dialog box. This will display the Extract Database To dialog box shown in Figure 19-19.

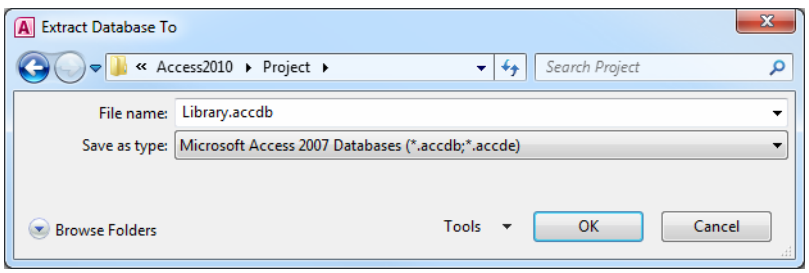


Figure 19-19. *Extractng the* `Library.accdb` *file*

www.it-ebooks.info

■ **Tip** The packaged file (`.accdc` extension) is a compressed file containing the original database (`.accdb` extension) as well as the certificate and electronic signature. When you open it, Access displays the publisher and certificate details. When you choose to open the database, you will actually be extracting the original file (`.accdb` extension). The Extract Database To dialog box is used to specify the location for this database.

6. Select a location to save the `Library.accdb` file and click the OK button.

Configuring Sandbox Mode

Even after you have trusted a database (or placed it in a trusted location) there are certain expressions that are considered unsafe, which are blocked. This is referred to as *sandbox mode*. This feature is designed to protect your system from expressions that can cause harm. These unsafe expressions are generally ones that access the Windows shell.

By default, the sandbox mode is enabled and unless you have a good reason to do otherwise, you should leave it that way. If you have a trusted document that is getting errors that refer to sandbox mode, you can try disabling the sandbox. While this does potentially open up your computer to a misbehaving database, the risk is actually fairly low, because this only applies to trusted documents. Non-trusted documents always run in sandbox mode regardless of how you have configured your system.

To disable the sandbox mode, you'll need to make a registry change. Run the `regedit.exe` program and navigate to HKLM ▯ Software ▯ Microsoft ▯ Office ▯ 14.0 ▯ Access Connectivity Engine ▯ Engines as shown in Figure 19-20.

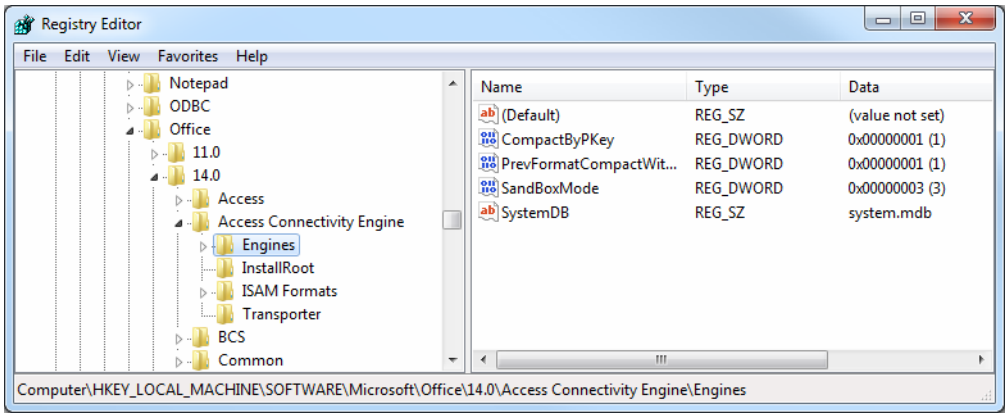


Figure 19-20. *Viewing the sandbox configuration*

Then select the `SandBoxMode` value and enter one of the following values:

- 3: Sandbox mode is enabled always (this is the default value)
- 2: Sandbox is used for non-Access (ODBC) database but not for Access databases

www.it-ebooks.info

- 1: Sandbox is used for Access databases but not for non-Access databases
- 0: Sandbox is always disabled

Encrypting an Access Database

Switching gears a little, I want to show you how to encrypt your database to prevent unauthorized access. If your database has sensitive information that you need to restrict access to, encrypting is the perfect solution. To encrypt a database you must first open it in exclusive mode.

1. Close the `Library.accdb` file if you still have it open.
2. Start Access and click the Open button in the Backstage view.
3. In the Open dialog box browse to the `Library.accdb` file. Then click the dropdown icon next to the Open button and select the *Open Exclusive* link, as shown in Figure 19-21.

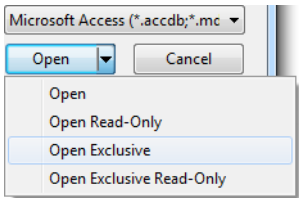


***Figure 19-21**. Opening the `Library.accdb` file in exclusive mode*

4. Then, from the Info tab of the Backstage view, click the Encrypt with Password button as shown in Figure 19-22.



***Figure 19-22**. Selecting the Encrypt with Password option*

5. You will be prompted for a password. Enter it twice as a confirmation, as shown in Figure 19-23.

www.it-ebooks.info

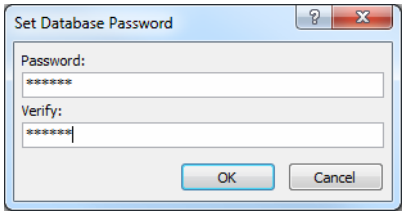


Figure 19-23. Entering a password

■ **Caution** Make sure you save this password so you won't lose it. There is no way to recover the database without the password.

---

6. After you enter a password you'll probably see the dialog box shown in Figure 19-24. Click the OK button to continue.

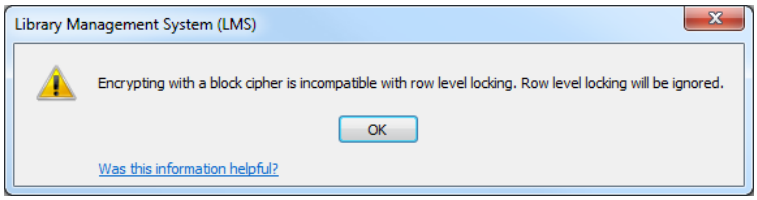


Figure 19-24. Switch to page-level locking

■ **Note** By default, Access uses row-level locking. When you edit a record, that record is locked to prevent other users from modifying the same record. You can also use page-level locking. In Access 2010 a page is 4K bytes, which may hold several records. With page-level locking, the entire page is locked when a record is edited, which may block other records that are not being edited. Because the database is decrypted a page at a time, Access must lock the entire page. Therefore, with encrypted databases, Access uses page-level locking. This dialog is just letting you know that this change is being made.

---

7. Close the database and re-open it. You should see a dialog prompting you for a password as shown in Figure 19-25. Enter the password to open the database.

www.it-ebooks.info

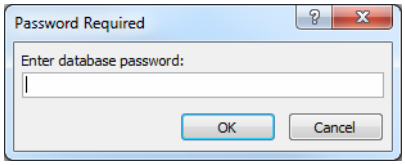


Figure 19-25. *Entering the database password*

If you later decide that you don't want to encrypt the database or password protect it, just click the Decrypt button in the Backstage view, as shown in Figure 19-26. Remember, you must also open the database in exclusive mode to decrypt it. You will also need to re-enter the previous password as a confirmation.

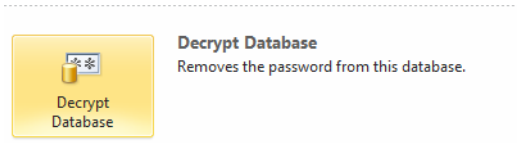


Figure 19-26. *Decrypting the database*

Summary

In this chapter, I explained some of the security features provided by Access 2010. A key component of this security is the use of trusted documents. Access allows you to open and view untrusted databases while restricting any feature that could harm your computer. At the same time, trusted documents are allowed to function without restrictions.

A significant factor in deciding to trust a database is having confidence in the originator. I showed you how to package your Access database with a certificate and an electronic signature. The certificate is used to validate the identity of the originator. This will require obtaining a certificate from a certificate authority (CA). The electronic signature is used to ensure that the database has not been tampered with since the package was initially signed.

To prevent access by unauthorized persons, you can encrypt your database, which will require a password to access it. You can then safely send an Access database that has sensitive information knowing the data is protected.

www.it-ebooks.info

APPENDIX A

Northwind Web Database

In this book, I explained the core concepts needed for developing a database solution using Access 2010. As I stated at the beginning, this was not intended to provide an exhaustive coverage of every feature or every application of each feature. Rather, my goal was to give you a broad but solid foundation, organized for easy assimilation. Equipped with this understanding, one of the best ways to build on that foundation is to explore the various sample templates that are available.

Access has traditionally included a sample Northwind database that demonstrates the many of the Access features. This is used by a fictitious company called Northwind Traders, which sell specialty dry goods. With the 2010 release, Microsoft provided a web version of the Northwind database. If you're interested in building SharePoint-hosted web databases, I highly recommend downloading this sample and exploring how it was implemented. I will show you how to install it and get you started.

■ **Tip** There is a large and growing list of sample databases that you can download. Not only are these great for learning, but they are fully functional applications. You may be able to use one of these as a starting point for your next project.

Installing the Northwind Web Template

You will start by downloading the Northwind web database and then publishing it to a SharePoint server. You will publish this to the Web just like you published your `Library` database.

Downloading the Template

The `Office.com` site has many available templates that you can load from the Access application.

1. Start Access 2010, which will display the File New page that you can use to select an existing database or create a new one.
2. In the Office.com Templates section, type **northwind** and click the search button as shown in Figure A-1.

www.it-ebooks.info

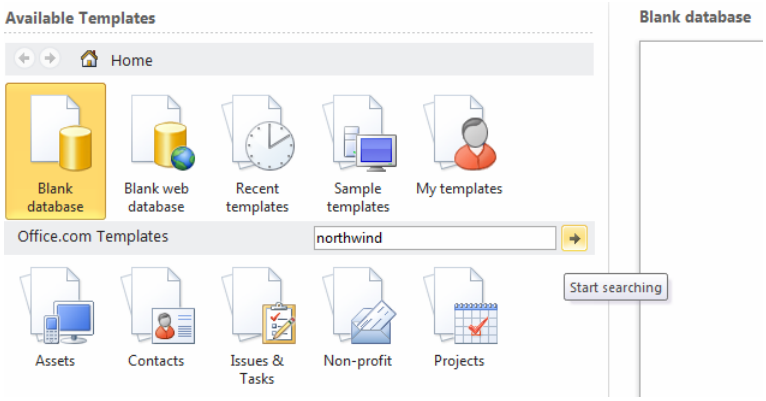


Figure A-1. Starting the template search

3. There should be two templates returned by the search, as shown in Figure A-2. `Northwind 2007` is a client application that was provided with Access 2007. `Northwind web database` is the web database developed for Access 2010. Select `Northwind web database`, select a download location, and click the Download button. While the template is being installed, you will see the dialog box shown in Figure A-3.

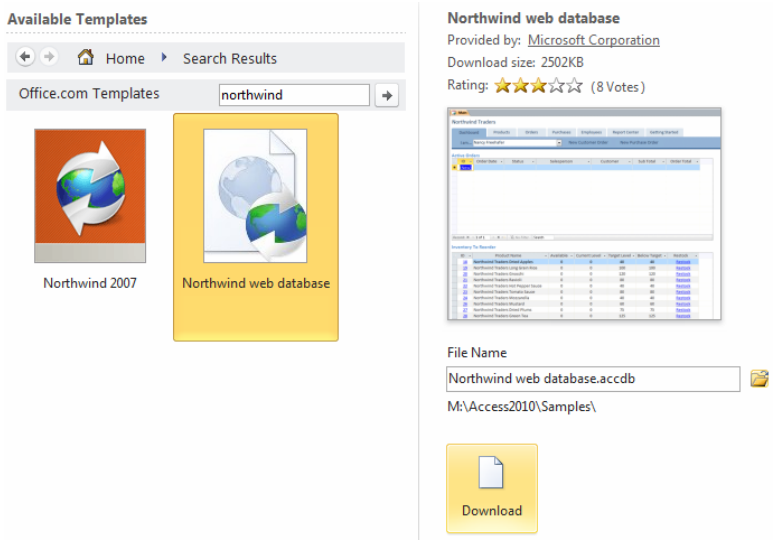


Figure A-2. Selecting the `Northwind web database`

www.it-ebooks.info

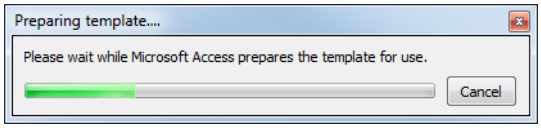


Figure A-3. *The template installation progress dialog box*

4. When the template has been downloaded, a new `.accdb` file is created in the selected folder based on this template. This database is then opened. The first thing you'll see is the `Login` form shown in Figure A-4. Select any employee to continue.

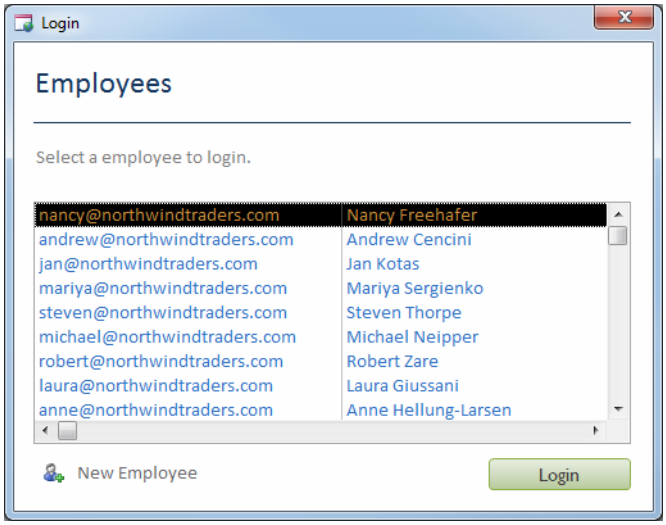


Figure A-4. *Selecting an employee to impersonate*

Publishing the Database

You publish a web database to a SharePoint server from the Save & Publish tab of the Backstage view. You should first verify the database is compatible with the Access Services in SharePoint.

1. Go to the Backstage view and select the Save & Publish tab. Then select the Publish to Access Services tab.
2. Click the Run Compatibility Checker button.
3. Because the `Main` window is open, you'll see the dialog box shown in Figure A-5. Click the Yes button to close all open objects.

www.it-ebooks.info



Figure A-5. Closing all open objects

4. The Compatibility Checker should not find any issues. Enter the URL of the SharePoint server, as shown in Figure A-6. The site name will default to Northwind Traders. Click the Publish to Access Services button.

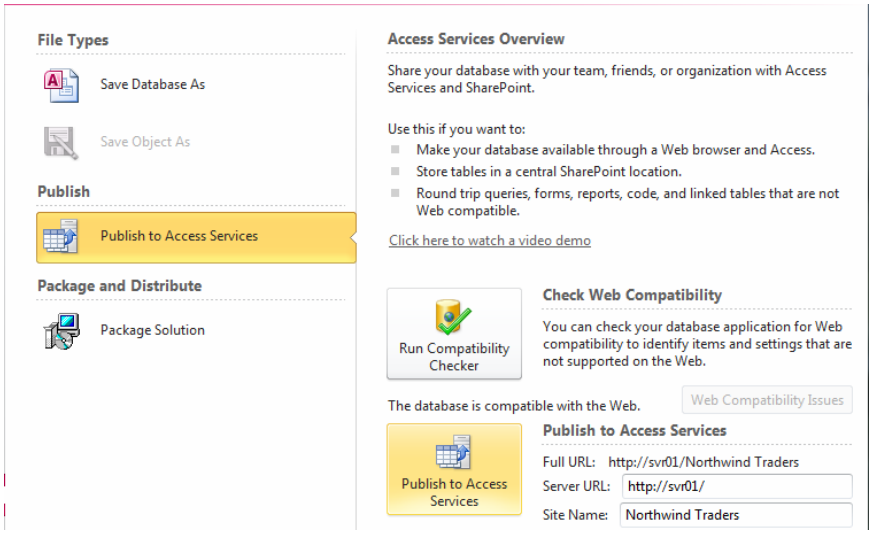


Figure A-6. Specifying the SharePoint server and site name

When the database has been successfully published, you'll see the dialog box shown in Figure A-7.

www.it-ebooks.info

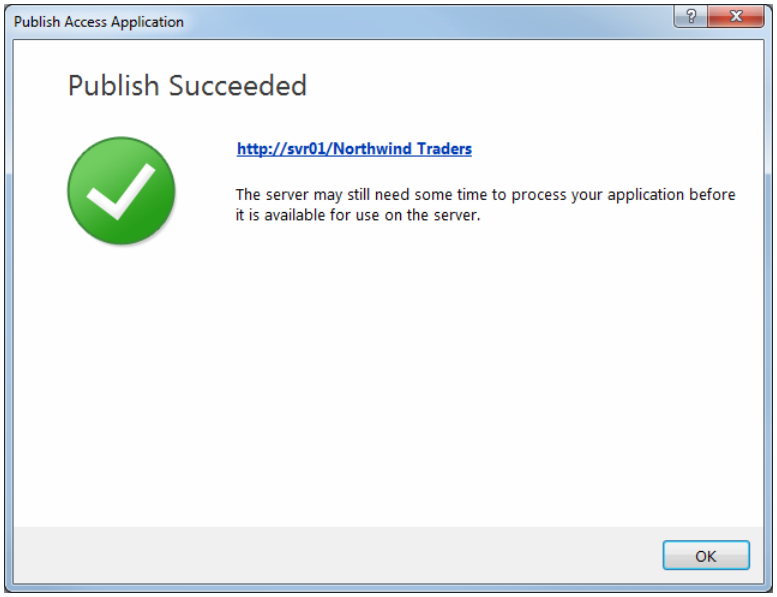


Figure A-7. *The Publish Succeeded dialog box*

Understanding the Sample Database

The Northwind web database provides lots of good examples for implementing a web database. I'll show you one of them and let you explore the application to find others.

Setting Up Your Employee Record

When you go to the SharePoint site you'll see one of the immediate benefits of hosting a web database in SharePoint. SharePoint utilizes integrated security to authenticate the users of the site. An `Employee` record is created for you but all SharePoint knows is your login so you'll need to supply the other details.

Click the link on the page shown in Figure A-7 to go to the new SharePoint site. The initial page is shown in Figure A-8.

www.it-ebooks.info

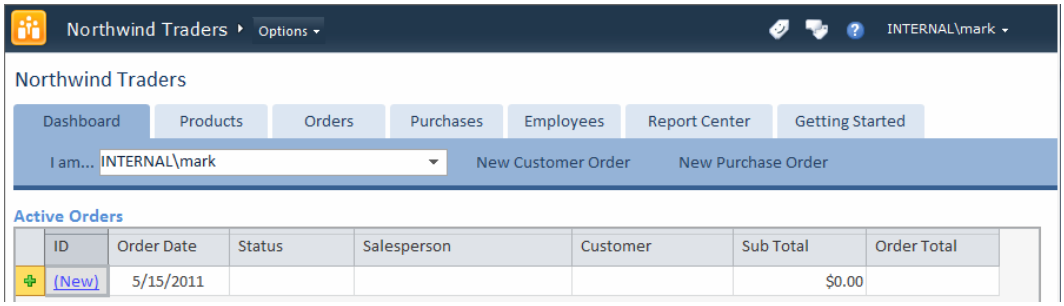


Figure A-8. *The initial page of the SharePoint site*

Notice that the `Login` page was not displayed; instead SharePoint determined the current user based on your Windows login. Select the Employees tab and you'll see a new `Employee` record was created for you as demonstrated in Figure A-9.

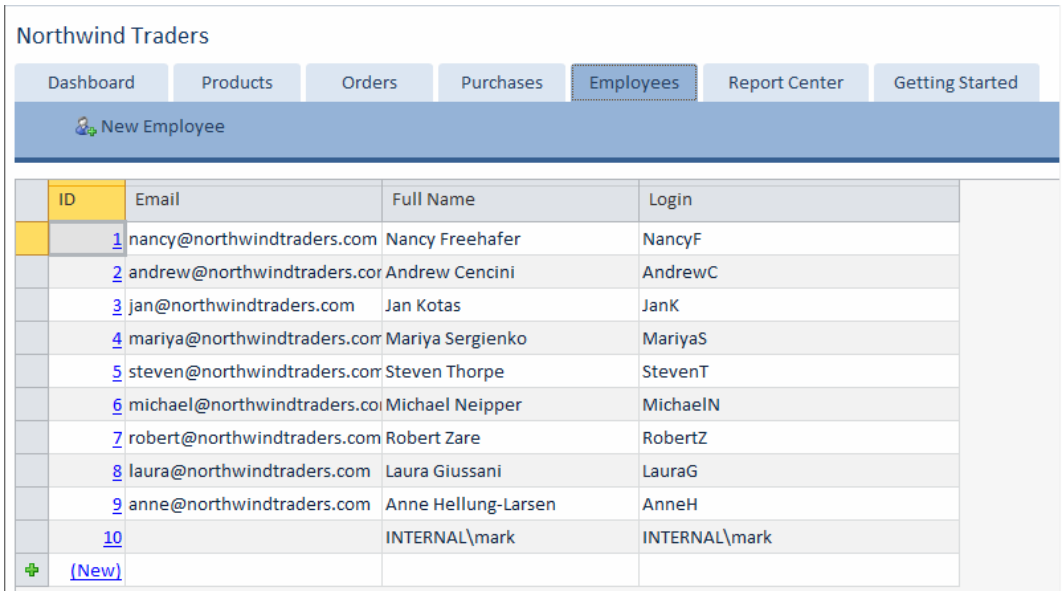


Figure A-9. *The Employees tab showing the new record*

Click the ID field of the new record, which is a link to the edit form, shown in Figure A-10. Enter the Full Name and E-mail fields and click the Save & Close button.

www.it-ebooks.info

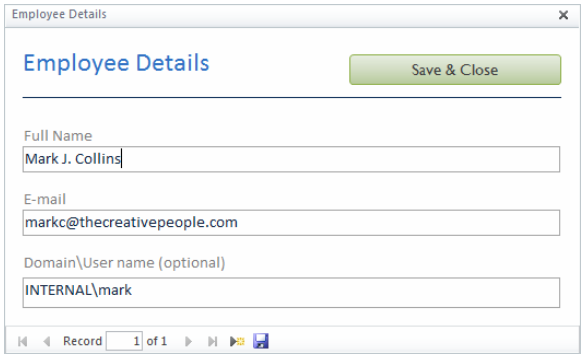


Figure A-10. *Entering your employee details*

Exploring the Access Client Application

Now go back to Access file. To refresh the application, close the file and then re-open it. Notice that the `Login` page is skipped and the `Main` form has defaulted the employee, as shown in Figure A-11.

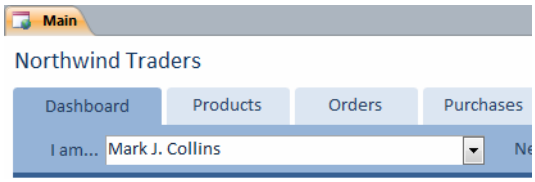


Figure A-11. *The `Main` form with the employee defaulted*

Go to the Employees tab and you'll see the employee details that you entered in the SharePoint site, as demonstrated in Figure A-12.

www.it-ebooks.info

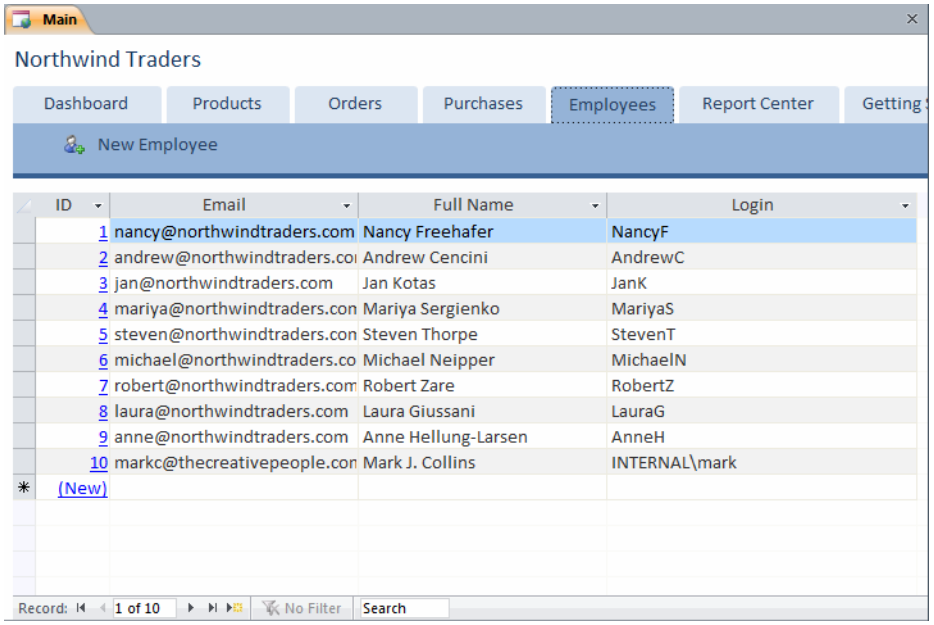


Figure A-12. *The Employees tab displayed in the client application*

To understand how the current user was determined, go open the `Main` form using the Layout View. In the Property Sheet, select the Form object and the Event tab. Notice that there is an embedded macro that handles the `OnLoad` event as shown in Figure A-13.

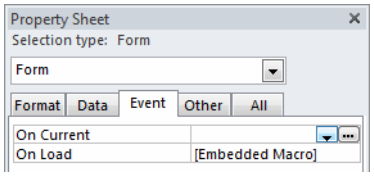


Figure A-13. *An embedded macro to handle the* `OnLoad` *event*

Click the ellipses next to this event, which will display the Macro Editor. The macro implementation is shown in Figure A-14.

www.it-ebooks.info

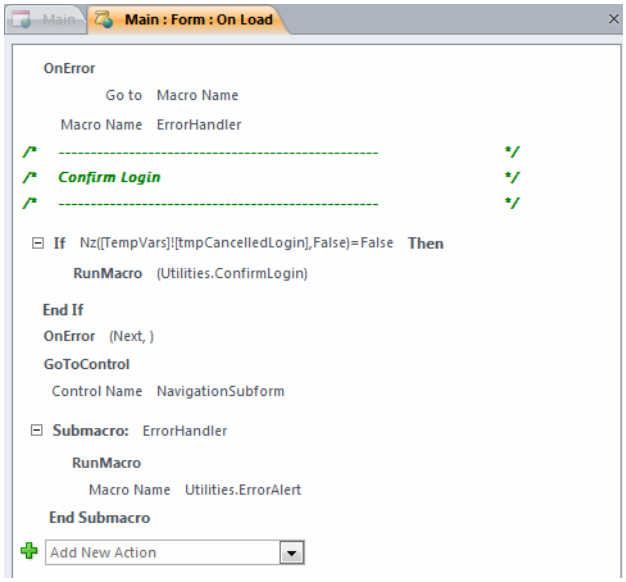


Figure A-14. *The implementation of the* `OnLoad` *event*

The real work is done by the `Utilities.ConfirmLogin` macro, which is called by the `RunMacro` action. The implementation of this submacro is shown in Figure A-15.

www.it-ebooks.info

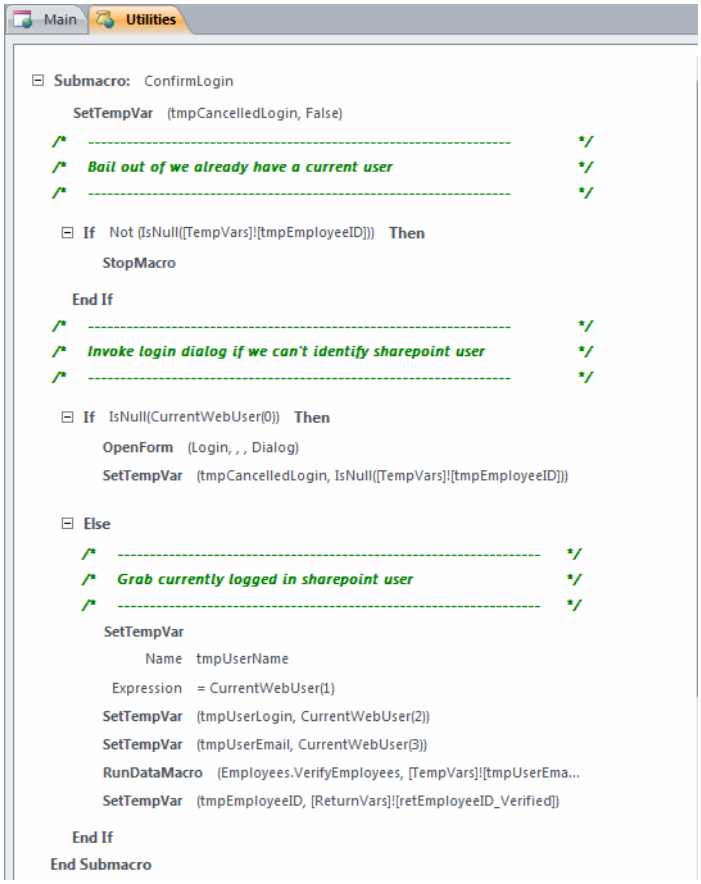


Figure A-15. *The implementation of the* `Utilities.ConfirmLogin` *submacro*

This macro takes advantage of the `CurrentWebUser` function. To see what this function does, open one of the expressions using the Expression Builder. Then select this built-in function as shown in Figure A-16.

www.it-ebooks.info

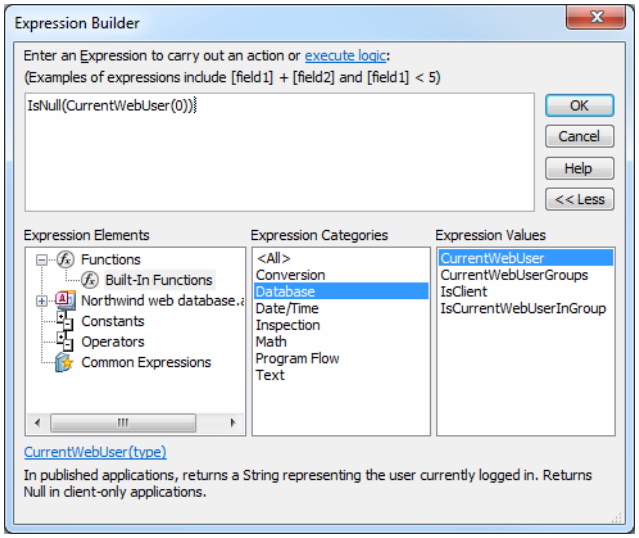


Figure A-16. *The* `CurrentWebUser` *function details*

The `CurrentWebUser` function only works in a published database like this one. It communicates with the SharePoint server to determine who is the currently logged in user and returns the name, login, or e-mail, depending on the type parameter that is passed in.

The `Utilities.ConfirmLogin` macro opens the `Login` page if the current user cannot be determined.

www.it-ebooks.info

Index

Special Characters & Numbers

- (minus) button, 61
#region blocks, 58
* field, 84
+ (plus) button, 61
< button, 99
<< button, 33, 78
=GetFullImagePath(txtPicture), 239, 253
2NF (second normal form), 18
3NF (third normal form), 18

A

.accdb file, 396–397, 406, 553
.accdc extension, 542
.accde extension, 288
.accde files, 396, 401–402
.accdr file, 396–397, 401
.accdw file, 442
Accent Color definitions, 295
Access, Microsoft. *See* Microsoft Access
Access Option dialog box, 268
Access Options dialog box, 262
Access Runtime, 393–397
    downloading, 397
    filename extensions for, 396–397
    simulating, 393–396
Access Services component, publishing databases to, 422–423
acNewRec constant, 186
Action Catalog, 58
action queries, 87–96
    AddInventoryItem, 87–91
        adding parameters, 89–90
        changing query type, 88–89
        running, 90–91
    Request feature, 91–96
        adding data macro, 94–95
        CancelOldRequests query, 93–94
        modifying Request table, 91–93
        running query, 95–96
actions, data, 60
Add document link, 445
Add Existing Fields button, 163, 175
Add Inventory, 449
Add New Action dropdown lists, 56, 60
Add or Remove Features option, 403
add-ins, data collection, 472–474
AddInventory macro design, 249
AddInventoryItem queries, 87–91
    adding parameters, 89–90
    changing query type, 88–89
    running, 90–91
Address control, 194
Address quick start field, 23
.adepsws file, 406
ad-hoc filters, 196
ad-hoc queries, 173–175
Administration group, 273–274
Administration page, 260
after events category, 53
After Insert button, 54, 57, 71
After Insert event, 54
After Update button, 71, 94
After Update macro, 94
Alias field, 56, 59
aliases, 59–60
All Relationships button, 37
AllLoans queries, 96–98, 104–105, 114
AllLoans reports, 317–335
    configuring grouping and sorting of, 329–331
    Design View view, 325–327
    Layout View view, 327–329
    Report Wizard function, 317–325
        grouping records, 318–321
        selecting data sources, 317–318
        selecting format, 323–325
        sorting and summarizing details, 322–323
    unbound controls, 332–335

www.it-ebooks.info

AllLoans reports (*cont.*)
displaying current date and time, 333–334
fixing percentage formula, 335
formatting page numbers, 332–333
totals and subtotals, 334–335
AllLoans view, 390
Allow Additions property, 155, 196
Allow Datasheet View property, Property Sheet, 211
Allow Deletions property, 155
Allow Filters property, 196
Allow Full Menus option, 287
Allow Zero Length property, 23, 26
animation, for timers, 513–515
APIs (application programming interfaces), 515–517
AppData folder, 407
Append button, 88
append query, 87
Append To row, 88
AppImages library, 443–444, 446
Application Log table, 73, 75
Application object, 517
application programming interfaces (APIs), 515–517
applications
choosing architecture of, 3–4
client, 557–561
sample, 4–13
testing, 75–76
Apply Theme to All Matching Objects link, 300
Apply Theme to This Object Only link, 299
ApplyFilter method, 160, 241
apress.com website, 291
ApressLogo image, 436
Apress.thmx file, 298
Arrange tab, 156, 182, 192, 198, 200, 208–209, 442
Ascending value, 85
Assigned To field, 495
AssignPendingRequest macro, 91
Author control, 149, 175
Author field, 149, 209
Auto_Header0, 436, 442
Auto_Logo0 control, 436
AutoCalc button, 107
autoexec macro, 281–282, 517–518, 532
auto-loading, Menu form, 262
AutoNumber Data Type, 45
AutoNumber field, 22, 28, 88, 232
AutoNumber type, 17
autorun.inf file, 410

B

Back Color property, 302
background images, 305–307
background/foreground colors, 294
Backstage view, 536
banners, 303–305
Basic page, 233–235
BasicClear property, 238
BasicClear_Click method, 241
BasicSearch property, 238
BasicSearch_Click method, 241
Before Change button, 62
Before Change event, 53, 69
Before Change macro, 63
Before Delete event, 53
before events category, 53
bitmap images, storing, 222
Black color, 301
Blank Form button, 198, 256, 435
Blank Report button, 312
Boolean field, 114
Border Color property, 164
Border Style property, 302
branding, 291–307
graphics, 303–307
background images, 305–307
banners, 303–305
Microsoft Office themes, 291–303
applying, 300–302
creating, 297–300
custom color, 292–296
font scheme, 296–297
visual adjustments to, 302–303
Bring to Front option, Position menu, 193
Browse link, 304
Browser property, 252
Build Event link, 203, 341
Build link, 85, 114

C

CalculateAllLateFees macro, 73–74
calculated columns, 85, 114
calculated fields, 38
CalculateLateFees macro, 260–261
CalculateLateFees UI, 391
callback method, 280–281
CancelOldRequests macro, 95, 260–261
CancelOldRequests queries, 93–94

www.it-ebooks.info

CanceOldRequests UI, 391
Caption property, 30, 196–199, 202, 212, 218
captions, of fields, 30
Captions tab, 108
Cascade Delete mode, 35, 41
categories
  collapsing, 116
  defining, 27
Category form, 133, 144, 146, 230, 305
Category tables, 26–27
Category web forms, 435–437
CategoryCode column, 127, 131
CategoryDescription column, 34, 111, 132, 211
CategoryDescription control, 176, 209
CategoryDescription fields, 27, 99, 108, 116, 127
CategoryID control, 149
CategoryID fields, 32, 471–472, 476, 483, 485
category/media type, 100
cbField, 235
cbMedia control, 237
certificates, 540–541
Change Chart Type button, 120
Change Shape button, 302
Chart control, 525–529
Chart Options link, 528
charts, changing type, 120–121
Checked Out status, 57
Checked Out value, 84
CheckedIn control, 209
CheckedIn date, 459
CheckedIn field, 48
CheckedIn property, 68
CheckedOut By Month field, 111, 116
CheckedOut By Week field, 111
CheckedOut column, 109, 429–430
CheckedOut control, 209
CheckedOut field, 208, 429, 431
CheckedOutItems query, 80
CheckOut button, 184–187, 259, 431
CheckOut forms, 153–190
  customer search feature, 153–172
    controls, 166–169
    CustomerDisplay form, 163–166
    CustomerSearch dialog box, 154–162
    invoking Search dialog, 169–172
    populating Customer table, 154
  designing details of, 180–184
    Detail section, 182–183
    Header and Footer controls, 183–184
    Loan table, 180–181
    LoanDetail query, 181–182
  implementing logic, 184–186
  InventoryItemLookup form, 173–178
    adding fields to, 175–176
    ad-hoc queries, 173–175
    finalizing details of, 176–178
    linking as subform to Checkout form, 178–180
    modifying design of, 176
  storing user name, 519–520
  testing, 187–190
CheckOut From Header, 179
CheckOut reports, 336–342
  data-bound controls, 336–337
  formatting Report Header form, 337–338
  modifying, 341–342
  subreports, 338–341
City table, 504
Clear Caption, 238
Clear_Click method, 160
"Click to Add" column, 23
client application, 557–561
Client Settings tab, 266
clocks, digital, 512–513
close button, 197
Close Button property, 197
Close_Click method, 241
CloseWindow action, 162
Code Builder option, 170
Code field, 26
Col1 option, 50
collapsing categories, 116
color picker, 294, 302
color properties, 337
color schemes
  modifying for Label Wizard function, 349–350
  viewing, 293
colors, custom themes, 292–296
Colors button, 292
Colors dialog box, 294
Column fields, 108–109, 115–116
column headings, changing, 102
Columnar layout option, for forms, 144–145
columns, 37
  adding to queries, 83–84
  calculated, 85, 114
  lookup
    adding to Item table, 39–41
    creating, 32–36
    with fixed options, 41–44
  MediaID, 37
Command Button Wizard, 159, 198, 257

www.it-ebooks.info

command buttons, 449–450
Command1_Click method, 162
CommandID control, 270
CommandID field, 281
Comment action, 55, 58
Comment field, 85
Compatibility Checker feature, 421–422
compatibility warning, Excel, 122
compiling databases, 287–289
Complete button, 183–184, 186
Complete_Click method, 520
computing overdue fees, 71–73
Condition control, 175
condition of items, specifying, 44
Conditional formatting feature, 242–245
connecting code, implementing, 202–205
constraints
  database, 19
  unique, 27
content, disabled, 531–535
Continuous Forms, 137–143
  creating, 137–140
  designing, 140–141
  modifying fields in, 141–143
Control Padding button, 164, 176, 210
controls
  arranging, 209
  customer, 166–169
  data-bound, 336–337
  Footer, 183–184
  form configuring, 195–197
  Header, 183–184
  Macro Editor, 61
  tab, 199–200
  unbound, 158, 332–335
    displaying current date and time, 333–334
    fixing percentage formula, 335
    formatting page numbers, 332–333
    totals and subtotals, 334–335
Controls section, ribbon, 198
Count function, 100, 107
Count link, 107
Country table, 504
CountryRegion field, 23
Create Named Macro link, 69
Create New Theme Colors dialog box, 293
Create New Theme Fonts dialog box, 296
Create ribbon, 26
Create tab, 191, 198, 208, 502
CreateRecord action, 59, 448
Criteria row, 92
cross tabulation, 96
crosstab queries, 96–102
  AllLoans query, 96–98
  LoanSummary query, 98–102
Crosstab Query Wizard, 98
currency table, 508
currency.xml file, 506
Current Database tab, 265
CurrentDb object, 517, 520
CurrentLoanID columns, 443
CurrentLoanID ComboBox control, 141
CurrentLoanID field, 81, 141
CurrentProject.FullName property, 227
CurrentView property, 216
CurrentWebUser function, 560–561
custom hover text, 283
custom ribbons, 266
Customer Admin form, 5–6
customer controls, 166–169
customer data, searching, 172
Customer Experience Improvement Program option, 376
Customer list, 11, 426
Customer List web forms, 438
Customer Profile tab, 191–206
  configuring form controls, 195–197
  CustomerUpdate form, 191–195
  designing Form Header, 198–199
  implementing connecting code, 202–205
  Profile page, 200–202
  tab controls, 199–200
  testing page, 205–206
Customer record, 198
customer search feature, 153–172
  controls, 166–169
  CustomerDisplay form, 163–166
  CustomerSearch dialog box, 154–162
    Detail fields, 155–157
    Search fields, 157–159
    Search logic, 159–162
  invoking Search dialog, 169–172
  populating Customer table, 154
Customer tables, 21–26
  adding customers to, 25–26
  input masks, 24–25
  populating, 154
  quick start fields, 22–23
  referencing, 45–47
  search fields, 23
CustomerAdmin code file, 218
CustomerAdmin form, 191–220

www.it-ebooks.info

Customer Profile tab, 191–206
configuring form controls, 195–197
CustomerUpdate form, 191–195
designing Form Header section, 198–199
implementing connecting code, 202–205
Profile page, 200–202
tab controls, 199–200
testing page, 205–206
Items on Loan tab, 206–217
configuring Datasheet view, 211–212
connecting pieces, 214–217
CustomerLoan form, 208–210
enhancing LoanDetails query, 207–208
Items on Loan page, 212–217
Loan History tab, 218–220
CustomerDisplay form, 163–166, 196–198, 200–206
CustomerID column, 23, 464, 471–472
CustomerID control, 192–194, 209
CustomerID field, 192, 205, 471, 479
CustomerID filter, Datasheet View, 214
CustomerLoan form, 208–210
arranging controls, 209
initial form, 208–209
modifying, 210, 253–254
CustomerLoans code file, 215
CustomerLoans form, 212, 215, 218, 514–515
CustomerLoans object, 215
CustomerLoans.Form syntax, 215
CustomerLoansHistory form, 219
customers, adding to Customer tables, 25–26
CustomerSearch dialog box, 154–162
Detail fields, 155–157
Search fields, 157–159
Search logic, 159–162
CustomerSearch form, 156, 159, 161, 166, 229, 238, 241
CustomerUpdate form, 191–197, 201–205
CustomerUpdate subform, 201, 203
Customer.xlsx file, 154

D

DailyLoans reports, auto-generating, 350–357
DailyReport macro, 351–357
scheduled tasks, 353–356
DailyReport macro, 351–353, 356–357
Darker attribute, 302
data actions, 58, 60
data blocks, 58–59
Data Blocks action, 58
data collection, 470–487
e-mail, 471–483
installing data collection add-in, 472–474
resending, 486–487
results table, 472
processing replies, 483–486
data contexts, 59
data files, names of in Microsoft Outlook, 497–498
data integrity, 35
data macros, 53–76, 448–449, 460–461
adding current Loan record reference, 66–68
Loan After Insert event, 66–67
Loan After Update event, 68
Lookup field, 66
adding to Request feature, 94–95
computing overdue fees, 71–73
creating, 54–57
debugging, 73–75
LogEvent action, 74–75
viewing Application Log table, 75
handling requested items, 68–71
implementing Before Change event, 69
named data macro, 69–71
limitations of, 54
Loan Before Change event, 61–65
checking item availability, 61–62
due dates, 63
late fees, 63–64
validating renewals, 65
Macro Editor tool, 58–61
aliases, 59–60
data actions, 60
data blocks, 58–59
navigating, 61
testing, 57, 75–76
updating, 416–421
and upsizing, 390–391
Data Mode property, 95
data prompt, 31
data source, 309
Data tab, Property Sheet, 202, 209, 212–213, 218
database schema, 20
Database Splitter dialog box, 363
Database Tools ribbon, 37, 51
databases
constraints, 19
encrypting, 548–550

www.it-ebooks.info

databases (*cont.*)
locking, 283–289
database, 287–289
removing navigation, 284–287
normalizing, 18–19
Northwind web. *See* Northwind web database
publishing, 422–433
to Access Services component, 422–423
fixing default value, 429–431
Microsoft SharePoint sites, 424–427
restoring Microsoft VBA code, 428–437
web databases, 431–433
signing, 540–547
certificates, 540–541
installing, 543–547
packaging databases, 541–542
data-bound controls, 336–337
data-bound images, 221–229
adding to item tables, 223–224
displaying, 226–229
code for, 226–227
loading image files, 228–229
modifying item forms, 224–225
support for, 221–223
Datasheet Caption property, Property Sheet, 211
Datasheet layout option, for forms, 144
Datasheet View, 21, 131–132, 211–212, 215–216, 218–219, 508
date, displaying current, 333–334
date fields, 109–111
date properties, 110
datetime fields, 415
DaysOverdue column, 85
db_datawriter role, 381
debugging, 73–75
LogEvent action, 74–75
viewing Application Log table, 75
Decimal Places property, 29
Default Value property, 180
default values, 30–31, 429–431
Default View property, 214
default web forms, setting, 440–441
defining theme colors, 295
Delete Row link, 149
Demo ribbon, 265–266, 282
Demo tab, 263, 266
denormalization, 66
denormalized query, 87
Descending value, 85
Description control, 150, 176, 210, 521
Description field, 149, 521–522, 524
Description text, 524
DescriptionHTML, 524
design practices, for tables, 17–19
databases, 18–19
primary keys, 17
design report rules, 311
Design ribbon, 26
Design tab, ribbon, 198–199, 201, 203, 212, 214–215, 218
Design View button, 21, 130
Design View view, 325–327, 329, 341
Detail fields, 155–157
Detail section, 182–183, 195, 197, 199, 238–240
dicing, 103–104
digital clocks, 512–513
Digital Signature Details dialog box, 545
digital signatures, 540
dimensions, 103
Disable Trusted Document check box, 539
disabled content, 531–535
Display Navigation Pane option, 288
DisplayCurrentLoan method, 216–217
DisplayImage method, 227, 400
distributing applications, 393–413
installation package for, 402–413
preparing application for, 397–402
creating executable file, 401–402
relinking linked tables, 400–401
supporting images in, 398–400
using Access Runtime, 393–397
downloading, 397
filename extensions for, 396–397
simulating, 393–396
DoCmd.OpenQuery, 517
documents, trusted, 531–539
configuring, 538–539
disabled content, 531–535
trusted locations, 536–538
dotted lines, in reports, 313
double-click method, 84
downloading, Northwind web database template, 551–553
drilling, 103–104
due dates, calculating, 63
DueDate column, 464
DueDate control, 209
DueDate field, 48, 495, 515

www.it-ebooks.info

E

EditRecord action, 56, 59–60, 63
EditRecord data block, 66
e-mail
  data collection, 471–483
    installing data collection add-in, 472–474
    resending, 486–487
    results table, 472
  sending, 457–463
    data macros, 460–461
    OverdueItems query, 457–459
    UI macros, 461
    with VBA, 463–470
E-mail field, 479
Employee record, setting up, 555–557
Employees tab, 556
Enable Data Integrity checkbox, 37, 41, 47, 66
encrypting databases, 548–550
ErrorExit label, 227
event handlers, Profile Page, 203
Event Procedure option, Property Sheet, 203
Excel Microsoft, exporting PivotTable views to, 122–123
exchange rates, importing, 505–509
executable file, creating for distributed application, 401–402
ExitForEachRecord action, 70
Export to Excel button, 122
exporting data, to Excel, 123
Expression Builder, 29, 334–338
Extensible Markup Language. *See* XML
external Access database options, 230
external data, 489–509
  importing XML files, 503–509
    exchange rates, 505–509
    process of, 504
  linking
    and importing, 489
    Microsoft Outlook folders, 497–503
    Microsoft SharePoint lists, 489–496

F

FaithToFaith query, 503
fees
  calculating late, 63–64
  computing overdue, 71–73
field captions, changing, 113
Field List button, 106
Field List window, 106
Field Properties pane, 41, 43, 48
Field Properties window, 180
field validation prompt, 31
fields, 106–108, 112–117
  adding to Loan tables, 48–49
  calculated, 38
  calculated columns, 114
  captions, 30
  Column, 108–109, 115–116
  date, 109–111
  Filter, 111–112
  hierarchy, 116–117
  multiple values, 112–114
  rearranging in PivotTable View, 117
  required, 26–27
  switching, 119
  validation, 29–30
File Download dialog box, 432
File Info page, 75
file name extensions, for Access Runtime, 396–397
file system, storing images in, 223
files, data names of in Microsoft Outlook, 497–498
Files folder, 410
filter, 196
Filter fields, 111–112
Filter On Load property, 178, 197
Filter property, 197, 214, 216
filter values, selecting, 111
final installation dialog box, 413
Finish button, 198
Finish_Click method, 204
FirstName field, 22
fixed options, 41–44
folders, Microsoft Outlook, 497–503
  data file names, 497–498
  linked tables, 498–503
font schemes, 296
FontBold property, 514
fonts, schemes, 296–297
Fonts button, 296
Footer control, 183–184
Fore Color property, 301, 306
ForEachRecord action, 59, 460
ForEachRecord data block, 460
Form button, ribbon, 191, 208
form controls, configuring, 195–197
Form Header layout control, 158, 199
Form Header section, 168, 198–199

www.it-ebooks.info

Form Header/Footer link, 155, 198
form navigation, 255–256, 261–262
Form object, 130, 211
Form property, 215
Form View, 195–197, 206, 208, 212–213, 215–216, 219
Form Wizard, 127–130
Form_Current method, 165, 241, 250, 523
Form_Deactivate handler, 513
Form_Item code file, 400
Form_Load event, 214, 218
Form_Load method, 246, 513, 518
Form_Timer method, 514, 518
Format property, 48
Format tab
  Caption property, 199
  Height property, 194
  Property Sheet, 193–194, 202, 210, 212, 218
  Visible property, 193
FormatDateTime( ) function, 63
Form.InventoryItem, 151
forms, 127–152
  continuous, 137–143
    creating, 137–140
    designing, 140–141
    modifying fields in, 141–143
  layout options for, 143–147
    Columnar layout option, 144–145
    Datasheet layout option, 144
    implementing, 146–147, 149–150
    Justified layout option, 145
    Tabular layout option, 143–144
  sorting records in, 132–133
  split, 133–137
    creating, 133–135
    modifying fields in, 135–137
  subforms, adding, 150
  using Form Wizard, 127–130
  views for, 130–132
full joins, 83
Function Name parameter, 281

G

General tab, 44
GetFullImagePath function, 226, 399–400, 524
getImage attribute, 263
GetImage function, 264
GetRibbonDefinition method, 280–282
GetStaticImagePath function, 400
GetUser( ) method, 518
GetUserName method, 515–516
GoToPage action, 252
GoToRecord method, 185–186
Grand Total column, 108
Graphic field, 281
graphics, 303–307
  background images, 305–307
  banners, 303–305
Group action, 58, 61
group configuration options, 330
group customization, 321
group sections, of reports, 311–312
GroupID control, 270
GroupID field, 271
GroupID lookup column, 270
grouping
  configuring, 329–331
  with Report Wizard function, 318–321
Grouping Options button, 323

H

HappyFace icon, 264
hash symbol, 327
Header control, 183–184
Height property
  Format tab, 194
  Property Sheet, 210
Hide Details button, 107
Hide key column checkbox, 35, 40, 46
History page, ribbon, 218
Hover Color property, 302
HTML content, 520–524
  Item form, 521–522
  passing to Web Browser control, 522–524
Hyperlink Data Type, 224

I

id attribute, 263
ID field, 495
If action, 58, 69
images
  background, 305–307
  bitmaps, storing, 222
  data-bound, 221–229
    adding to item tables, 223–224
    displaying, 226–229

www.it-ebooks.info

modifying item forms, 224–225
support for, 221–223
loading files, 228–229
storing in file system, 223
storing in Microsoft SharePoint, 444–446
supporting in distributed application, 398–400
Images folder, 228, 233, 303, 407–408, 446
Images\Static folder, 228, 284, 399, 409
imgPicture property, 225, 239, 253
importing
external data, 489
XML files, 503–509
importing exchange rates, 505–509
process of importing, 504
infinity symbol, 82
inner joins, 83
InProgress column, 520
InProgress control, 208
InProgress field, 180
InProgress flag, 186
Input Mask property, 24
Input Mask Wizard, 24
input masks, 24–25
Insert Above button, 303
Insert Below button, 200
Insert Image button, 304, 306
Insert Page link, 218, 256
Insert Right button, 158, 192, 198, 200, 209
inserting records, 87
installation completion dialog box, 377
installation package, 402–413
installing databases, 543–547
inventory items, adding to Item form, 247–249
InventoryItem alias, 59
InventoryItem form, 139, 146–147, 150
InventoryItem labels, 343–350
InventoryItemDetail queries, 343–344
Label Wizard function, 345–348
formatting labels, 346–348
selecting label stock template, 345–346
modifying color scheme, 349–350
InventoryItem records, 40, 54, 173, 391, 416–417, 440, 444, 448
InventoryItem subforms, 248–249, 441, 443–444
InventoryItem tables
records, 44–45
referencing, 47–48
InventoryItem web form, 444
InventoryItem.AddInventoryItem data macro, 449
InventoryItemDetail queries, 343–344
InventoryItemID control, 209
InventoryItemID fields, 51, 173, 443
InventoryItemID label, 141
InventoryItemLookup form, 173–178
adding fields to, 175–176
ad-hoc queries, 173–175
finalizing details of, 176–178
linking as subform to Checkout form, 178–180
modifying design of, 176
InventoryItem.Status field, 84
IsInsert property, 53
IsLoaded function, 170
IsNull function, 114
Item client, 442
item data, importing, 230–233
Item forms, 247–254, 521–522
adding inventory items to, 247–249
adding WebBrowser control to, 249–251
CustomerLoan form, 253–254
modifying, 224–225
page breaks, 251–253
Item pictures
displaying, 446–447
and Web Browser control, 444
Item Search form, 7–8
Item tables, 32–38
adding data-bound images to, 223–224
adding items, 38
calculated fields, 38
columns
lookup, 32–36, 39–41
MediaID, 37
relationships, 37
Item web forms, 442–453
command buttons, 449–450
data macros, 448–449
initial form, 442–443
InventoryItem records, 448
InventoryItem subform, 443–444
Item pictures
displaying, 446–447
and Web Browser control, 444
Navigation form, 451–453
search features, 450–451
storing images in Microsoft SharePoint, 444–446
Web Browser control, 447
Item1 table, 231–233
Item.accdb file, 230–231

www.it-ebooks.info

ItemCount: LoanID, 458
ItemID ComboBox control, 141
ItemID control, 140, 149
ItemID field, 442
itemID parameter, 90, 448–449
ItemInventory tables, 39–45
  adding InventoryItem table records, 44–45
  adding lookup columns
    with fixed options, 41–44
    to Item table, 39–41
  specifying item condition, 44
items
  adding to Item tables, 38
  checking availability of, 61–62
  handling requested, 68–71
    implementing Before Change event, 69
    named data macro, 69–71
  specifying condition of, 44
Items on Loan page, 212–213, 217
Items on Loan tab, 206–217
  configuring Datasheet view, 211–212
  connecting pieces, 214–217
  CustomerLoan form, 208–210
    arranging controls, 209
    initial form, 208–209
    modifying layout, 210
  enhancing LoanDetails query, 207–208
  Insert Page link, 218
  Items on Loan page, 212–213, 217
ItemSearch dialog box, 242, 524
ItemSearch form, 229–247
  Conditional formatting feature, 242–245
  designing layout of, 233–241
    basic Search page, 233–238
    detail section, 238–240
    VBA code, 240–241
  importing item data, 230–233
  invoking, 245–247
  testing, 241

J, K

JOIN logic, 86
Join Properties dialog box, 82–83
Join Properties link, 82, 97
joins, 81–84
  adding columns to queries, 83–84
  Join Properties dialog box, 82–83
  multiplicity, 82
.jpg file, 298
Justified layout option, for forms, 145

L

label attribute, 263
Label control, 164, 195, 197
Label field, 277
Label Wizard function, 345–348
  custom template, 345
  formatting labels, 346–348
  maximum lines, 347
  selecting label stock template, 345–346
labels
  formatting, 346–348
  selecting stock template, 345–346
Last Record button, 196
LastName control, 193
LastName field, 22
late fees, calculating, 63–64
layout control, 192
layout options, for forms, 143–147
  Columnar layout option, 144–145
  Datasheet layout option, 144
  implementing, 146–147, 149–150
  Justified layout option, 145
  Tabular layout option, 143–144
Layout View, 191–192, 194, 201, 209, 212, 218, 327–329
lblAddress control, 164
lblDate, 512, 515
lblEmail control, 164
lblName control, 164
lblPhone control, 164
lblRegion control, 164
lblTime, 512, 515
lblTitle control, 177
Legend button, 118
Library application, 221, 291, 390, 394, 398, 413, 489, 492
Library certificate, 542
Library database, 297, 373, 379–381, 519, 542, 551
Library Management System (LMS), 284
Library menu controls, 276
Library ribbon, 282–283
Library SQL Server database, 389
Library tab, 282
Library_Access database, 428
Library_Access_be.accdb database, 363, 365
Library_Access_be.accdb file, 399–400

www.it-ebooks.info

Library_Access.accdb file, 399, 401, 404, 428
Library_Access.accde file, 401–402, 406
Library_Access.msi file, 410
Library_Azure.accdb file, 382, 387
Library_SharePoint database, 428, 430–431
Library_SharePoint.accdb file, 416, 428, 433
Library_SQL.accdb database, 388
Library.accdb file, 21, 262, 363, 457, 532, 541, 546, 548
Library.accdc file, 543
Library.accde file, 288
Library.accdw file, 432
Library.ico file, 284, 407
LibraryUser, 380, 386
License Management dialog box, 378
License Refreshed notification, 378
Lighter attribute, 302
Like operator, 160
Limit to List property, 36, 41, 43, 66
Line chart, 120
Line type, 120
Link Child Fields property, 168, 202, 212, 218
Link Master Fields property, 168, 202, 212, 218
Link Tables checkbox, 384
Linked Table Manager, 364, 373–374, 388, 401
linked tables
  Microsoft Outlook, 498–503
  Microsoft SharePoint, 492–496
  relinking, 400–401
  upsizing using, 361–362
linking
  external data
    Microsoft Outlook folders, 497–503
    Microsoft SharePoint lists, 489–496
  views in SQL Azure, 389–390
lists, Microsoft SharePoint, 489–496
  linked tables, 492–496
  Team SharePoint site, 490–491
LMS (Library Management System), 284
LoadCustomUI method, 280
LoadRibbon method, 281, 532
Loan - InventoryItem join, 83
Loan After Insert event, modifying, 66–67
Loan After Update event, modifying, 68
Loan Before Change event, 61–65
  calculating, 63–64
  checking item availability, 61–62
  validating renewals, 65
Loan History tab, 219
Loan records, adding current reference, 66–68
  Loan After Insert event, 66–67
  Loan After Update event, 68
  Lookup field, 66
Loan tables, 45–49
  altering, 180–181
  fields, 48–49
  referencing
    Customer table, 45–47
    InventoryItem table, 47–48
Loan.BeforeChange, 416
Loan.CheckedIn, 458
LoanDetail queries, 181–182, 207–208
LoanID field, 107, 214
Loan.LoanID, 458
LoanPeriod column, 29
LoanPeriod control, 210
Loan.SendOverdueEmails, 461
LoanSummary queries, 98–102
_local suffix, 370
locations, trusted, 536–538
Locked property, 155
locking database, 284
LogEvent action, debugging, 74–75
logicCheckOut, implementing, 184–186
Login form, 553
Login page, 556–557, 561
logo image, 197
Look Up A Record In parameter, 55
lookup columns
  adding to Item table, 39–41
  with fixed options, 41–44
Lookup field, 66
Lookup tab, 41
Lookup Wizard, 32–33, 44–45, 87
LookupRecord actions, 55, 59, 416
LookupRecord block, 56
LostFee control, 149, 449
LostFee Label, 150
LostFocus event, 180, 227

M

Macro button, 95
Macro Editor tool, 58–61
  aliases, 59–60
    default, 59
    EditRecord action, 60
    when to use, 60
  data actions, 60
  data blocks, 58–59
  navigating, 61

www.it-ebooks.info

Macro Name property, 71
macros
  data, 416–421, 448–449, 460–461
  named data, 69–72
    calling, 71
    creating, 69–70
  UI, 461
  user-executed, 73
Main code file, 516
Main form, 557–558
Main module, 171, 264, 280
Main object, 399
Main window, 553
Maintenance group, 274
Maintenance page, 260–261
Manage Data Collection Messages dialog box, 486
master database, 380
max function, 106
Media form layout, 136, 146
Media tables, 28–31
  default values, 30–31
  defining media types, 31
  fields
    captions of, 30
    validation, 29–30
Media Type chart, 529
media types, defining, 31
Media web forms, 437–438
MediaCode field, 31
MediaDescription columns, 211, 236
MediaDescription control, 176, 209
MediaDescription field, 37, 100, 111, 115–116, 236, 525, 527
MediaID column, 37, 135
MediaID control, 149
MediaID field, 135–136, 437, 442, 471–472, 476, 483, 485
MediaTypesChart, 528
Menu forms, 256, 261–262, 270–272
Menu tables, populating, 272–275
Me.Requery statement, 186
Microsoft Access
  description of, 3–4
  getting started with, 13–14
  sample application, 4–13
  upsizing with, 363–365
  use of this book to understand, 4
Microsoft Access Security Notice dialog box, 546
Microsoft Excel, exporting PivotTable views to, 122–123
Microsoft Office
  Object Library, 264
  themes, 291–303
    applying, 300–302
    creating, 297–300
    custom color, 292–296
    font scheme, 296–297
    visual adjustments to, 302–303
Microsoft Outlook, 457–487, 497–503
  data collection, 470–487
    e-mail, 471–483
    processing replies, 483–486
  data file names, 497–498
  folders, 497–503
    data file names, 497–498
    linked tables, 498–503
  linked tables, 498–503
  sending e-mails, 457–463
    data macros, 460–461
    OverdueItems query, 457–459
    UI macros, 461
    with VBA, 463–470
Microsoft Outlook 14.0 Object Library, 466
Microsoft SharePoint
  lists, 489–496
    linked tables, 492–496
    Team SharePoint site, 490–491
  sites, 424–427
  storing images in, 444–446
Microsoft VBA (Visual Basic for Applications) code, restoring, 428–437
Microsoft Windows API (application programming interface) calling, 515–517
migrating data, to SQL Azure, 381–387
minus (-) button, 61
Miscellaneous category, 260
mnu prefix, 269
mnuCommand form, 270
mnuCommand table, 269–271
mnuCommands query, 276–278, 280–281
mnuGroup form, 272
mnuGroup table, 269, 271
mnuTab table, 269–270
mnuTab.Label field, 277
Modal Dialog link, 154
modified form layout, 226
Modify button, 197, 204–205
ModifyCustomer_Click method, 204
More Colors link, 294
More Fields button, 22
More Forms button, 154

www.it-ebooks.info

More link, Design View, 329
.msi file, 402
mso image gallery, 276
MSO images, 275–276
MSysASO, 426
Multiple Items, 435
multiple tables, adding, 80
multiple values, 112–114
multiplicity, 82

N

Name property
  Design View, 196
  Other tab, 199
  Property Sheet, 212, 218
  TextBox control, 198
named data macros, 69–72
  calling, 71
  creating, 69–70
Named Macro button, 69
navigation
  forms, 255–256, 261–262
  of Macro Editor tool, 61
  removing from database, 284–287
  ribbons, 262–268
    displaying system objects, 267–268
    sample ribbon tab, 263–267
Navigation Buttons property, 196–197
Navigation Caption property, 196
navigation controls, 196
Navigation forms, 5, 439–440, 451–453
Navigation Options button, 267
Navigation Options dialog, 268
Navigation pane, 191, 208
Navigation web form, 439
New button, 198
New Query dialog box, 77
new record, 196
New Values property, 22
NewCustomer_Click method, 204
No Duplicates option, 27
None link, ribbon, 210
normalizing databases, 18–19
Northwind web database, 551–561
  client application, 557–561
  installing template, 551–555
    downloading, 551–553
    publishing database, 553–555
  setting up Employee record, 555–557
NotFound.tif file, 228, 398–400, 409
Now( ) function, 91

O

Object Linking and Embedding (OLE) objects, storing, 222–223
Office, Microsoft. *See* Microsoft Office
Office Object Library, Microsoft, 264
Office setup program, 402
Office.com site, 551
Old property, 68
OLE (Object Linking and Embedding) objects, storing, 222–223
On Error Goto ErrorExit statement, 227
On Error Resume Next statement, 217, 280
onAction attribute, 263
OnBeforeNavigate event, 250
OnClick event, 160, 162, 185, 204, 217, 246
OnCurrent event, 165, 217, 227, 241, 243, 521–522
OnLoad event, 558–559
OnLoan page, 212–214
OnMenuAction function, 264
OpenForm action, 257
OpenForm method, 170
OpenQuery action, 248
OpenRecordset code, adjusting, 374–375
OpenRecordset method, 375, 387
OpenTasks, 496
Operations group, 273
Operations page, 257, 260
Option Compare Database statement, 516
options, fixed, 41–44
OrderBy property, 219
Other tab
  Name property, 199
  Property Sheet, 211–212, 218
outer joins, 83
Outlook, Microsoft. *See* Microsoft Outlook
overdue fees, 71–73
OverdueFee field, 31, 112, 136, 209
OverdueItemDetails query, 463–465
OverdueItems query, 457–460

P

Package Solution Wizard
  creating installation package using, 402–413
  Dialog 1, 405

www.it-ebooks.info

Package Solution Wizard (*cont.*)
Dialog 2, 406
Dialog 3, 408
Dialog 4, 409
Package Wizard feature, 404
packaging databases, 541–542
page breaks, 251–253
Page control, 199
Page Footer, 311
Page Header, 311
Page keyword, 332
page numbers, 332–333
Page sections, of reports, 311
Page Setup button, 315
Page Setup tab, 313–315
Page1 button, 252
Page1 property, 252
Page2 button, 252
Pages keyword, 332
paper size
customization, 314
selecting, 315
parameters, adding to AddInventoryItem queries, 89–90
Parameters link, 89
parentheses, 281
percentage formula, fixing, 335
Phone Number mask, 24
PhoneNumber field, 24–25
Picture Alignment property, 306
Picture control, 224, 239
Picture field, 224, 239, 253, 446
PivotChart legend, 119
PivotChart view, 118–121
changing chart type, 120–121
configuring, 118–120
PivotChart View link, 118
PivotTable View button, 105
PivotTable View link, 104
PivotTable views, 103–123
exporting to Microsoft Excel, 122–123
fields, 106–108, 112–117
Column, 108–109
columns, 114–116
date, 109–111
Filter, 111–112
hierarchy, 116–117
multiple values, 112–114
layout of, 105–106
PivotChart view, 118–121
changing chart type, 120–121
configuring, 118–120
rearranging fields in, 117
refreshing, 112
slicing dicing and drilling, 103–104
PivotTable/PivotChart View, defining, 121
plus (+) button, 61
Position menu, Bring to Front option, 193–194
Primary Key button, 26, 39, 45
primary keys, 17
Print Preview Viewback color property, 349
Privacy Options button, 286
product administration, 221–254
data-bound images, 221–229
adding to item tables, 223–224
displaying, 226–229
modifying item forms, 224–225
support for, 221–223
Item form, 247–254
adding inventory items to, 247–249
adding WebBrowser control to, 249–251
CustomerLoan form, 253–254
page breaks, 251–253
ItemSearch form, 229–247
Conditional formatting feature, 242–245
designing layout of, 233–241
importing item data, 230–233
invoking, 245–247
testing, 241
Product Survey, 480
Profile page, 197, 200–202
CustomerDisplay form, 200
CustomerUpdate form, 200
event handlers, 203
Modify button, 197
Program Flow action, 58
Properties link, 108
Property Sheet
Allow Datasheet View property, 211
Caption property, 212, 218
Data tab, 209, 213
Datasheet Caption property, 211
Event Procedure option, 203
Format tab, 193–194
Height property, 210
Link Child Fields property, 202, 212, 218
Link Master Fields property, 202, 212, 218
Name property, 212, 218
Source Object property, 212, 218
Visible property, 202
Property Update Options button, 30
.pst file, 497–498

www.it-ebooks.info

Publish Succeeded dialog box, 555
publishing, 415–453
  databases, 422–433
    fixing default value, 429–431
    Microsoft SharePoint sites, 424–427
    publishing to Access Services component, 422–423
    restoring Microsoft VBA code, 428–437
    web, 431–433
  Northwind web database, 553–555
  preparing for, 416–422
    Compatibility Checker feature, 421–422
    updating data macros, 416–421
  web forms, 434–441
    Category, 435–437
    Customer List, 438
    Item, 442–453
    Media, 437–438
    Navigation, 439–440
    setting default, 440–441

Q

queries
  action, 87–96
    AddInventoryItem query, 87–91
    Request feature, 91–96
  adding columns to, 83–84
  ad-hoc, 173–175
  changing type, 88–89
  crosstab, 96–102
    AllLoans query, 96–98
    LoanSummary query, 98–102
  LoanDetail, 181–182
  running, 95–96
  select, 77–87
    adding tables, 80–81
    creating, 77–80
    making changes to, 84–86
    using joins, 81–84
    using queries as views, 87
Queries option, 98
Query Design button, 92, 181
Query Design table, 93
Query Name property, 95
Query Parameters dialog box, 89
Query Wizard button, 77
Quick Start category, 22
quick start fields, 22–23

R

RaiseError action, 65
Ready to install dialog box, 412
rearranging fields, in PivotTable View, 117
Received field, 502
record selector, 196–197
Record Selectors property, 196–197
Record Source property, 155, 174
records
  grouping, 311
  inserting, 87
  sorting in forms, 132–133
Rectangle shape, 302
referencing
  Customer table, 45–47
  InventoryItem table, 47–48
  Loan records, 66–68
Refresh Pivot button, 112
regedit.exe program, 547
relational database, 17
relationships, 37, 51–52
Relationships button, 37, 51
Remove Layout button, ribbon, 192–193, 208
Renew button, 217
Renew_Click method, Renew button, 217
renewals, validating, 65
Renewals control, 209
Renewals field, 48
RenewalsAllowed control, 135, 209
RenewalsAllowed field, 30, 135–136, 437
ReplacementCost control, 149, 224, 232
ReplacementCost field, 37–38
Report button, 312
Report Design button, 313
Report Header form, 337–338
report sections
  brief description, 309
  Detail, 311
  Footer, 309
  Header, 309
  Page Footer, 311
  Page Header, 311
  Report Footer, 311
  Report Header, 311
Report View, 309, 316
Report Wizard function, 317–325
  grouping records, 318–321
  sections, 326
  selecting data source, 317–318

www.it-ebooks.info

Report Wizard function (*cont.*)
selecting format, 323–325
selecting source, 317
sorting and summarizing details, 322–323
reports, 309–357
AllLoans, 317–335
configuring grouping and sorting, 329–331
Design View view, 325–327
Layout View view, 327–329
Report Wizard function, 317–325
unbound controls, 332–335
auto-generating DailyLoans, 350–357
DailyReport macro, 351–357
scheduled tasks, 353–356
CheckOut, 336–342
data-bound controls, 336–337
formatting Report Header form, 337–338
modifying, 341–342
subreports, 338–341
creating with
Blank Report button, 312
Report button, 312
Report Design button, 313
data source, 309
export to PDF, 351–352
and forms, 309
grouping and ordering, 312
grouping manually, 321
InventoryItem labels, 343–350
InventoryItemDetail queries, 343–344
Label Wizard function, 345–348
modifying color scheme, 349–350
Label Wizard, 312
Layout View, 312
manipulating content, 309
margin configuration, 314
nested groups, 312
options for creating, 312
orientation, 315
page breaks, 313
printable area, 313
read-only, 309
Report Wizard, 312
rules design, 311
sections of, 309–312
group, 311–312
Page, 311
simple, 312–317
Page Setup tab, 313–315
report layout, 315–317
Requery action, 450
Requery method, 217
Request feature, 91–96
adding data macro, 94–95
CancelOldRequests query, 93–94
modifying Request table, 91–93
running query, 95–96
request items, handling, 68–71
Request record, 67
Request tables, 49–51
InventoryItemID field, 51
modifying, 91–93
Status field, 49–50
required fields, 26–27
Required property, 23
Restrict Delete option, 36, 41, 47
Result Type property, 38
results, sorting, 85
results tables, 472
Ribbon and Toolbar Options section, 265
ribbon element, 263
Ribbon Name combo box, 282
RibbonName field, 265, 280
ribbons, 268–283
Arrange tab, 192
Blank Form button, 198
Controls section, 198
Create tab, 191
custom, 276–283
autoexec macro, 281–282
generating XML, 278–280
implementing callback methods, 280–281
Library ribbon, 282–283
mnuCommands query, 276–278
Form button, 191, 208
Insert Right button, 192, 198, 209
Menu forms, 270–272
Menu tables, populating, 272–275
MSO images, 275–276
navigation, 262–268
displaying system objects, 267–268
sample ribbon tab, 263–267
None link, 210
Remove Layout button, 192–193
Remove layout button, 208
Show Table button, 207
Stacked button, 198, 200
Subform button, 200–201, 212
Tab Control button, 199
tables, 269–270
View Code button, 203, 214–215

www.it-ebooks.info

RibbonXML field, 265, 280
Row and Column fields, switching, 119
Run button, 80
Run from My Computer link option, 403
RunCode action, 281, 468
RunCommand method, 215, 219
RunDataMacro action, 449, 461, 468
RunMacro action, 260, 559
/runtime command line parameter, 393, 396

S

sandbox mode, configuring, 547–548
SandBoxMode value, 547
Save & Publish tab, 287
Save As dialog box, 21
Save button, record selector, 197
Save Current Theme dialog box, 297
Save Database As option, 287
Save parameter, 162
scheduled tasks, 353–356
screentip attribute, 263
ScreenTip field, 283
Scripting Runtime, 522
Scroll Bars property, 156
Search button, 198
Search Caption, 238
Search dialog, invoking, 169–172
search features, 450–451
search fields, 23, 157–159
Search logic, 159–162
Search page, basic, 233–238
Search_Click method, 160, 170, 204, 246
second normal form (2NF), 18
security, 531–550
  configuring sandbox mode, 547–548
  databases
    encrypting, 548–550
    signing, 540–547
  trusted documents, 531–539
    configuring, 538–539
    disabled content, 531–535
    trusted locations, 536–538
Select Layout button, 158
select queries, 77–87
  adding tables, 80–81
  creating, 77–80
  making changes to, 84–86
    calculated columns, 85
    sorting results, 85
    using SQL view, 86
  using joins, 81–84
    adding columns to queries, 83–84
    Join Properties dialog box, 82–83
    multiplicity, 82
  using queries as views, 87
Select Row button, 159
Selected Fields list, 39, 48, 99
SendEmail action, 460
SendOverdueEmails function, 460–461, 466
SendOverdueEmails macro, 462–463, 468
Sequence fields, 277
SetField actions, 56, 59, 69, 448
SetFilter actions, 450–451
SetFocus method, 215
SetWarnings action, 248
Shared Documents library, 490
SharePoint, Microsoft. *See* Microsoft SharePoint
SharePoint server, 554
SharePoint site details, 423
"Show add-in user interface errors" check box, 266
Show check box, 84
Show Date Picker property, 48, 91
Show Signature Detail link, 544
Show Table button, 80, 207
Show Table dialog box, 80, 92–93, 502
signing databases, 540–547
  certificates, 540–541
  installing, 543–547
  packaging, 541–542
Simple Query Wizard option, 78
simulating Access Runtime, 393–396
Site Actions dropdown, 425
Site Permissions link, 424
size attribute, 263
Size Mode property, 304
slicing, 103–104
Sort property, Descending, 208
sorting
  configuring, 329–331
  records, in forms, 132–133
  with Report Wizard function, 322–323
  results, 85
Sorting and Grouping link, Design View, 329
Source Object property, Property Sheet, 212, 218
Split Forms, 133–137
  creating, 133–135
  modifying fields in, 135–137
Split Horizontally button, 201

www.it-ebooks.info

SQL (Structured Query Language) view, 86
SQL Azure database, 384, 389
SQL Azure Server Administration page, 379
SQL Azure, upsizing with, 375–390
  creating user account, 380–381
  installing SSMA, 375–378
  linking views in, 389–390
  migrating data to, 381–387
  setting up database server, 378–380
  viewing results of, 387–389
SQL Server Migration Assistant (SSMA), installing, 375–378
SQL Server Reporting Services (SSRS), 104
SQL Server, upsizing with, 365–375
  adjusting OpenRecordset code, 374–375
  using Upsizing Wizard, 365–372
  viewing results of, 372–374
SQL View button, 86
SSMA (SQL Server Migration Assistant), installing, 375–378
SSMA project, 382
SSMA welcome dialog box, 381
SSRS (SQL Server Reporting Services), 104
Stacked button, 146, 158, 163, 198, 200
Standard color, 301
startFromScratch attribute, 263
State table, 504
Static folder, 228, 303
Status ComboBox control, 141
Status controls, 141, 175, 443
Status field, 41, 49–50, 141, 443, 495
Status from, 141
storing
  bitmap images, 222
  Checkout form user name, 519–520
  images in file system, 223
  images in Microsoft SharePoint, 444–446
  OLE objects, 222–223
Structured Query Language (SQL) view, 86
subform, 338
Subform button, 167, 200–201, 212, 218
Subform control, 200, 213
subform data properties, 168
SubForm Wizard, 167, 271
Subform Wizard dialog box, 200–201
subforms in forms, adding, 150
subreport, 338
subtotal sunbound controls, 334–335
sum function, 106
Summary Option dialog box, 322
supertip attribute, 263
SuperTip field, 283
support, 221–223
Survey table, 472, 474, 483, 486–487
Switch Row/Column button, 119
switching fields, 119
syntax errors, 266
SysCmd method, 171
system objects, displaying, 267–268

T

Tab Control button, 199, 256
Tab Control object, 300
tab controls, 199–200
Tab key, 27
TabControl, 199–200, 220
Table Design button, 26, 28, 32, 265
Table tab, 53
Table1 table, 492
tables, 17–52, 269–270
  adding to select queries, 80–81
  designing, 19–51
    Category tables, 26–27
    Customer tables, 21–26
    Item tables, 32–38
    ItemInventory tables, 39–45
    Loan tables, 45–49
    Media tables, 28–31
    practices, 17–19
    Request tables, 49–51
  items, adding data-bound images to, 223–224
  linked
    Microsoft Outlook, 498–503
    Microsoft SharePoint, 492–496
  viewing relationships, 51–52
Tables/Queries dropdown list, 78
tabs, sample ribbon, 263–267
Tabular button, 156
Tabular layout, 138, 143–144
tag attribute, 263
Target column, 281
TargetType column, 281
tasks, scheduled, 353–356
Tasks list, 490–491, 494
Tasks.Task Group column, 495
Team SharePoint site, 490–491
Team site, 493
template customization
  barcode support, 349
  maximum lines, 347

www.it-ebooks.info

measurements, 346
name, 348
sort order, 348
templates, installing Northwind web database template, 551–555
TempVars collection, 517, 519
TempVars variables, 517–520
storing CheckOut form user name, 519–520
UserName, 517–519
testing
applications, 75–76
CheckOut form, 187–190
Customer Profile page, 205–206
Items on Loan page, 217
ItemSearch form, 241
Text Align property, 194, 198, 210
Text Box button, 157
Text option, 258
Text/Background color, 294
TextBox control, 157, 198, 204
Theme color, 295, 301
Theme file, 297
themes, Microsoft Office, 291–303
applying, 300–302
creating, 297–300
custom color, 292–296
font scheme, 296–297
visual adjustments to, 302–303
Themes button, 292
third normal form (3NF), 18
time, displaying, 333–334
timers, 511–515
digital clock, 512–513
simple animation, 513–515
Title control, 149, 209
Title field, 32, 149, 175, 448
total sunbound controls, 334–335
Transact-SQL syntax, 86
Trust Center dialog box, 396, 536, 543
trusted documents, 531–539
configuring, 538–539
disabled content, 531–535
trusted locations, 536–538
Trusted Documents tab, 539
Trusted Location tab, 537
trusted locations, 536–538
txtAuthor control, 239, 242–243
txtAuthor field, 244
txtCustomerID control, 168, 203
txtCustomerID_LostFocus handler, 204–205
txtCustomerID_LostFocus method, 168
txtDescription control, 244
txtEmail control, 158
txtFirstName control, 158
txtInventoryItemID control, 178–179, 186
txtInventoryItemID LostFocus event, 180, 184
txtKeywords control, 234, 241
txtKeywords property, 233
txtLastName control, 158
txtPicture control, 224, 227, 239, 248, 253
txtPostal control, 158
txtSelectedItemID control, 243, 246
txtSelectedItemID property, 233
txtTitle control, 244
Typical installation option, 412

U

UIs (user interfaces)
defining, 3
elements, 291
macros, 461
unbound controls, 332–335
displaying current date and time, 333–334
fixing percentage formula, 335
formatting page numbers, 332–333
totals and subtotals, 334–335
unique constraints, 27
Update button, 92
Update link, 305
update query, 92
Updated( ) function, 63
Upload Multiple Files link, 445
upsizing, 361–391
with Access, 363–365
and data macros, 390–391
overview, 361–362
with SQL Azure, 375–390
creating user account, 380–381
installing SSMA, 375–378
linking views in, 389–390
migrating data to, 381–387
setting up database server, 378–380
viewing results of, 387–389
with SQL Server, 365–375
adjusting OpenRecordset code, 374–375
using Upsizing Wizard, 365–372
viewing results of, 372–374
using linked tables, 361–362
Upsizing Wizard, 366, 371
URL control, 224

www.it-ebooks.info

URL field, 224
Use As Form And Report Icon option, 284
Use Control Wizards icon, 257
Use Control Wizards link, 167, 198, 257
user account, for SQL Azure, 380–381
user experience, 255–289
  locking database, 283–289
    compiling, 287–289
    removing navigation, 284–287
  navigation
    forms, 255–262
    ribbons, 262–268
  ribbons, 268–283
    custom, 276–283
    Menu forms, 270–272
    Menu tables, 272–275
    MSO images, 275–276
    tables, 269–270
user interfaces. *See* UIs
user names, for CheckOut form, 519–520
user-executed macros, 73
UserInfo list, 494
UserInfo tables, 495
UserName field, 518–520
UserName TempVar variable, 517–519
USysRibbons table, 265, 267–268, 280
Utilities.ConfirmLogin macro, 559, 561
Utilities.ConfirmLogin submacro, 560

V

validation, fields, 29–30
Validation Rule property, 29
Validation Text property, 29
values
  default, 30–31
  multiple, 112–114
VBA (Visual Basic for Applications), 463–470
  adding code, 240–241
  implementing VBA code, 465–468
  OverdueItemDetails query, 463–465
  restoring code, 428–437
VBA editor, 160, 162, 203, 214, 218, 351
View button, 86, 105
View Code button, ribbon, 203, 214–215
View link, PivotTable, 104
viewing color schemes, 293
views
  for forms, 130–132
  using queries as, 87
Visible flag, 209
Visible property, 193, 202, 521
Visual Basic for Applications. *See* VBA

W

web
  databases, 431–433
  forms, publishing, 434–441
Web Browser control, 447
  adding to Item form, 249–251
  Item pictures and, 444
  passing HTML content to, 522–524
web databases, Northwind. *See* Northwind web database
webBrowser control, 250, 252
webCategory form, 437, 441
webCustomer, 438
webInventoryItem, 444
webItem form, 444, 449, 451
webNavigation form, 440–441
Where Condition, 60
wildcard character, 160
Windows Security dialog, 542
Windows Task Scheduler, 350, 353

X, Y, Z

X button, 61
XML (Extensible Markup Language)
  generating for custom ribbons, 278–280
  importing files, 503–509
    exchange rates, 505–509
    process of, 504
.xml file, 298

www.it-ebooks.info

Pro Access 2010 Development



Mark Collins

Apress®

www.it-ebooks.info

Pro Access 2010 Development

Copyright © 2011 by Mark Collins

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-4302-3578-1

ISBN-13 (electronic): 978-1-4302-3579-8

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

President and Publisher: Paul Manning
Lead Editor: Matthew Moodie
Technical Reviewer: Michael Mayberry
Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Frank Pohlmann, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Coordinating Editor: Corbin Collins
Copy Editor: Tracy Brown
Compositor: Bytheway Publishing Services
Indexer: BIM Indexing & Proofreading Services
Artist: April Milne
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail **orders-ny@springer-sbm.com**, or visit **www.springeronline.com**.

For information on translations, please e-mail **rights@apress.com**, or visit **www.apress.com**.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at **www.apress.com/bulk-sales**.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at **www.apress.com**. You will need to answer questions pertaining to this book in order to successfully download the code.

www.it-ebooks.info

To my beautiful wife, Donna. You're the best thing this side of heaven.

www.it-ebooks.info

Contents

■ **About the Author** xix

■ **About the Technical Reviewer** xx

■ **Acknowledgments** xxi

■ **Preface** xxii

PART 1 ■ ■ ■ Introduction 1

■ **Chapter 1: Introduction** 3

What Is Access? 3

Defining the User Interface 3

Choosing the Application Architecture 3

Using This Book 4

The Sample Application 4

Getting Started 13

PART 2 ■ ■ ■ Defining the Database Schema 15

■ **Chapter 2: Defining and Relating Tables** 17

Considering Design Practices 17

Using Primary Keys 17

Normalizing a Database 18

Database Constraints 19

Designing the Tables 19

Creating the Customer Table 21

Creating the Category Table 26

www.it-ebooks.info

Creating the Media Table 28
Creating the Item Table 32
Creating the ItemInventory Table 39
Creating the Loan Table 45
Creating the Request Table 49
Viewing the Relationships 51
Summary 51
Chapter 3: Using Data Macros 53
Understanding Data Macros 53
Identifying Data Macro Limitations 54
Creating Your First Data Macro 54
Testing Your Macro 57
Exploring the Macro Editor 58
Understanding Data Blocks 58
Using Aliases 59
Using Data Actions 60
Navigating the Macro Editor 61
Implementing the Loan Before Change Event 61
Making Sure the Item is Available 61
Calculating the Due Date 63
Calculating the Late Fee 63
Validating Renewals 65
Adding the Current Loan Reference 66
Adding the Lookup Field 66
Modifying the Loan After Insert Event 66
Modifying the Loan After Update Event 68
Handling Requested Items 68
Implementing the Before Change Event 69

www.it-ebooks.info

Creating a Named Data Macro 69
Calling a Named Macro 71
Computing Overdue Fees 71
Creating a Named Data Macro to Compute Overdue Fees 71
Creating a User-Executed Macro 73
Debugging Data Macros 73
Using the LogEvent Action 74
Viewing the Application Log 75
Testing the Application 75
Summary 75
Chapter 4: Designing Queries 77
Creating Select Queries 77
Creating a Simple Query 77
Adding Tables 80
Using Joins 81
Making Additional Changes 84
Using Queries as Views 87
Creating Action Queries 87
Creating the AddInventoryItem Query 87
Enhancing the Request Feature 91
Creating a Crosstab Query 96
Building the AllLoans Query 96
Designing the LoanSummary Query 98
Summary 102
Chapter 5: Creating PivotTables 103
Slicing, Dicing, and Drilling 103
Creating a PivotTable View 104

www.it-ebooks.info

Understanding the PivotTable Layout 105
Using the PivotTable Fields 106
Defining the Column Field 108
Using Date Fields 109
Adding a Filter Field 111
Refreshing the PivotTable 112
Adding Other Fields 112
Including Multiple Values 112
Using Calculated Columns 114
Adding Additional Column Fields 115
Creating a Field Hierarchy 116
Using the PivotChart View 118
Configuring a PivotChart View 118
Changing the Chart Type 120
Exporting a PivotTable View to Excel 122
Summary 123
PART 3 ■ ■ ■ Creating Forms and Reports 125
■ Chapter 6: Standard Forms 127
Creating a Single Form 127
Using the Form Wizard 127
Using the Available Views 130
Sorting the Records 132
Using Split Forms 133
Generating the Media Form 133
Modifying the Form Fields 135
Using Continuous Forms 137
Generating the InventoryItem Form 137

www.it-ebooks.info

Designing the InventoryItem Form 140
Modifying the Form Fields 141
Understanding the Layout Options 143
Using the Tabular Layout Option 143
Using the Datasheet Layout Option 144
Using the Columnar Layout Option 144
Using the Justified Layout Option 145
Using a Layout 146
Creating the Item Form 147
Using the Standard Form Template 147
Arranging the Form's Layout 149
Adding a Subform 150
Summary 152
Chapter 7: Creating a CheckOut Form 153
Implementing a Customer Search Feature 153
Populating the Customer Table 154
Creating the CustomerSearch Dialog Box 154
Building a CustomerDisplay Form 163
Creating the CheckOut Form 166
Providing a CheckOut Feature 173
Building an InventoryItemLookup Form 173
Linking the InventoryItemLookup Subform 178
Designing the CheckOut Details 180
Implementing the CheckOut Logic 184
Testing the Application 186
Summary 189

www.it-ebooks.info

Chapter 8: Creating a Customer Admin Form 191

Building the Customer Profile Tab 191

Creating a CustomerUpdate Form 191

Configuring the Form Controls 195

Creating the CustomerAdmin Form 197

Testing the Profile Page 205

Building the Items on Loan Tab 206

Enhancing the LoanDetails Query 207

Designing a CustomerLoan Form 208

Configuring the Datasheet View 211

Designing the Items on Loan Page 212

Connecting the Pieces 214

Testing the Page 217

Building the Loan History Tab 218

Summary 220

Chapter 9: Enhancing Product Administration 221

Using Data-Bound Images 221

Image Support in Access 221

Adding a Picture to the Item Table 223

Modifying the Item Form 224

Displaying an Image 226

Implementing Item Search 229

Importing Item Data 230

Designing the Search Form 234

Testing the Search Function 242

Using Conditional Formatting 243

Invoking the ItemSearch Form 246

www.it-ebooks.info

Enhancing the Item Form....248
Adding an Inventory Item....248
Adding a Web Browser....250
Using Page Breaks....252
Modifying the CustomerLoan Form....254
Summary....255
Chapter 10: Enhancing the User Experience....255
Form Navigation....255
Creating the Menu Form....256
Auto-Loading the Menu Form....262
Ribbon Navigation....262
Implementing a Sample Ribbon Tab....263
Displaying the System Objects....267
Building a Custom Ribbon....268
Designing the Tables....269
Creating the Menu Forms....270
Populating the Menu Tables....272
Using MSO Images....275
Implementing the Custom Ribbon....276
Locking Down the Database....283
Removing Navigation....284
Compiling the Database....287
Summary....289
Chapter 11: Branding with Themes and Styles....291
Using Office Themes....291
Understanding Office Themes....291
Applying an Office Theme....301

www.it-ebooks.info

Making Other Visual Adjustments 303
Using Graphics 304
Adding a Banner Graphic 304
Using a Background Image 306
Summary 308
Chapter 12: Reports 309
Exploring Access Reports 309
Understanding Report Sections 309
Options for Creating Reports 312
Creating a Simple Report 312
Modifying the Page Setup 313
Modifying the Report Layout 315
Creating the AllLoans Report 317
Using the Report Wizard 317
Exploring the Design View 325
Using the Layout View 327
Configuring Grouping and Sorting 329
Using Unbound Controls 331
Creating a CheckOut Report 335
Adding the Data-Bound Controls 336
Formatting the Report Header 337
Adding a Subreport 337
Modifying the CheckOut Form 341
Generating InventoryItem Labels 343
Creating an InventoryItemDetail Query 343
Using the Label Wizard 345
Modifying the Color Scheme 349

www.it-ebooks.info

Auto-Generating a DailyLoans Report. 350
Creating the DailyLoans Report. 350
Creating a DailyReport Macro. 351
Creating a Scheduled Task. 353
Modifying the Daily Report Macro. 356
Summary 357
PART 4 ■ ■ ■ Multiuser Considerations 359
■ Chapter 13: Upsizing 361
Understanding Upsizing 361
Using Linked Tables. 361
Project Overview. 362
Upsizing with Access 363
Upsizing with SQL Server 365
Using the Upsizing Wizard. 365
Viewing the Upsizing Results. 372
Adjusting the OpenRecordset Code. 374
Upsizing with SQL Azure 375
Installing SQL Server Migration Assistant. 375
Configuring a SQL Azure Database. 378
Migrating the Data to SQL Azure. 381
Viewing the Migration Results. 387
Linking the Views. 389
Data Macros with Linked Tables 390
Data Macros in a Split Access Database. 391
Data Macros in Linked SQL Server Tables. 391
Summary 391

www.it-ebooks.info

Chapter 14: Distributing the Application 393

Using the Access Runtime 393

Simulating the Access Runtime 393

Understanding the Filename Extensions 396

Downloading the Access Runtime 397

Modifying the Application 397

Supporting Images in a Distributed Environment 398

Modifying the Application 399

Relinking the Linked Tables 400

Creating an Executable File 401

Creating an Installation Package 402

Installing the Package Solution Wizard 402

Using the Package Solution Wizard 404

Deploying the Installation File 410

Summary 413

Chapter 15: Publishing to the Web 415

Preparing to Publish 416

Updating the Data Macros 416

Using the Compatibility Checker 421

Publishing the Access Database 422

Publishing to Access Services 422

Exploring the SharePoint Site 424

Exploring the Access Database 427

Restoring the VBA Code 428

Fixing the Default Value 429

Using Web Databases 431

www.it-ebooks.info

Creating Web Forms 434
Creating a Category Web Form 435
Creating a Media Web Form 437
Creating a Customer List Web Form 438
Creating a Navigation Form 439
Setting the Default Web Form 440
Adding an Item Web Form 442
Creating the Initial Form 442
Creating the InventoryItem Subform 443
Adding the Picture and Web Browser 444
Adding an InventoryItem Record 448
Providing a Search Feature 450
Modifying the Navigation Form 451
Summary 452
PART 5 ■ ■ ■ Advanced Topics 455
■ Chapter 16: Integrating Outlook 457
Sending E-mails 457
Creating the OverdueItems Query 457
Creating a Data Macro 460
Adding a UI Macro 461
Sending the E-mails 462
Sending E-mails with VBA 463
Creating the OverdueItemDetails Query 463
Implementing the VBA Code 465
Sending the E-mails 468
Using Data Collection 470
Understanding Data Collection 470
Defining a Data Collection E-mail 471

www.it-ebooks.info

Processing the Replies 483
Resending the E-mail 486
Summary 487
Chapter 17: Using External Data 489
Linking and Importing 489
Linking a SharePoint List 489
Creating a Team SharePoint Site 490
Creating a Linked Table 492
Using the Linked Tables 495
Linking an Outlook Folder 497
Modifying the Data File Name 498
Creating the Linked Table 499
Using the Linked Table 502
Importing an XML File 504
Understanding the XML Import Process 505
Importing the Exchange Rates 506
Summary 510
Chapter 18: Miscellaneous Features 511
Using Timers 511
Adding a Digital Clock 512
Creating Simple Animation 513
Calling the Windows API 515
Using TempVars 517
Creating the UserName TempVar 517
Storing the CheckOut User 519
Displaying HTML Content 520
Modifying the Item Form 521

www.it-ebooks.info

Passing HTML Content to a Web Browser Control 522
Adding a Chart 524
Summary 529
Chapter 19: Security 531
Using Trusted Documents 531
Understanding Disabled Content 531
Trusting a Document 535
Signing a Database 540
Creating a Certificate 540
Packaging an Access Database 541
Installing a Signed Database 543
Configuring Sandbox Mode 547
Encrypting an Access Database 548
Summary 550
Appendix A: Northwind Web Database 551
Installing the Northwind Web Template 551
Downloading the Template 551
Publishing the Database 553
Understanding the Sample Database 555
Setting Up Your Employee Record 555
Exploring the Access Client Application 557
Index 563

www.it-ebooks.info

About the Author



Mark Collins has been developing software solutions for more than 30 years. His wrote his first Access application in 1994 and has witnessed the tremendous improvement in developer tools over the years. Some of the other key technology areas that highlight his career include COM, .Net, and SQL Server. He currently supports a large non-profit organization, serving as data architect and back office automation and integration specialist.

You can see more info on his web site (`www.TheCreativePeople.com`). If you have questions or comments, contact Mark at `markc@thecreativepeople.com`.

www.it-ebooks.info

About the Technical Reviewer

Michael Mayberry is a systems architect who concentrates on web services and BI solutions. He leads his team in adopting new technologies and incorporating them into enterprise applications within the non-profit industry. He has extensive experience in developing data software ranging from small-scale applications to large-scale data entry and BI solutions. Data collection and analysis using SQL Server have been his primary focus for over 10 years.

www.it-ebooks.info

Acknowledgments

I want to say a very big thank you to my beautiful wife, Donna. I can honestly say that I would not be who I am if it were not for what you have sown into my life. I am truly blessed to be able to share my life with you. Thank you for your loving support and for making life fun!

Next, I'd like to thank all the people at Apress who made this book possible and for all their hard work that turned it into the finished product you see now. Everyone at Apress has made writing this book a pleasure.

Also, I want to thank Michael Mayberry for checking the content for accuracy. A second set of eyes from another very capable developer is so important for a work like this. Thank you for being that for me.

I would like to say a special thank you to Corbin Collins who kept this project on schedule despite several unexpected delays. You were part coach, part administrator, and part cheerleader; which was just what we needed.

Finally, I want to thank Matthew Moodie who help shaped the initial drafts into the final text. I really appreciated your challenging me to fine tune the details to make this book even better.

www.it-ebooks.info

Preface

Access 2010 can be used to easily create some pretty feature-rich solutions. Its scope, however, is extremely broad. Non-technical power-users can create fairly sophisticated applications, and developers can do even more with it. Because the 2010 release provides integration with SharePoint, you can even use Access to build SharePoint solutions.

When I first started writing this book, the first challenge I had was to define the target audience, because Access users range from completely non-technical to senior developers. While some of the chapters are fairly technical in nature, this book is designed to be accessible for novice power users to intermediate developers. This book does not require any prior knowledge of Access.

The next challenge was how to best present such a vast amount of information. My goal in writing this book is to really teach you Access; not just give you a lot of facts. My intent is to cover the core concepts and cover them well, rather than detail an exhaustive list of features. I have always found the mentor-apprentice model to be very effective. So with that in mind, I decided to build a solution using Access 2010 and invite you to work alongside me. As we work together, I'll answer the whys, as well as the hows.

Each chapter builds upon the previous one and explains a core competence that Access developers will need to master. We'll start with by defining a data model, and then write data macros to implement business rules. Along the way, you'll add forms and reports, with each chapter introducing new concepts. The later chapters will cover more advanced topics such as distributing the application, publishing to SharePoint and integrating external data sources.

I recommend that you work through each chapter in sequence. However, I have saved a copy of my database at the end of each chapter, which you can download from www.apress.com. So if you want to skip to Chapter 7, for example, you can download the database from Chapter 6 and use that to work through the exercises in Chapter 7.

In each chapter, I provide detailed set-by-step instructions so you can follow along and implement each exercise yourself. If you prefer, you can also download the completed chapter and refer to the implementation as you read the chapter.

www.it-ebooks.info