

THE EXPERT'S VOICE® IN ORACLE

Beginning Oracle Application Express 4

*Your ticket to easy and robust web-application development
using Oracle's powerful toolset for power-users, programmers,
and database administrators*

Doug Gault, Karen Cannell,
Patrick Cimolini, Martin Giffy D'Souza,
and Timothy St. Hilaire

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.



Contents at a Glance

Contents	v
About the Authors	xv
Acknowledgments	xvii
■ Chapter 1: An Introduction to APEX 4	1
■ Chapter 2: A Developer's Overview	7
■ Chapter 3: Identifying the Problem & Designing the Solution.....	31
■ Chapter 4: SQL Workshop.....	39
■ Chapter 5: Application and Navigation.....	61
■ Chapter 6: Forms and Reports – The Basics	99
■ Chapter 7: Forms and Reports – Advanced.....	155
■ Chapter 8: Programmatic Elements	207
■ Chapter 9: Security	245
■ Chapter 10: Application Deployment.....	273
■ Chapter 11: Understanding Websheets	281
■ Chapter 12: A Worksheet Example	313
■ Chapter 13: Extended Developer Tools.....	333
■ Chapter 14: Managing Workspaces	355
■ Chapter 15: Team Development	375
Index.....	405



An Introduction to APEX 4

Welcome to the wonderful world of Oracle Application Express (APEX). You're about to learn how to use a tool that we believe will revolutionize the way you think about and approach writing web-based Oracle systems. It certainly has for us.

Prior to the advent of APEX, developing fully interactive web based systems over data that resided within an Oracle database almost always meant learning a new and often complex language like Java, .NET, or PHP and then figuring out how to integrate your chosen language seamlessly with the data that resided in your database. Often this also meant trying to incorporate business rules that were already coded in the form of PL/SQL.

In such situations, it could take months or even years just to become proficient enough with your chosen language to begin to write a functional system. If you're like many, you'll become frustrated with the fact that you've spend an inordinate amount of time to do what seems like a relatively easy task.

Fear not! The days of long-winded and complex web development platforms may just be behind us.

What is APEX?

APEX is a 100% browser-based rapid application development (RAD) tool that helps you to create rich interactive Oracle-based web applications very quickly and with relatively little programming effort.

There are many RAD development tools and platforms on the market. If you're dealing with data that resides in an Oracle database, there are a number of things that makes APEX distinctive and thus more attractive as a development platform. First and foremost is the fact that APEX is built on and uses as its core languages, SQL, and PL/SQL. This is a huge advantage for those of you who have already been working with the Oracle database because it means you can immediately draw on what you know. Even for those who don't have an Oracle background, if you are going to be working with an Oracle database, you'll need to learn about its particular flavor of SQL and will at some point likely find a need for the PL/SQL procedural language.

This fact becomes even more beneficial if you are migrating an Oracle-based system to APEX that already has a significant amount of business logic coded into stored PL/SQL program units. In this instance, you can almost immediately take advantage of that logic with very little effort or change to the existing code.

Another great advantage is that APEX is a declarative tool that provides a feature rich core designed to make your job easier. Because APEX takes care of many of the underlying functions common to all web-based applications you are able to focus on the logic specific to your application.

A large share of what you need to accomplish can be done by using one of the many built-in wizards provided as part of the APEX Application Builder. The wizards walk you through the process of defining what you want your application to do and then stores that information as metadata. Once the wizard is complete you can edit and enhance the functionality, or even replace it with your own custom SQL and PL/SQL routines. After you become proficient with APEX you might even find yourself bypassing the wizards altogether and generating more complex definitions directly.

During the course of the book, you will discover that you will likely want a few other tools at your disposal, but in truth, you could easily develop a very rich application using nothing but your web browser and what APEX provides for you.

A Brief History of APEX

APEX has been around for quite some time—perhaps even longer than most people know. The first public release of APEX, or HTML DB as it was called then, came in 2004, but its history reaches back quite a long way.

Ancient History

APEX has its roots in technology that has been around for quite some time. In fact, parts of the PL/SQL Web Toolkit, which is used under the covers by APEX to generate the HTML that is sent to the browser, date back to as early as 1994.

At that point in time, you *could* actually write web applications in PL/SQL by hand, and unfortunately we did. This required not only a thorough knowledge of PL/SQL and HTML but also the patience of a saint and the determination of a headstrong mule. The end result wasn't very pretty, and it was definitely not secure by today's terms, but it was functional, if somewhat limited.

Not long after, Oracle introduced PL/SQL Server Pages (PSPs). This involved first coding the static HTML and including special Oracle markup to indicate where dynamic data would go. Once you had the output looking as you wanted, you then ran it through a program called LOADPSP. This would translate the raw HTML and the special Oracle markup into a PL/SQL procedure that, again, used the PL/SQL Web Toolkit to emit the HTML including the dynamic data you requested. At the time, this was a huge leap forward. I worked at a company where we actually built an entire framework around using PSP technology and deployed it at several clients.

Finally, in 1997, WebDB came on the scene. The true grandfather of what we now call APEX, WebDB was revolutionary in that it was a 100% web-based tool that allowed developers to design web applications. It was written entirely in PL/SQL even though Java seemed to be taking over the world. Developers could point WebDB at their database and generate code that would produce forms, reports, charts, and calendars. There was no session state management or templates; once the code was generated, there was no going back through the tool.

WebDB allowed a large number of companies who wanted to jump on the web-based bandwagon to do so without spending vast amounts of time and effort in retraining their staff. As a tribute to its success, I know of a number of companies that still have WebDB system running in production environment.

Unfortunately WebDB's days were numbered. Because it generated code (and if you didn't like the code it generated, then too bad for you), it had already begun to fade from favor by the time it was absorbed into Oracle's Portal product. However, creator Mike Hichwa did not forget the glimpse of greatness that WebDB had seen.

More Recent History

Around 1999, Oracle's CEO, Larry Ellison presented Mike Hichwa (VP of Software Development) with the task of creating an internal calendaring and scheduling system for Oracle Corp. The original remit was to use WebDB to generate the initial code and then hand code all the changes from that point forward. Mike, however, saw this as an opportunity to completely re-write WebDB into something that could be far more useful. Thus, with the help of Joel Kallman and Tom Kyte, Oracle Flows was born.

Based on the success of the internal Calendaring and Scheduling system, the team was allowed to move forward towards making Oracle Flows a product. In 2001, using what then known as Flow Builder,

Mike and team begin implementing systems for various customers, including one situation where they managed to replace a Java development project that was going horribly wrong.

By 2003, the team had proven the tool's power and they were given permission to release it as a product. HTMLDB 1.5 was released to the public as a no-cost option of Oracle 10gR1.

Since then, various releases have been introduced; each providing improved features and functionality. The following is a very brief list of the releases and some of the more notable features:

- HTMLDB 1.6 (2004) introduced themes, master-detail forms, page groups, page locking, and some multilingual capabilities.
- HTMLDB 2.0 (2005) introduced SQL Workshop, graphical query builder, database object browser, and session state protection.
- APEX 2.2 (2006) introduced packaged applications, the APEX dictionary views, and the access control wizard.
- APEX 3.0 (2007) introduced PDF printing with BI Publisher, migration from Microsoft Access, and page and region caching.
- APEX 3.1 (2008) introduced Interactive Reports, runtime-only installation capability, and improved security.
- APEX 3.2 (2009) introduced a migration helper for Oracle Forms based systems and various security enhancements.

APEX 4 and the Future

And so we arrive at the present day with the release of APEX 4.0. In my opinion, this release of APEX has truly brought the development environment into the realm of “forces to be reckoned with.” The entire focus of APEX 4 was to make development of rich interactive Web 2.0 applications easier by making the process as declarative as possible. APEX 4 has introduced so many new features—indeed, new ways to attack problems—that it will be hard *not* to choose APEX as the preferred development platform for Oracle-based applications.

APEX 4.0's Dynamic Actions provide a way for you to define client side behaviors, such as enabling or disabling fields or regions declaratively without JavaScript. With some JavaScript knowledge under your belt, you can create complex dynamic actions that do client side calculations, AJAX, and more.

An improved Charting Engine based on AnyChart 5.1 provides declarative Flash-based charts, gauges, maps, and Gantt charts. All chart types are interactive and drillable, and several charts can be combined into a dashboard style interface.

Probably the most exciting new feature is the new plug-in architecture that provides an extensible framework allowing APEX Community members to build and share their own custom item, region, process, and dynamic action types. While the ramifications of this might not be immediately apparent, the possibilities of what can and will be developed using the plug-in architecture are virtually limitless—and that is very good news for *all* APEX developers.

As users of the APEX development platform, we no longer have to wait for the APEX Team to respond to specific feature requests. We can now take the future of APEX into our own hands and code missing features, actions, and item types ourselves. In fact, I see a future where the APEX Team themselves use the plug-in architecture to extend APEX in many different directions.

I almost can't overstate the significance of plugins. While APEX 4 is definitely a giant leap forward from the architecture of APEX 3, the plug-in architecture blows the doors wide open to change from the broad and growing community of APEX developers.

APEX now comes with a Team Development feature that eases the management of the development process by tracking features, to-do lists, bugs, and milestones. A user feedback mechanism is also included that allows users to provide inline feedback while using the system. The feature automatically

captures the user's session state information so you can see exactly what was going on during their session. You can then take this information and create a bug or a to-do entry with the simple click of a button.

Websheets provide a fast and direct way for end users to gather and share information without IT intervention. Armed with only a web browser and access to the Websheets application, end users can define page content, data grids, and reports and decide who else in the enterprise has access to that data. Websheets page content supports standard wiki syntax and pages can be organized hierarchically. Users can also add annotations to pages and content in the form of files, notes, and tags.

What You Need to Get Started

The goal of this book is to get you started using APEX, to launch you in a way that enables you to grow towards mastery of the product. To begin, you need three things: access to an APEX instance, access to a web browser, and a copy of SQL Developer.

Access to an APEX Instance

This is definitely a hands-on book, so to work through the examples and exercises you'll need access to an instance of APEX 4. There are a number of different ways you can access APEX; depending on your level of comfort and expertise with Oracle, some may be better for you than others. Here is a description of the three most common scenarios:

- By far the easiest is to sign up for an account on Oracle's hosted version of APEX at <http://apex.oracle.com>. It's free for non-production applications and is a great place to get started, as you don't have to worry about installing either the database or APEX.
- If you already have an Oracle database installed locally, you can download and install APEX 4 into that instance. Simply go to the Oracle APEX home page at <http://otn.oracle.com/apex> and download the latest version of the software.
- If you don't have an Oracle database already but would like to install one locally, you can download a free developers license from OTN at . Both Oracle 10g and 11g will run APEX 4; however, Oracle 11g will allow you to install APEX as an option in the database install.

While having a locally accessible instance of the Oracle database will give you more direct access to the data, it's definitely not necessary to complete the exercises in this book. All code and instructions have been written so that they can be completed on Oracle's hosted instance with no special access required.

■ **Note** Oracle provides very good documentation on the installation process for both the database and APEX, so it is not covered in detail here. However, if you are planning to install APEX 4 on an environment in your organization, you will want to coordinate with the database administrator responsible for that instance to ensure no mishaps occur.

Web Browser

The APEX documentation states that to view or develop APEX applications, the web browser must support cookies, JavaScript, HTML 4.0, and CSS 1.0. However, although you can deploy to any browser that support these things, the list of supported browsers is fairly narrow. Currently, the following browsers are supported: Internet Explorer 7+, Firefox 3.5+, Apple's Safari, and Google Chrome.

I don't want to get into a religious debate about which web browser is the best on the market, but my preference for development is Firefox. There are a number of free add-ons and plug-ins (Such as FireBug and The Web Developer Toolbar) that will help you with APEX development. Note that because of the difference in the way each browser interprets HTML and JavaScript, you must test your application in any and all web browsers that your target audience might use.

SQL Developer

As mentioned before, all the exercises and scripts in the book can be loaded and run directly within the APEX interface. However, if you have chosen to install or have access to a local instance of the Oracle database, a SQL IDE will definitely make your life easier.

SQL Developer is a free SQL and PL/SQL IDE provided by Oracle. You can download SQL Developer from the Oracle Technology Network's home page at <http://otn.oracle.com/>

Using SQL Developer you can browse database objects, edit row data, develop and test stored PL/SQL program units, code and test SQL statements, and interactively debug PL/SQL code. SQL Developer also has many direct integration points with APEX that make monitoring and maintaining APEX instances and applications easier. We won't cover those in this book, but it's definitely worth your time to look into this tool.

Summary

Oracle Application Express has come a long way from its simple beginnings and we are poised at the beginning of a new cycle of growth for the APEX community. APEX 4 provides so much possibility and promise that it's hard not to be excited about what the future holds. With that spirit, we start you on your journey to discover how APEX can make development easier and more fun.



A Developer's Overview

You're probably anxious to get started, but there are a few concepts that you should understand before you jump into APEX development headfirst. This chapter will introduce the fundamental development architecture of APEX and then will walk you through the different areas of the developer interface.

You'll be delving deeper into the details as you go through the book and put the architecture to work for you, but it will help tremendously to know how things are structured ahead of time. This chapter is designed to ease you in, but it isn't a complete guided tour of every nook and cranny. Be patient; you'll get there.

The Anatomy of a Workspace

APEX was designed from the beginning to be a multi-tenant architecture where many different development environments (called workspaces) could exist within a single APEX instance. For instance apex.oracle.com, Oracle's free hosted instance, holds over 10,000 active workspaces, each of which is a completely separate environment unable to see or interact with any of the other workspaces. You can think of this as Software as a Service (SaaS) or a cloud computing architecture, but basically, it means that each workspace is distinct and segregated from all others.

In simple terms, each workspace represents a virtual private container in which developers create and deploy their APEX applications. The development process takes place within the context of a workspace so it's important to know how a workspace is structured. Figure 2-1 uses database entity relationship diagram parlance to help explain the makeup of the objects within a workspace. I'll explore the details in the following sections.

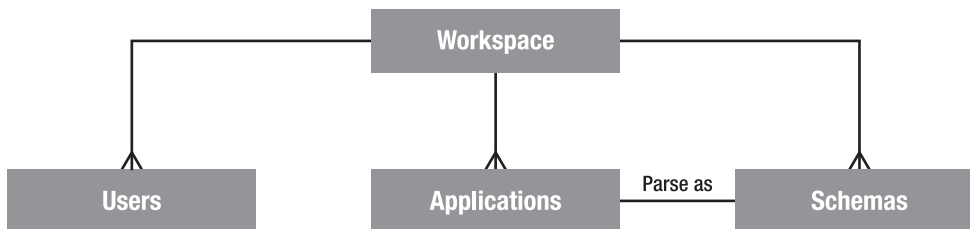


Figure 2-1. Logical makeup of a workspace

APEX Users

To login to an APEX workspace you must have access to a valid APEX user. There are a number of different user roles available that dictate what you are able to do when you log in. The roles are as follows:

Instance Administrators are special users that manage and maintain the overall APEX instance. They are able to set instance level preferences and messages, create and manage workspaces, monitor space utilization, and many other actions related to the overall APEX installation. Instance Administrators are only able to log into the special INTERNAL workspace, which houses the APEX Admin Services application.

Workspace Administrators are responsible for managing the details of a specific workspace and can manage user accounts related to the workspace, monitor workspace activity, view log files, override developer locks and settings, etc. Although it is not good practice, the Workspace Administrator can also act as a developer, creating and modifying applications.

Developers are the users who create and edit the applications within the workspace. They have access to the underlying tables in the schema(s) assigned to the workspace and may create and modify database objects and stored PL/SQL units. Most people writing APEX applications only need this level of access.

End Users are only able to run applications within a workspace. They do not have direct access to any of the underlying database objects, nor do they have access to any of the APEX development modules. End users can't log directly into a workspace.

APEX Users are specific and unique to a workspace, meaning that you can have a user of the same name in multiple workspaces within a single APEX instance, but each of these users is unique. They can have their own password, settings, and are not linked together in any way.

When you're developing, you should get in the habit of logging in as a Developer as opposed to a Workspace Administrator. There are several safeguards available to help keep developers from stepping on each other within a workspace. If you log in as an administrator, these safeguards will be bypassed and you may accidentally interfere with something someone else is working on. While this won't be a problem in a workspace with only one developer, it's still good to get yourself into that habit.

■ **Note** In the context of this book, we'll be using the last three types of user. We're going to assume that APEX has been installed, a workspace has been created, and you have been given the Workspace Administrator's login credentials. If you're using the hosted instance at **apex.oracle.com**, then the username you were given when you signed up will have the credentials of a Workspace Administrator. If, however, you are using a local instance, either refer to the APEX documentation or get your Instance Administrator to help you set up a workspace.

Applications, Pages, Regions, and Items

Although a workspace starts off basically empty, you can have many applications that reside within a workspace. While there is no specific rule, it's likely that all of the applications within a workspace will

share something: they might all use the same underlying database objects, target the same user community, or use the same method for authenticating users.

As you build an application, you'll be adding new pages and building out those pages with regions and items. Figure 2-2 shows the hierarchy between the different types of objects.

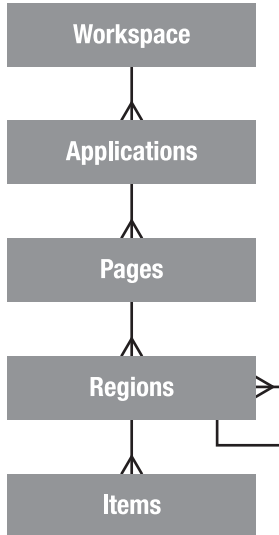


Figure 2-2. General application hierarchy

Applications are basically groups of pages that perform a task (or set of tasks) related to a business function. During the course of this book you'll be building one application within a single workspace, but it's important to know that in a typical development environment, you'll probably be working on many applications across several workspaces.

Pages are the basic building blocks of applications and contain both the user interface components and the programming logic that will process the user's input. We'll cover the rendering of the UI versus the processing of user input later, but for now consider a page roughly equivalent to a screen in desktop UI lingo.

Regions are UI items that serve as content containers. You can have any number of regions on a page, and in APEX 4, regions can be nested within other regions. This gives you the opportunity to create things like dashboards where you might nest a data report region and a graph region within a single parent HTML region.

Items are the HTML form elements that are used to present the UI to the user. These include things such as buttons, select lists, text fields, check boxes, radio groups, etc. There are two categories of items; Page Level Items and Application Level Items; the difference is that the latter is defined at the application level and is not rendered directly on the page. You can think of these as global variables. Page Level Items are defined on a specific page and assigned to a region in order to control where and how they display to the user.

While there is obviously a lot more to an application than these simple building blocks, if you understand the basic hierarchy between these, you'll have a jumpstart when it comes to building your first pages and a solid foundation when it comes to the more intricate tasks.

Workspaces, Applications, and Schemas

While the relationship between workspaces and applications is straightforward, it becomes a bit more complex when you introduce the relationship with database schemas. Figure 2–3 diagrams this relationship.

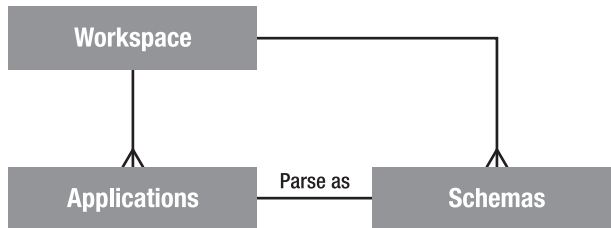


Figure 2–3. How schemas relate to workspaces and applications

When a workspace is created, it is linked with at least one, and possibly many, underlying database schemas. This provides access to database objects such as tables, views, stored PL/SQL program units, etc.

When an application is created, it is assigned a single “parse as” schema from the list of schemas associated with the workspace. A “parse as” schema is the Oracle database user in which all SQL queries and PL/SQL calls run by that application will be executed. So, if your application was defined with a “parse as” schema of DOUG, a query such as:

```
select * from emp
```

would execute in the database as if it were written

```
select * from DOUG.emp
```

Because APEX applications are portable and may not necessarily be run in the same schema they were developed in, it’s not good practice to hard code the schema names into your SQL or PL/SQL. Instead, APEX provides a replacement variable (one of many that you’ll be introduced to throughout the course of this book) for the “parse as” schema. The `#OWNER#` replacement variable will be substituted with the actual “parse as” schema for the application at run time. So the following statement

```
select * from #OWNER#.emp
```

would resolve to

```
select * from DOUG.emp
```

In the most common implementations a workspace will be created and associated with a single underlying database schema. The applications developed in that workspace will have their “parse as” schema set to the only schema associated with the workspace and will utilize the database objects belonging to that schema.

Where a workspace has more than one schema assigned to it, things can become a little more complex. You might be tempted to think that if you associate three schemas with a workspace, any application in that workspace can automatically access the data in all three schemas. However, you would be mistaken.

Because an application is assigned one—and only one—“parse as” schema, all SQL statements and PL/SQL calls will be executed as that schema. It doesn’t matter that the workspace is associated with a given schema; the application itself is not. If you want to access data within a schema other than the

application's "parse as" schema, you must make sure the correct database level grants are in place, just as you would when using any other Oracle tool or development environment.

Take the example shown in Figure 2–4 where two tables you wish to join together as part of a SQL statement are owned by two separate schemas.

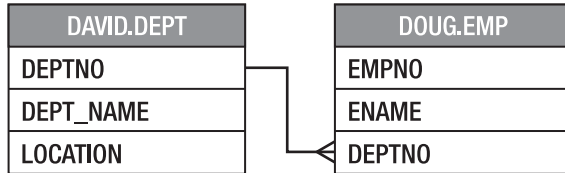


Figure 2–4. Tables joined across schemas

If your "parse as" schema is DOUG, then you must be specifically granted privileges on the objects within the DAVID schema to be able to access it. To do this, you would sign on to the database as DAVID (or as a DBA) and grant the appropriate database privileges on DAVID.DEPT to DOUG.

In this example, if you needed to join the two tables together in a select statement, granting the SELECT privilege on DAVID.DEPT to DOUG would suffice. Then you could write your select statement as follows:

```
select e.empno,
       e.ename,
       d.dept_name,
       d.location
  from #OWNER#.emp e,
       DAVID.dept d
 where e.dpetno = d.deptno
```

The #OWNER# substitution variable would be resolved to your "parse as" schema (DOUG) and the join would work correctly as long as the correct privileges were in place.

■ **Note** Because the grants that allow the select from the DAVID schema are put in place at the database level, it is not necessary to associate the DAVID schema to your workspace. You only need to associate a schema to a workspace if you will be using it as the "parse as" schema for an application within that workspace or need to access the schema objects directly from within the SQL Workshop.

A Final Word on Workspaces

As you have learned, an APEX instance can have many workspaces. But how many workspaces should there be? The answer isn't straightforward.

Unless you are a very small organization with very few apps, you probably shouldn't have only one workspace. On the other hand, you probably shouldn't be creating a new workspace for every new application that you code, either.

There are a couple schools of thoughts on this but I tend to think in terms of application suites. If there are a number of applications that are performing similar tasks against the same underlying data

sets and are aimed at the same target set of users, then they would probably do well in the same workspace.

The key here is to use your judgment and try to keep things easy to develop and maintain. There is nothing worse than logging into a workspace to find you have to page through tens or even hundreds of apps to find the one you want to work on.

A Tour of the APEX Modules

Now that you have a little background on how things are logically architected, it's time to get a closer look at the APEX development environment itself. This section will introduce you to the different sections of the APEX environment and will give you an overview of how things are laid out.

Figure 2–5 shows a hierarchical layout of the APEX menu structure. Later, you'll have a look at each of the main sections and get a glimpse of what's under the covers; this is just an introductory tour. You'll get a much deeper look as we work our way through the development processes.

As you can see, the development environment is broken down into four main sections:

- **The Application Builder** is where you create and modify applications and pages, and it's where you'll probably be spending most of your time.
- **The SQL Workshop** is where you will deal directly with the underlying database objects and their related data. Think of it as a web-based version of SQL*PLUS with some GUI goodness thrown in to make things easier.
- **Team Development** is the section that lets you enter and track information related to the development of APEX applications.
- **Administration** is where you can manage the details of your workspace, its defaults, users, groups, etc. Be aware that a Workspace Administrator will have more options available to them than a standard developer.

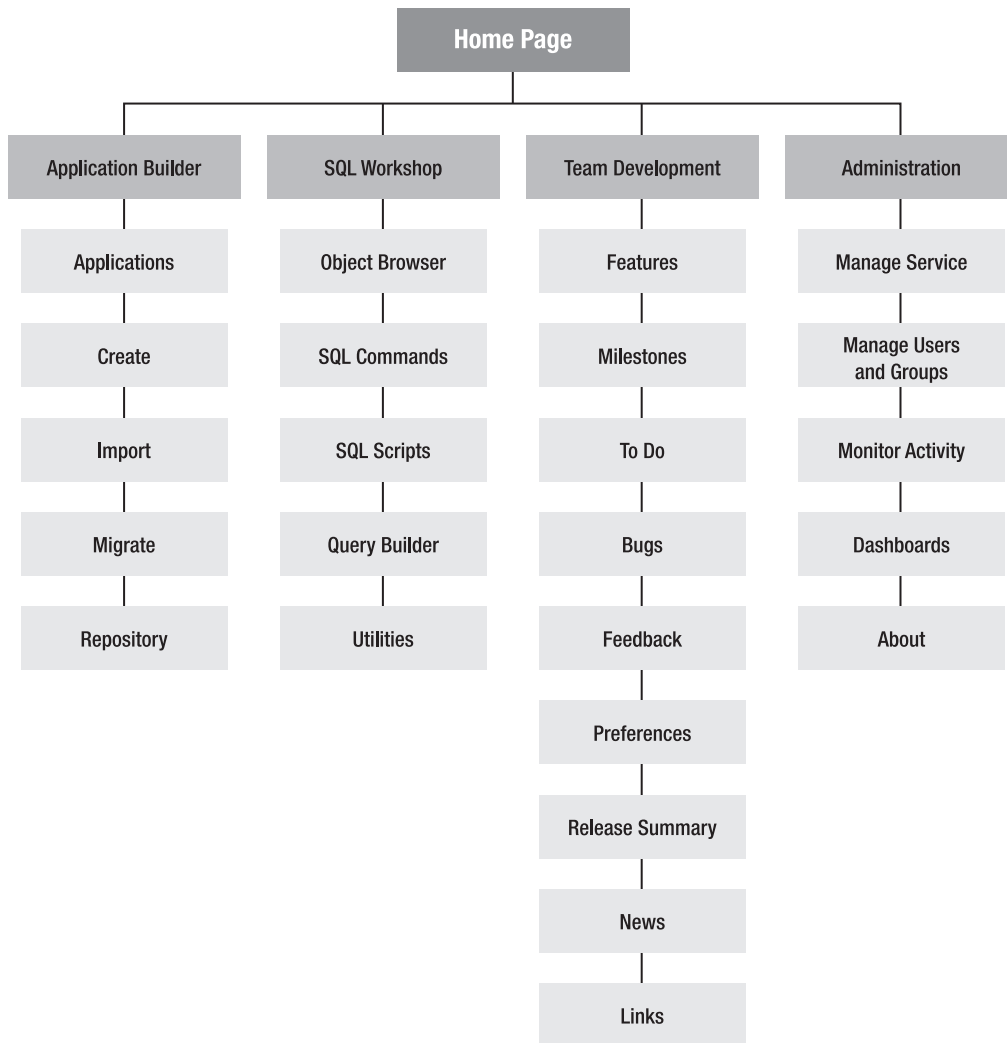


Figure 2–5. APEX 4 hierarchical menu structure

The Home Page

Once you login to your workspace, you'll be presented with the Workspace Home Page, as shown in Figure 2–6. The Home Page is your gateway to the rest of the development environment and provides some high level information about what's going on within the workspace.

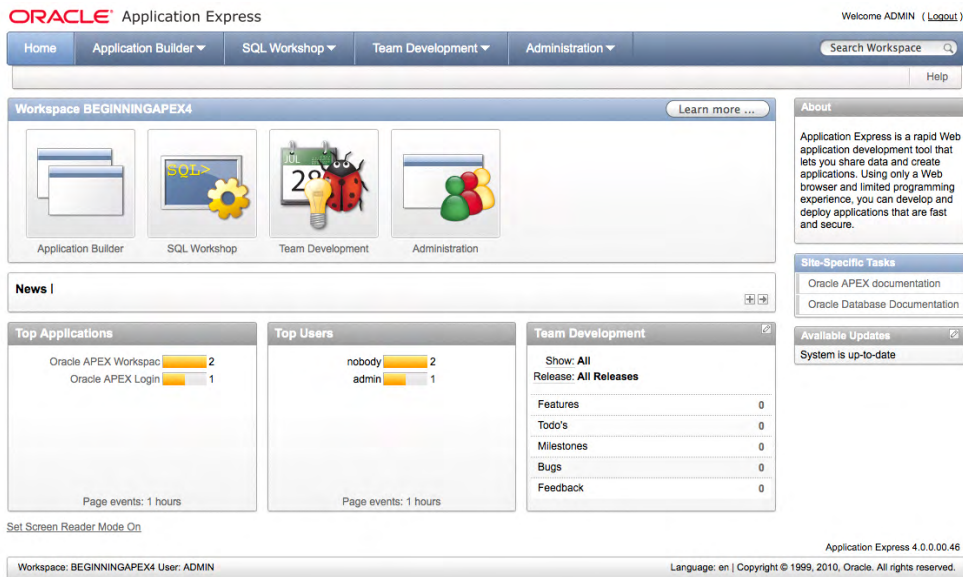


Figure 2-6. APEX development home screen

Along the very top is the APEX Logo and to the right of that is the Navigation Bar that contains a welcome message including your username and a logout link enabling you to logout of this workspace and navigate back to the main APEX login page. Just below the Navigation Bar is the main menu bar that will be available to you throughout the developer interface. It gives direct access to many of the sections you'll need to get to quickly while you're developing applications. It's worth noting that each section of the menu bar is broken down into two pieces. For instance, if you were to click directly on the Application Builder item, you would immediately be taken to the Application Builder home page. However, if you were to click on the small downward pointing triangle just to the right, you would be presented with a more detailed drop-down menu that would allow you to choose your destination a bit more granularly, as in Figure 2-7.

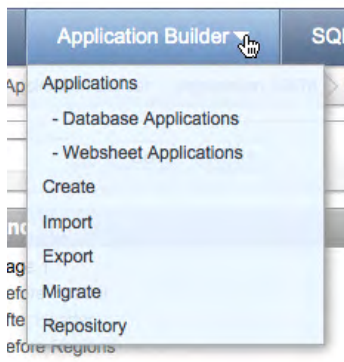


Figure 2-7. Using the drop-down menus on the menu bar

At the far right of the menu bar is a search box that allows you to perform context-sensitive searches. The context of the search depends on where you are in the application builder. For instance, if you are on the workspace home page, your search will be across the entire workspace. However, if you are in the Application Builder or the Administration section, the search will be limited contextually to those specific areas.

Beneath the main menu bar is the breadcrumb region. This not only gives you a visual clue of where you are in the hierarchy of the workspace, but each breadcrumb is also a quick link that will take you back to that specific spot in the hierarchy. You'll be creating breadcrumbs in your own application that will perform a very similar job.

Lastly, to the far right of the breadcrumbs is the HELP link. This pops open a new window containing context-sensitive and searchable help for APEX. In my experience, most of the details that are in the PDF manuals provided by Oracle are also contained within the context-sensitive help. This is a great resource for quickly looking things up if you don't have the manuals or PDFs on hand.

The rest of the page is dedicated to either giving you a quick link to the four main sections or providing you with information about what's going on in the workspace. There is a News region, as shown in Figure 2-8, that allows the developers within a workspace to enter information they want others within the workspace to see. If there is more than one news item active, this region will scroll through the news items, wrapping back around to the first item when it has reached the end of the list.



Figure 2-8. Home page News scroller

The three regions at the bottom of the home page show an overview of the activity within the workspace. The regions, from left to right, show the Top Applications, Top Users, and Team Development activity within the workspace. In a new workspace, there probably won't be anything in these regions, but as you work your way through the book, you'll see that start to change.

You'll notice that most of the main pages for each section of the development environment adhere to this dashboard style home page interface, the notable exception being the Application Builder. Let's look at that section first.

Application Builder

The Application Builder is the core of the APEX Application Development Environment. While you'll use the SQL Workshop to manipulate the underlying database objects, you'll use the Application Builder to do most of the real work when it comes to coding, testing, and debugging your applications.

The Application Builder Home Page

Clicking on the Application Builder menu option takes you to the Application Builder home page. Like most of the home pages, it's laid out with the menu bar across the top and regions that hold tasks and quick links down the right hand side.

The main difference is that Application Builder's home page does not house any dashboard-style summaries. Instead, this is where you will see a list of the different applications contained within your workspace. (Figure 2-9 provides an example). It is possible, depending on your APEX Instance settings, that you might see some sample applications installed by the workspace administrator, but don't be alarmed if you don't see any applications at all.

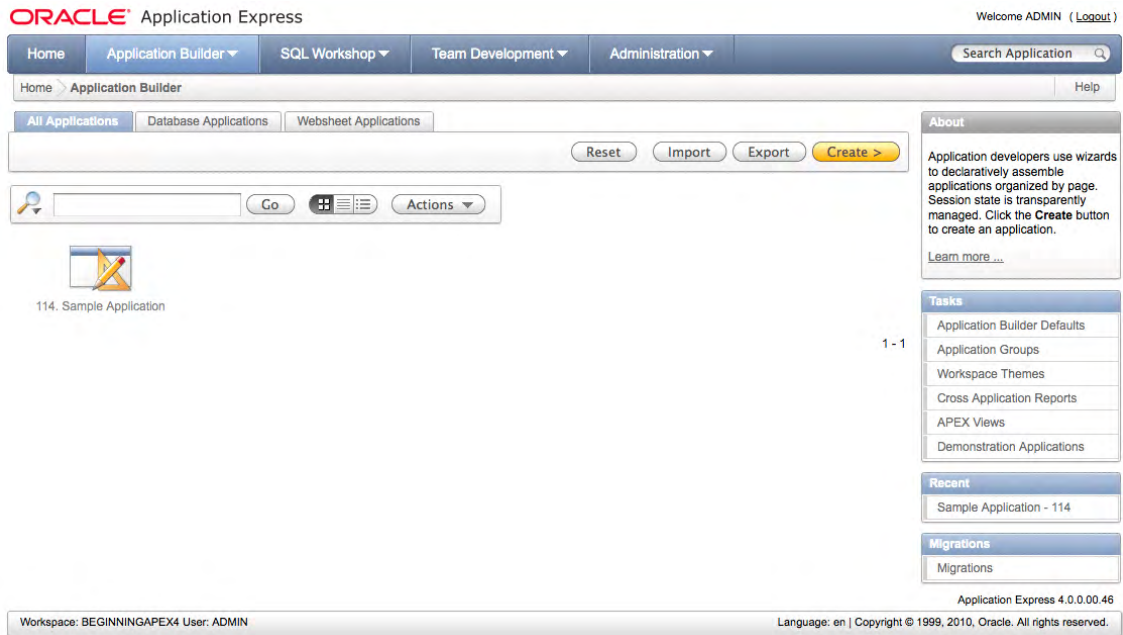


Figure 2–9. The Application Builder home page

Notice the set of tabs above the application list. This tab set provides a high level filter of which applications you see from all of those in your workspace.

All Applications shows all application types (Database and Websheet).

Database Applications shows only those applications that are built on top of a database schema. These are considered standard APEX applications.

Websheet Applications shows only those applications that are Websheet-style applications. These are new to APEX 4 and We'll talk more about them in Chapters ?? and ??.

As you can see, there is one application in my workspace named **Demo Application**. However, there isn't much information about it other than its name and the Application ID (16878) associated with it. This is where you begin to see the beauty of what APEX can do, not only in the Developer UI, but also in your applications.

The list of applications you see is actually a style of report called an Interactive Report or IR. IRs allow you to customize how the report and its contents are displayed. IRs are used throughout the APEX Development Interface and can also be used when creating your own applications. They're extremely powerful tools and you'll be using them a lot.

On the right side of the page are three regions that show Application Builder related tasks, recently edited applications, and a link to the Application Migration wizard. You'll be dealing more with these later, but for now you want to drill in to see the details of an application.

The Application Home Page

Clicking on one of the applications listed will drill into the Application home page, as shown in Figure 2–10. This page is very similar to the Application Builder home page, but instead shows all the pages within a specific application. Again, it uses an IR, so you can customize the way you see this data.

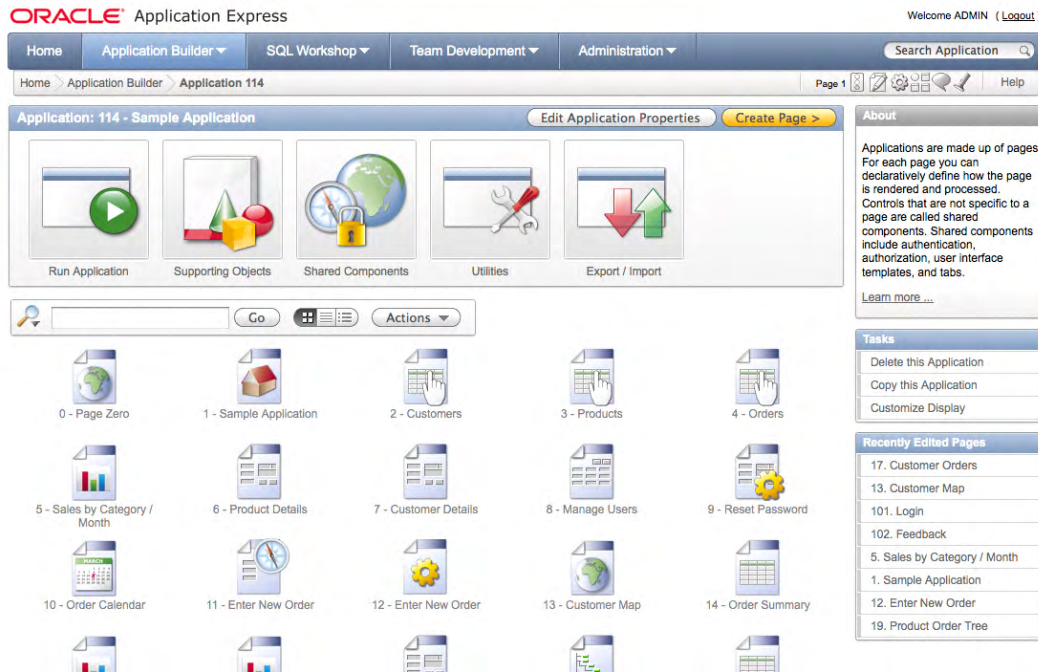


Figure 2–10. The application home page

Again, notice the way the page is structured with page related tasks and recently edited pages presented along the right side of the page. This layout will become a familiar theme as you navigate through the interface.

From here, you can click on any of the listed pages to edit that page. You can also run, export, and import the application, edit the supporting objects or shared components, and access the application related utilities.

We'll wait until you get into the depths of writing an application to go any further in the application builder, but this gives you a flavor of what to expect as you move forward.

SQL Workshop

The SQL Workshop is a suite of tools that provide developers the ability to view and manage database objects in the underlying schema(s) assigned to the workspace. The SQL Workshop home page shown in Figure 2–11 provides access to each of the underlying tools and some high level information about recently created objects and commands that have been run.

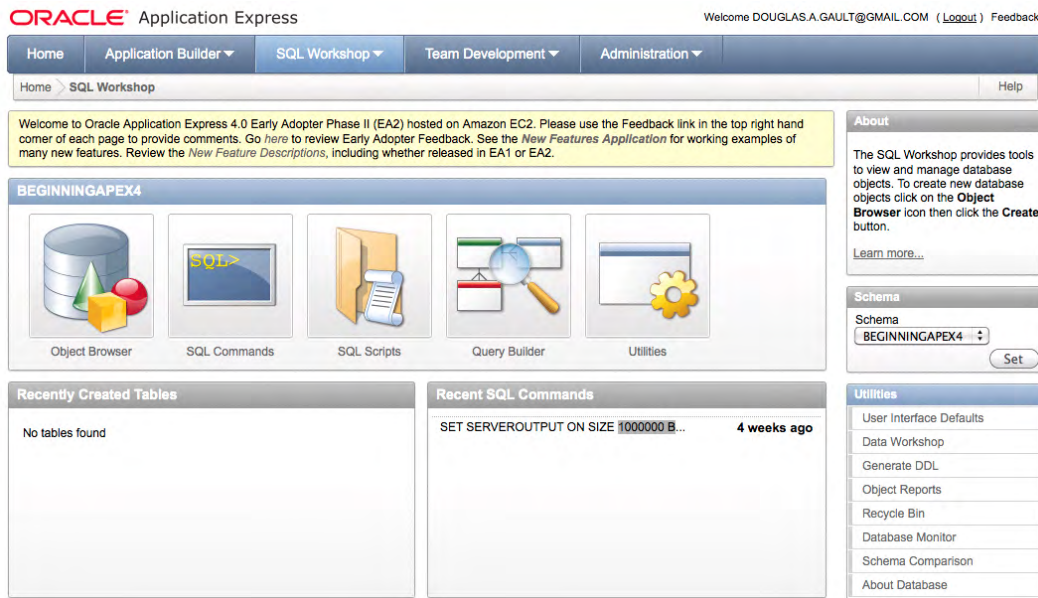


Figure 2–11. The SQL Workshop home page

Because there may be more than one schema assigned to the workspace, there is a schema selection dialog on the right of the screen that allows you to select and set the default schema for all of the tools. You may change the schema you're working in within each of the tools as well.

Below the Schema Selection dialog is a set of direct links to each of the individual utilities you have access to. You can also reach these by clicking the utilities icon in the main section of the screen.

Scrolling down on this page reveals quick links to creating new database objects. Again, each of these can be done from within the Object Browser, but they are provided here as a quick link to the specific functionality.

Lastly, there is a link to a wizard that will create a PL/SQL API on top of a table or set of tables. This is a handy utility when you need to introduce logic between the UI and the actual data tables.

Each of the individual tools deserves its own introduction, so let's spend some time now looking at what they are and what they can achieve. You'll be using this area of APEX more heavily when you create the database object for your application.

The Object Browser

If you've been working with databases for any length of time, you've probably worked with one of the more popular GUI tools that allow you to browse and manage database objects within a schema. The APEX Object Browser is a very similar tool presented to you through your web browser. Figure 2–12 shows the Object Browser being used to examine the table `DEMO_PRODUCT_INFO`.

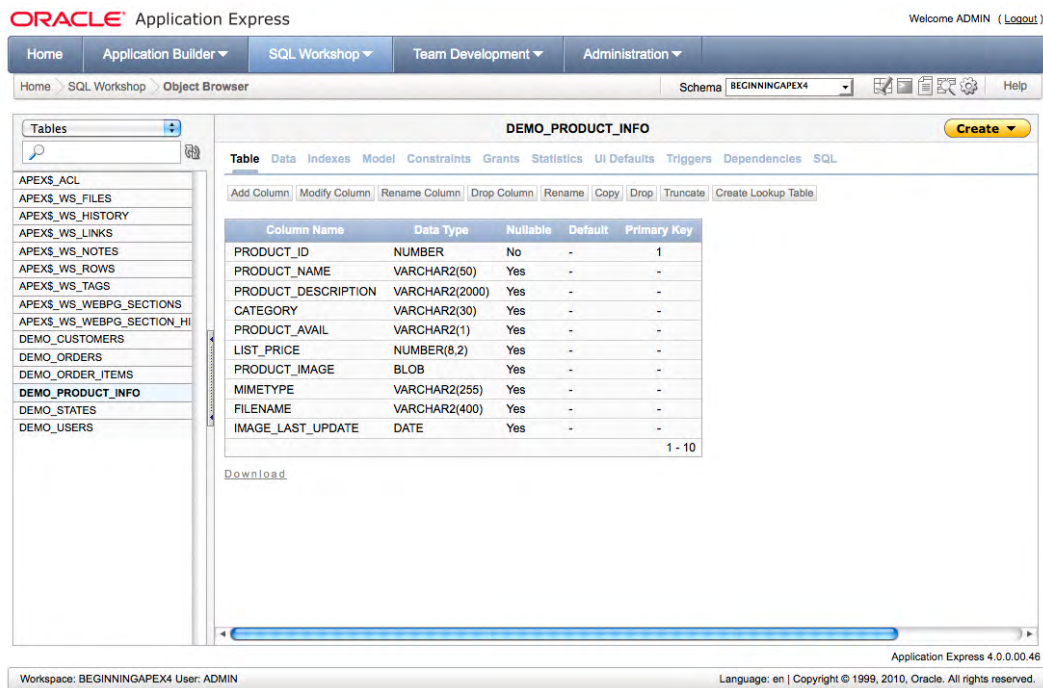


Figure 2–12. The APEX Object Browser

The name *object browser* is somewhat of a misnomer as the tool can actually be used not only to browse the objects in the underlying schema(s), but also to create new objects, browse and edit data, delete objects, and edit object definitions. While there are some limitations to the types of objects it can manipulate, it's powerful enough to do most of the daily tasks that an application developer will need to tackle.

You chose the object type you want to work with by selecting it from the drop down list in the upper left hand corner. You can search the selected object type by entering a text string into the search box just below it and clicking the refresh icon just to the right. Clicking on the name of an object displays its properties and the links to drill into more details.

While the interface for the Object Browser is pretty intuitive, there are some interesting things to note. In the upper right hand corner is a drop down list that allows you to set the current schema. The list contains all schemas currently assigned to the workspace. You can switch between them simply by choosing a new one from the list.

Also, to the right of the drop down list is a set of quick link icons that takes you directly to the other tools within the SQL Workshop.

The SQL Commands Interface

The SQL Command Interface allows you to interact with the underlying schema(s) using standard SQL commands or PL/SQL as you would in any other GUI tool or SQL*Plus. The difference is that you can save the statements for use at a later time. Figure 2–13 shows a simple SQL statement as executed in the SQL Command Interface.

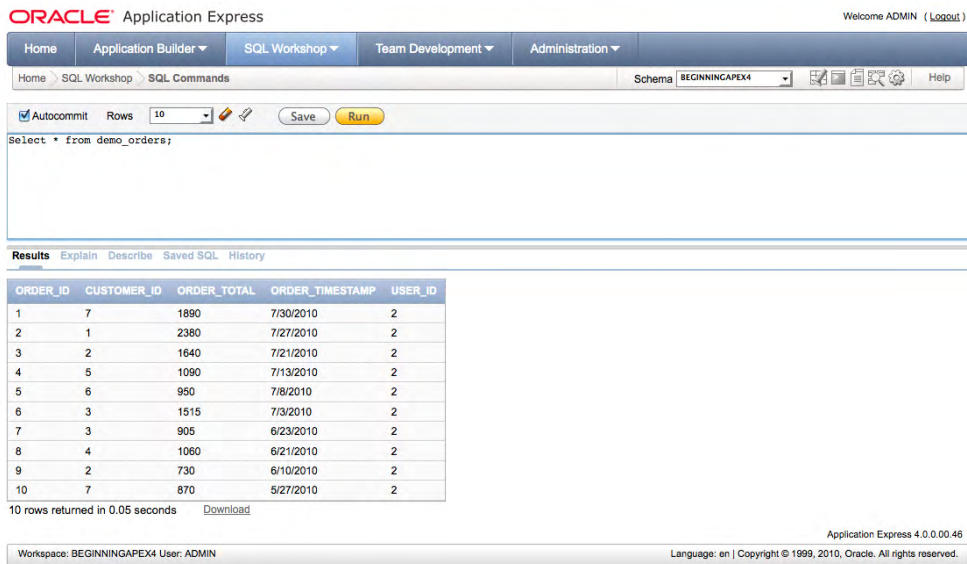


Figure 2–13. The SQL Command Interface

While its core function is quite straight forward, the SQL Command Interface is more robust than it first appears. Beyond the ability to save and retrieve SQL and PL/SQL, it also has the ability to run explain plans on statements and allows you to view your statement history. Therefore, if you ran a script or statement that was particularly useful, but forgot to save it, you still have the potential to retrieve it from the history buffer.

The SQL Command Interface also integrates with the Query Builder (described below) allowing you to load and manipulate saved statements built in the Query Builder.

■ **Note** By default, all SQL statements executed via the SQL Commands Interface are automatically committed. To override this setting and enter into transactional mode, uncheck the Autocommit checkbox in the tool bar. Once this is done, you will be able to manually commit and rollback your SQL statement.

There is no way to turn off Autocommit permanently so you will need to remember to do this any time you want to enter transactional mode.

SQL Scripts Interface

The SQL Scripts Interface allows you to manage and run sets of SQL commands saved into script files. A single script can contain one or more SQL statements or PL/SQL blocks. SQL scripts that are coded outside of APEX can be loaded into the SQL Scripts Repository and edited or run from there. You may also create SQL scripts from scratch using the SQL Scripts Interface. Figure 2–14 shows the main SQL Scripts Interface page.

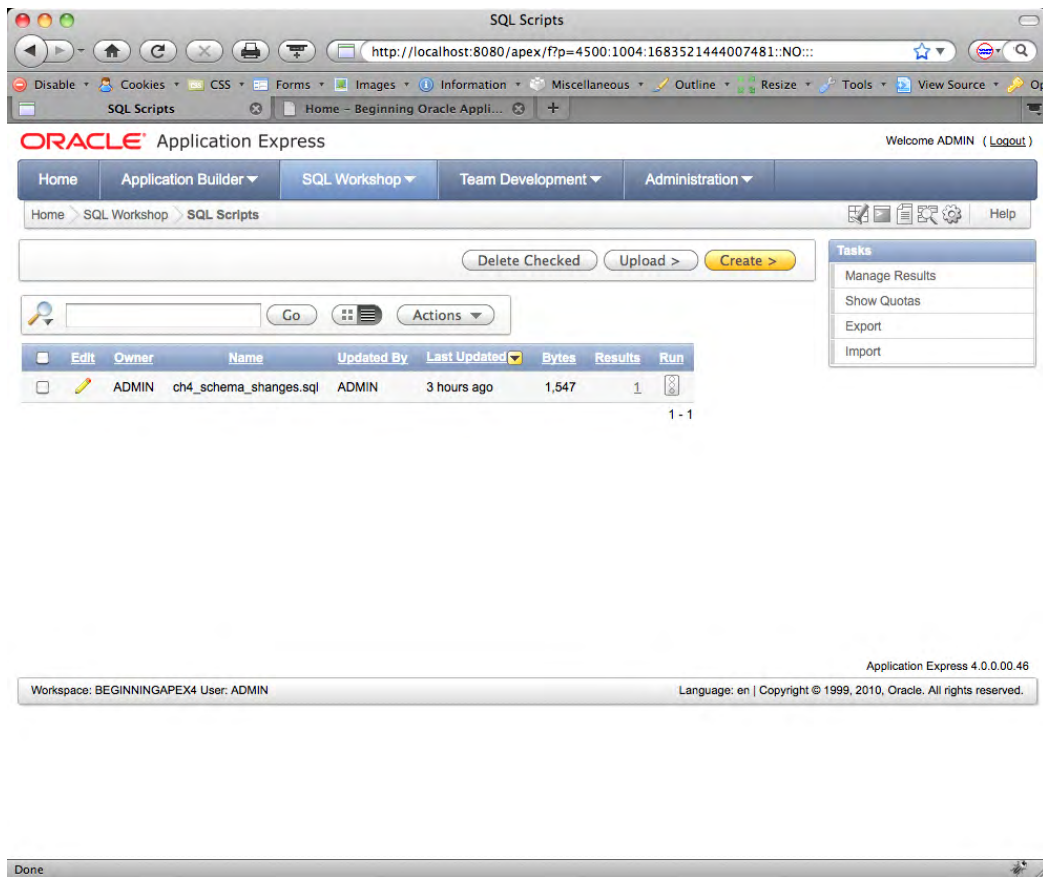


Figure 2–14. The main SQL Scripts Interface page

In this example, there is one script, called `CLEAN_SCHEMA`, loaded into the script repository. By clicking on the edit icon, you can edit the contents of the script, as shown in Figure 2–15. APEX 4 now provides syntax highlighting inside the script editor. The editor also has a Find and Replace function as well as undo and redo.

You can also download the script to a local file so that you can edit it in your favorite local text editor. When you're done, simply cut and paste it back into the editor or upload it as a new script file.

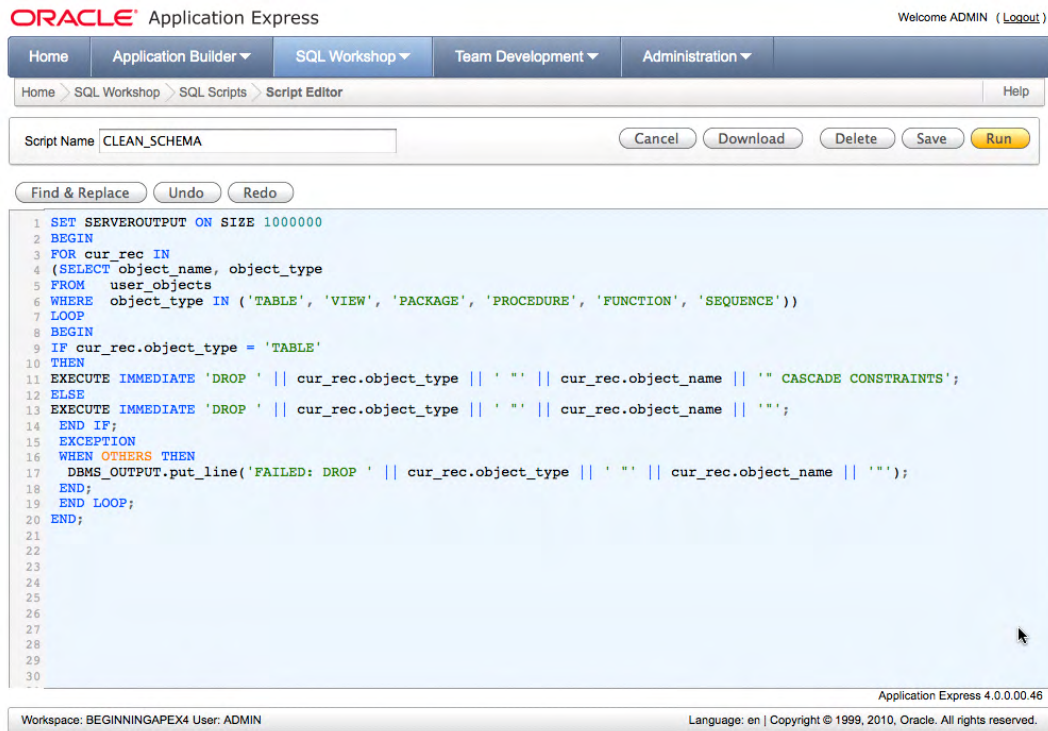


Figure 2–15. The SQL Script Editor

■ **Note** When uploading a script file to the repository, the name of the script must be unique. You can't overwrite an existing script file of the same name with a new version without first deleting the script from the script repository.

Once a script is ready to run, you can click the Run icon in the list (or the Run button in the editor) and you will be stepped through the Run Script wizard. This allows you to choose whether you want to run the script immediately or run it in batch mode. If you chose batch mode, your script will be entered into a queue where it will be executed when it reaches the front of the queue.

Either way you will be taken to the Manage Script Results page of the SQL Script Interface as shown in Figure 2–16. This screen allows you to see the status and certain high level details of the script's execution. In the case of scripts that have been submitted in batch mode, you can also see the status of the script within the queue.

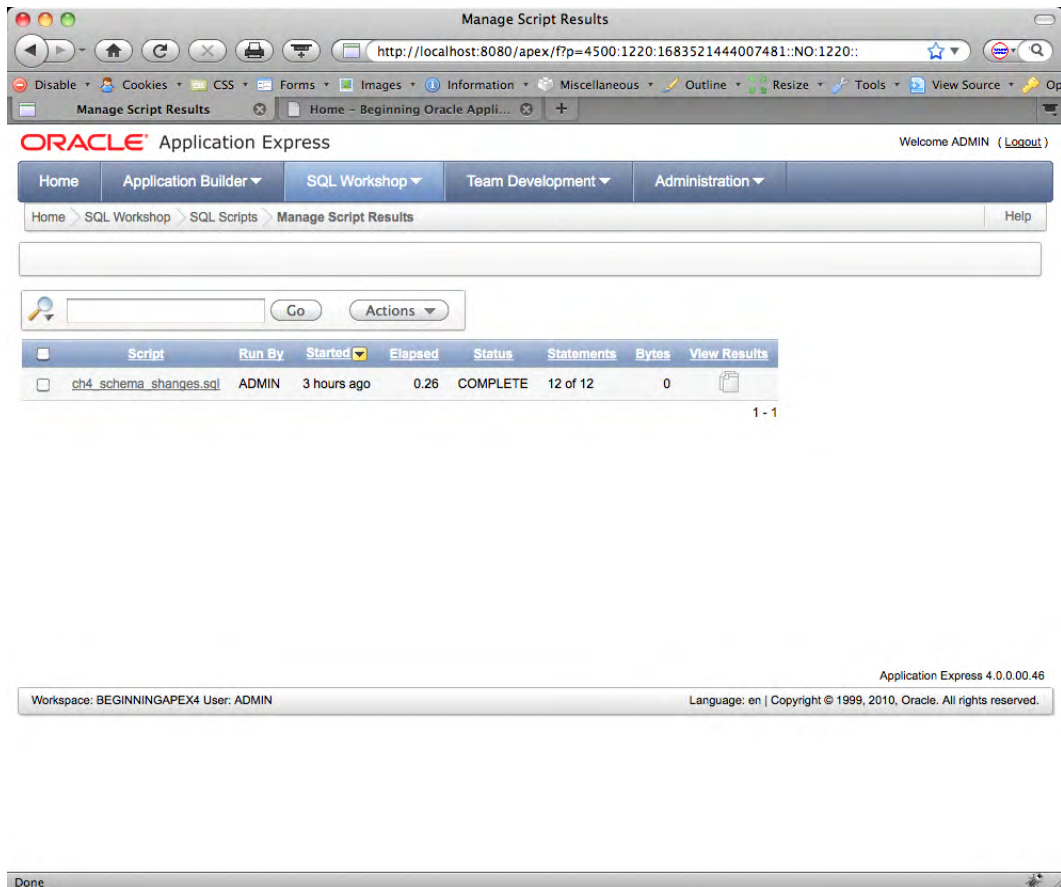


Figure 2–16. The Manage Script Results page

Clicking on the View Results icon will show you the final results of running the script. In Figure 2–17, you can see that the script had an error, the details of which are displayed within the body of the report. If the script were successful, no errors would be shown and the statement results at the bottom of the page would show zero errors.

The screenshot shows the Oracle Application Express interface. At the top, there's a navigation bar with tabs: Home, Application Builder, SQL Workshop (selected), Team Development, and Administration. Below this is a breadcrumb trail: Home > SQL Workshop > SQL Scripts > Results. The main area displays the execution results for a script named 'CLEAN_SCHEMA'. The status is 'Complete'. Below the status, there's a table with columns: Number, Elapsed, Statement, Feedback, and Rows. The table contains one row with the following data:

Number	Elapsed	Statement	Feedback	Rows
1	0.10	BEGIN FOR cur_rec IN (SELECT object_name, object_type	ORA-06550: line 8, column 1: PLS-00103: Encountered the symbol "IF" when expecting one of the following: := , (@ % ; The symbol "" was substituted for "IF" to continue. ORA-06550: line 14, column 2: PLS-00103: Encountered the symbol "EXCEPTION" when expecting one of the following: (begin case declare end exit for goto if loop mod null pragma raise return select update while with	-

Below the table, it says 'row(s) 1 - 1 of 1'. At the bottom left, there's a 'Download' link. At the bottom right, there's a status bar showing 'Workspace: BEGINNINGAPEX4 User: DOUGLAS.A.GAULT@GMAIL.COM' and 'Language: en | Copyright © 1999, 2010, Oracle. All rights reserved.'

Figure 2–17. An example of an error from the SQL Script Interface

The Query Builder

The Query Builder is a utility that allows you to build SQL Select statements using a more graphical interface. While it's not quite drag and drop, it is fairly intuitive.

When you first enter the Query Builder, you're presented with a screen that lists all of the tables and views that are available in the currently active schema. Figure 2–18 shows the initial Query Builder screen.

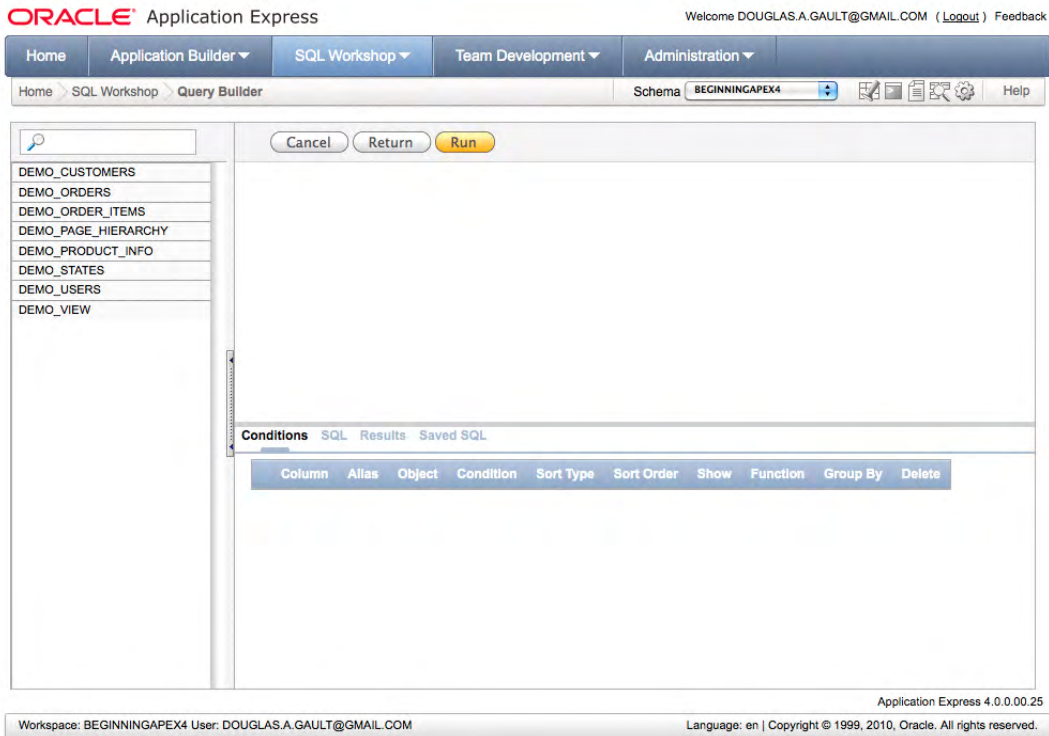


Figure 2–18. The initial Query Builder screen

From here you may begin to build your query. To include a table in your select statement, simply click on it in the list to the left. A representation of the table will be placed in the blank region of the screen above the Conditions region. You may add as many tables as you like to your query and may even include the same table more than once simply by clicking on it again. You'll notice that if you include more than one instance of the same table, the new instance will be suffixed with a sequence number differentiating it from the original table.

Figure 2–19 shows an example graphical representation for the DEMO_ORDERS table and outlines the different interactive features.

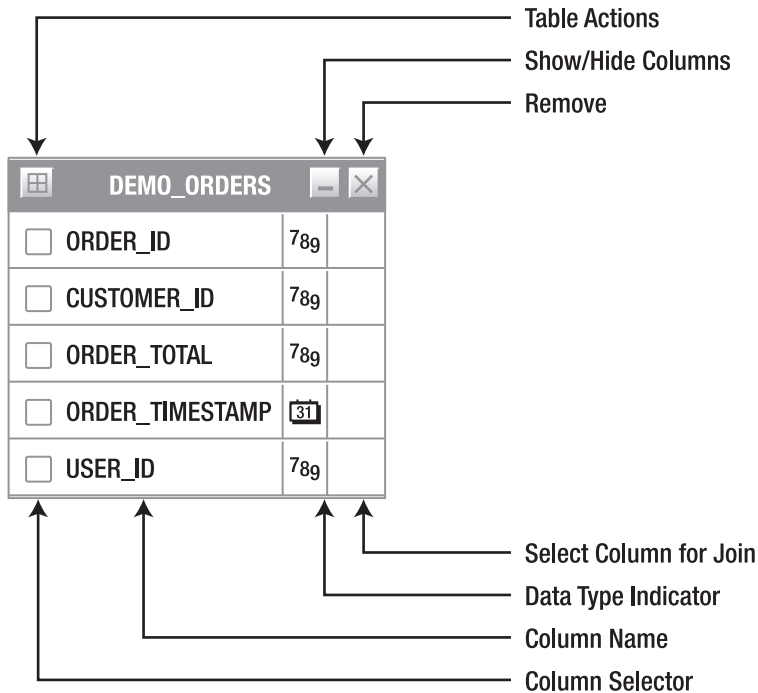


Figure 2-19. The DEMO_ORDERS table as represented in the Query Builder

Taken from top to bottom as they appear in Figure 2-19, these action areas are:

Table Actions displays a dialog allowing you to do one of several things:

- **Check All** allows you to quickly select or deselect all columns of the object for inclusion in the query being built.
- **Add Parent** allows you to select and add a parent table, as defined by foreign key relationships, to the query builder.
- **Add Child** allows you to select and add a child table, as defined by foreign key relationships, to the query builder.

Show/Hide Columns expands and collapses the object so that the column definitions are shown or hidden.

Remove deletes the table and any of its related clauses from the select statement.

Select Column for Join is activated by clicking on the blank square next to a column name. This will darken the square and put the Query Builder into “Table Link” mode. Then you can click on another blank square, either within another table or within the same table, and the Query builder will insert an EQUALITY where clause between the two columns in the SQL statement.

Data Type Indicator indicates the data type of the column, such as number, character, date, etc.

Column Name indicates the column name as defined in the table description.

Column Selector allows you to individually select or deselect columns to be included in the SQL Statement for processing. This may also include columns that you may want to use in the where clause but not display in the output of the SQL statement. The basic rule is that you need to select all the columns you want to display, but you do not necessarily have to display all the columns you select.

As you begin to add and join tables and select columns to operate upon, the region at the bottom of the screen will begin to change. This region is subdivided into several tabs, as follows:

Conditions Tab shows one row for each column selected in the area above and allows you to further define its attributes. (More on this feature in just a moment.)

SQL Tab displays the SQL statement as the wizard builds it. While it's not directly editable, you can easily highlight and cut the statement to the clipboard from here.

Results Tab shows the results of running the SQL statement and allows you to download the resulting data in CSV format.

Saved SQL Tab allows you to save, recall, and manage statements that have been built with the Query Builder. There are also filters that allow you to search and limit which saved queries display.

All but the Conditions Tab are self explanatory, so I'll go over this one in a little bit more detail. Figure 2–20 shows an example three table join with five columns selected to operate upon.

The screenshot shows three tables: DEMO_CUSTOMERS_2, DEMO_ORDERS, and DEMO_USERS. DEMO_CUSTOMERS_2 has columns: CUSTOMER_ID, CUST_FIRST_NAME, CUST_LAST_NAME, CUST_STREET_ADDRESS1, CUST_STREET_ADDRESS2, CUST_CITY, CUST_STATE. DEMO_ORDERS has columns: ORDER_ID, CUSTOMER_ID, ORDER_TOTAL, ORDER_TIMESTAMP, USER_ID. DEMO_USERS has columns: USER_ID, USER_NAME, PASSWORD, CREATED_ON, QUOTA, PRODUCTS, EXPIRES_ON, ADMIN_USER. Lines connect DEMO_CUSTOMERS_2 to DEMO_ORDERS on CUSTOMER_ID, and DEMO_ORDERS to DEMO_USERS on USER_ID.

Below the tables is the 'Conditions' tab with the following table:

Column	Alias	Object	Condition	Sort Type	Sort Order	Show	Function	Group By	Delete
USER_NAME	USER_NAME	DEMO_USERS		Asc		<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>
CUSTOMER_ID	CUSTOMER_ID	DEMO_CUSTOMERS_2		Asc		<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>
CUST_FIRST_NAME	CUST_FIRST_NAME	DEMO_CUSTOMERS_2		Asc	2	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>
CUST_LAST_NAME	CUST_LAST_NAME	DEMO_CUSTOMERS_2		Asc	1	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>
ORDER_TOTAL	SUM_OF_ORDERS	DEMO_ORDERS	<500	Asc		<input checked="" type="checkbox"/>	SUM	<input type="checkbox"/>	<input type="checkbox"/>

Figure 2–20. An example three table join

In this example, the following modifications have been applied to the query:

- Changed the alias of the ORDER_TOTAL column to SUM_OF_ORDERS.
- Limited the result set to only those records where ORDER_TOTAL is less than 500.
- Sorted the records returned by CUST_LAST_NAME, CUST_FIRST_NAME Ascending.
- Performed a SUM function on the ORDER_TOTAL column.
- Grouped the query by USER_NAME, CUSTOMER_ID, CUST_FIRST_NAME, CUST_LAST_NAME.

Based on the column selections and the restrictions and changes introduced in the Conditions tab, the SQL statement (as it appears in the SQL Tab) would look like this:

```
select  "DEMO_USERS"."USER_NAME" as "USER_NAME",
        "DEMO_CUSTOMERS_2"."CUSTOMER_ID" as "CUSTOMER_ID",
        "DEMO_CUSTOMERS_2"."CUST_FIRST_NAME" as "CUST_FIRST_NAME",
        "DEMO_CUSTOMERS_2"."CUST_LAST_NAME" as "CUST_LAST_NAME",
        sum(DEMO_ORDERS.ORDER_TOTAL) as "SUM_OF_ORDERS"
from    "DEMO_CUSTOMERS" "DEMO_CUSTOMERS_2",
        "DEMO_USERS" "DEMO_USERS",
        "DEMO_ORDERS" "DEMO_ORDERS"
where   "DEMO_USERS"."USER_ID"="DEMO_ORDERS"."USER_ID"
and     "DEMO_CUSTOMERS_2"."CUSTOMER_ID"="DEMO_ORDERS"."CUSTOMER_ID"
and     "DEMO_ORDERS"."ORDER_TOTAL" <500
group by DEMO_USERS.USER_NAME, DEMO_CUSTOMERS_2.CUSTOMER_ID,
        DEMO_CUSTOMERS_2.CUST_FIRST_NAME, DEMO_CUSTOMERS_2.CUST_LAST_NAME
```

While the query builder is very useful and allows you to put together a basic query fairly quickly using a simple GUI, it does have its limitations. I personally use the Query Builder to get the skeleton of a query defined. I then take the query to the SQL Commands window or into my favorite SQL IDE and I fine tune it from there.

As a final note, it's worth mentioning that the Query Builder is linked to from several places within APEX, so any time you're prompted for a SQL statement (for example, as the basis for a report) you will be able to open the Query Builder in a pop-up window and return the query to the calling form.

Utilities

The SQL Workshop Utilities section gives you access to tools and reports that help you view and manage information about the underlying database objects and their data. In this section, I'll introduce each tool set and its main purpose. However, the majority of these tools are very straightforward, so in most cases I'll leave the deep details for you to explore on your own.

The Utilities home page (as shown in Figure 2-21) presents you with a quick icon based menu to each of the individual utility areas. Clicking on any one of these icons takes you directly to the tools page for that category.

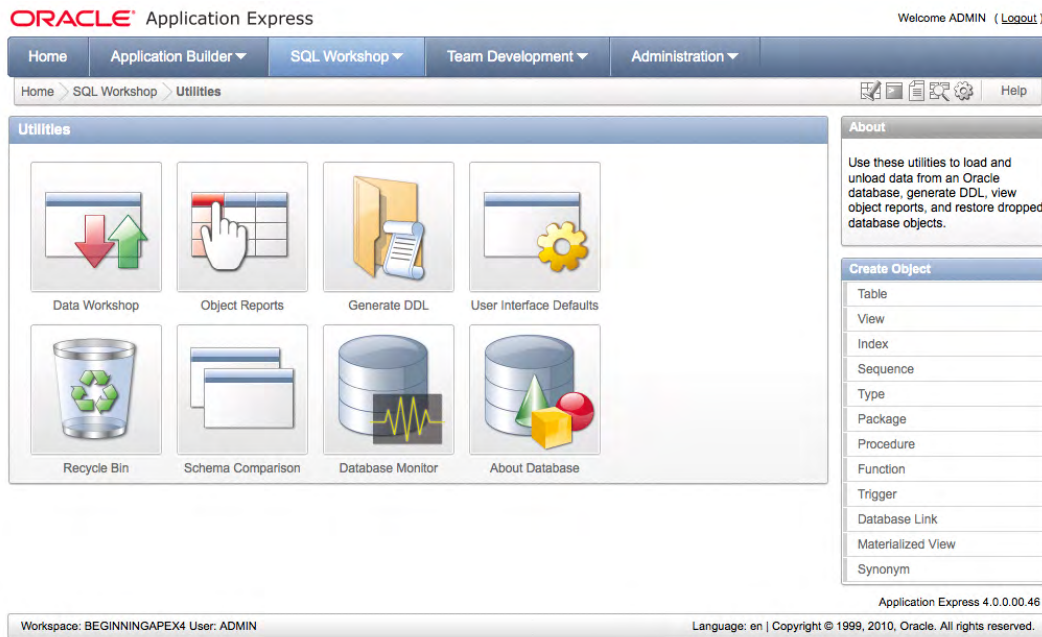


Figure 2–21. The SQL Workshop Utilities home page

The Data Workshop provides tools that import and export data in many different formats including comma or tab separated, XML, or spreadsheet data. These tools also help you manage files that you have loaded into either the Text or Spreadsheet repository.

The Object Reports provide detailed reporting capabilities on all of the objects within the schemas tied to the workspace. Reports include information about the tables and objects, their sizes, statistics, grants and privileges, their status, and exception reports to show possible problem areas.

The Generate DDL wizard allows you to choose a schema associated with the workspace and generates a script that can be used to recreate some or all of the objects with that schema based on your selection. The generated script does not include any insert statements for the data that resides within the database objects, but it's a good way to easily recreate the underlying objects an application might use.

When an object is dropped, Oracle does not immediately remove the space associated with the table, but instead renames the table and places it and its associated storage in the Recycle Bin. The Recycle Bin utility allows you to view and potentially recover objects that have been dropped from the schemas associated with a workspace. You may also purge the Recycle Bin, allowing the space to be reclaimed by the Oracle database for use somewhere else.

The Schema Comparison utility allows you to compare the objects within two separate schemas and create a difference report. You may choose to compare only certain attributes or all attributes of the objects within the selected schemas. The limitation here is that both schemas must be assigned to the workspace in order for the comparison to take place.

The last two utilities (Database Monitor and About Database) are special utilities that require the user running them to have access to a database login that has been granted the DBA role. The Database Monitor utilities allow the privileged user to view Sessions, Systems Statistics, Top SQL, and Long Operations reports. The About Database report shows detailed information about the database instance and the APEX environment.

Administration and Team Development

The last two functional areas of the UI, Administration and Team Development, are complex enough to truly deserve their own chapters. Therefore, I'll refer you to the chapters that cover these areas in depth. Chapter ?? covers deploying applications; Chapter ?? is about managing workspaces; Chapter ?? goes over team development.

You'll definitely be dipping into administrative tasks throughout this book, so if you want to have a full understanding of administration before we start, I would suggest that you take a detour and read these chapters now to get a good foundation. However if you're prepared to learn on the fly, go to the next section where you'll start the real programming.

Summary

The architecture of APEX may seem a bit daunting at first, but once you actually start working with it, things will begin to fall into place and you'll understand more and more about how it all fits together. If you take away only one thing from this chapter, let it be that a workspace is essentially your development sandbox. Everything you do happens in the context of a workspace. Everything else—from a development standpoint—is much like any other development environment. Are you building a new application? Then it needs to be created within a workspace. Do you need access to a schema to build that app? Then it needs to be assigned to your workspace. You get the picture. Now ... on to the fun!



Identifying the Problem & Designing the Solution

Every computer system is (or at least should be) the result of solving some type of problem. While “Hello World” apps are great, I firmly believe that the best way to learn any technology is to apply it to a real problem and see how things really work.

We’re going to be adhering to that principle throughout this book. This chapter discusses a very common problem in most organizations that can be solved technically. You’ll also look at some of the specific things you need to consider when designing web-based systems in general and APEX specifically.

Identifying System Requirements

Almost every company, no matter the size, will at some point need to implement some sort of help desk. Whether it’s an internal one to track employee question and problems or an external one to track client issues with commercial software or hardware, the basics of a help desk system are fairly standard.

Most help desk systems are driven by the notion of a “Trouble Ticket” or “Ticket.” This term is a leftover from the days before computers: most problems were reported over the phone, and troubleshooters used a physical paper ticket to log a call. The information contained on that paper ticket included a description of the problem, the person having the problem, when the problem was logged, etc. Then, throughout the process of troubleshooting and, hopefully, solving the problem, the engineers would write down each step of the process and include any documentation of the problem they gathered along the way. Today, it would be very surprising to see a help desk system that wasn’t computerized, even if it’s only a spreadsheet of issues with notes and statuses.

You’re going to attack the help desk system with APEX. Before you dive in, you need to clearly understand the problems you are trying to solve. If nothing else, you need to review the current system.

Never a Clean Slate

Almost no computer system written today starts from scratch. There is almost always something in place, even if it’s just some loose guidelines or ideas.

For this example, let’s say that your company has a very basic system in place, but it’s no longer meeting the needs of your growing user community. Your goal is to create a new system that will make the logging of issues and their solutions much easier for everyone involved; however, to do that, you must understand the needs of the users and the functionality of the system that is in place now.

A Broken System

In general, the users of help desk systems can be categorized into two groups: people who log problems (end users) and people who help solve the problems (technicians). Depending on which user community you fall into, it's likely you will have different needs, but overall, the system should help the end users and the technicians communicate about the problem or issue.

The first step is to understand how your help desk is being managed today and why it's not working. Speaking to both the technicians and the end users can provide a huge amount of information, but the challenge is that this information usually comes in the form of complaints about the current system.

Quizzing the end users reveals that their main complaint is that they never know the status of the problems that they've logged. They can go days, sometimes weeks, without communication from the technicians, and in the eyes of the user, no communication means that no one is working on their problem. Another user complaint is that the help desk technicians often don't know how to contact them to ask further questions or communicate progress.

On the other end of the issue, the technicians are overloaded. Originally, the help desk was only one person, and ticket information was kept in an Excel spreadsheet. Now there are several technicians working independently. While performing their daily duties, each needs to update the spreadsheet with information regarding the tickets assigned to him. The increasing number of people accessing a single spreadsheet causes problems, as only one person is able to open and update the spreadsheet at any given time. The technicians are also tired of constantly being called by users wanting an update on the status of their issues.

It's obvious that the system is broken. Neither the users nor the technicians are happy about the situation. It's your job to take the information you've gleaned from these conversations and design something that will address the needs of both user communities.

How Do You Fix Things?

With the information you've gathered so far, you can now define some loose requirements and break them down by user type so you can have a much clearer understanding of what each community needs. Then, from those requirements, you can begin to think about the database design that you'll need to create in support of them.

Defining the Requirements

You can look at requirements from two perspectives. End users will have one set of requirements and technicians another. Some requirements will overlap between the two groups. Others will be unique to one group or the other.

End users should be able to:

- Create a new ticket outlining their problem.
- See the status and progress of tickets.

Technicians should be able to:

- Easily identify and view new tickets.
- Easily identify which tickets are directly assigned to them.
- Search existing tickets.
- Create new tickets on behalf of an end user.

- Assign tickets to other technicians.
- Add details (comments, information, and attachments) to tickets.
- Update the status of a ticket.

Although you could go a lot further, these requirements form the basis of a pretty complete help desk system. You can always add functionality to it later when you have a better understanding of what else the users and the company might need.

Extrapolating to a Database Design

Having stated the requirements, you can begin to extrapolate the database objects you'll need to create to store the data. If you're new to database design, here's a quick trick to help identify the entities for which you'll need to build tables: go back through your requirements and look for CONCRETE NOUNS that represent the highest-level objects you need to track. As you find these nouns, try to identify if they are actually at the highest level, or if they are merely an attribute of something bigger.

If you follow the described process with your brief requirement specification, the nouns USER and TICKET jump out as being the two main things that you want to track. It's tempting to split users down into two different sets, technicians and end users, but in fact the type of user is merely an attribute of a the user itself.

An object that is a little harder to identify is that of the TICKET DETAILS. It is completely valid to think that this would merely be an attribute of a TICKET; however, the clue comes in the fact that you can't concretely identify how many DETAIL entries there will be for any given TICKET. The fact that the number is unknown indicates that you should create a table that is a child of the TICKETS entity called TICKET DETAILS. This way you can enter as many detail records as you need.

So, you've identified three major entities: USERS, TICKETS, and TICKET DETAILS. You now need to think about the attributes of each of these entities and what type of data they will hold. Searching back through the statement of requirements, talking to the technicians about what they track today, and thinking about what types of things you'd want to be able to track during the process of solving a problem, you can identify a number of attributes about your objects. Tables 3–1 through 3–3 show these attributes.

Table 3–1. USER Attributes

Attribute Name	Type of Data	Comment
User ID	Text	A Unique ID for each user
User Name	Text	A login id for each user.
Password	Text	The password used to log into the system.

Table 3–2. TICKET Attributes

Attribute Name	Type of Data	Comment
Ticket ID	Number	A unique way to identify the ticket.
Subject	Text	A brief one-line statement of the problem.
Description	Text	A detailed description of the problem.
Status	Text	The status of the ticket during processing (OPEN, PENDING, CLOSED, etc).
Created By	Text	The user who logged the ticket.
Created On	Date	The date the user created the ticket.
Closed On	Date	The date the ticket was closed.
Assigned To	Text	The technician who is assigned to work on the ticket.

Table 3–3. TICKET DETAIL Attributes

Attribute Name	Type of Data	Comment
Ticket Details ID	Number	A unique way to identify this detail entry.
Ticket ID	Number	This tells us which ticket this detail is linked to.
Details	Text	A text description of any details entered by the technician.
Created By	Text	The user who logged the ticket.
Created On	Date	The date the user created the ticket.

While it's a good to try to be as detailed as possible as early on as you can, we don't have to be perfect here. We can always go back and alter or expand the data we wish to capture as we identify other potential attributes.

System Design with APEX in Mind

Because APEX not only resides within but is built upon the Oracle Database, you would think that designing database objects for APEX is the same as designing for any other system that uses Oracle as a data store—and in some aspects you would be right. However, there are definitely some things that need to be understood when designing for an APEX system that will make your life much easier.

Most of what you do with APEX is created, at least initially, using a series of wizards. If the database objects are designed with APEX in mind, the wizards will do far more work for you; therefore, you will be required to do far less fine tuning manually. In the following sections, I'll discuss the most important design considerations and how they affect what the wizards will do for you.

Table Definition and User Interface Defaults

One such area we'll talk about in more detail later is that of user-interface defaults (UI Defaults). It's important to know that when you make use of UI Defaults, certain table attributes will be translated into default settings used across APEX. Here are some of the more far-reaching things you can do at the table level to help make UI Defaults more useful:

- Placing comments on a table column will seed that item's UI Default help text with the text of the comment.
- Marking a column as NOT NULL at the database level will trigger a Required Flag to be set within the UI Defaults.
- Date and Timestamp data types will be set up to display as Date Pickers on input forms.
- The order in which the columns appear in the table will be the default order in which the UI Defaults will set them to display on a form or report.
- Defining a column as a BLOB sets the Form level UI Defaults to use APEX's declarative blob functionality.

You'll actually set up and modify UI Defaults in a later chapter, so you can see for yourselves how design decisions affect how UI Defaults are set up.

APEX and Primary Keys

APEX is set up to make best use of sequence-based surrogate primary keys of no more than two columns. While you can definitely still use APEX on table structures that use multi-column natural keys, it is far easier and you get much more out of the box if you give APEX what it likes.

I have worked with many systems over the years that implemented multi-column natural keys, and I've successfully implemented APEX systems on top of these types of data structures. However, I ended up hand coding the logic that APEX would have provided me for free had the structures used one or two column surrogate keys.

There may be no way for you to get around hand coding if you're writing systems on top of legacy data structures, but when you're designing new systems or migrating data from legacy systems into Oracle, implementing surrogate keys can save you hours of work and troubleshooting.

If you take the example of the TICKETS table, the ID for a ticket is an arbitrary piece of data used only to uniquely identify one ticket from another. Therefore, it easily fits into the realm of a surrogate primary key. Even if the spreadsheet that the helpdesk technicians currently use has IDs assigned to the ticket, you can merely load those values and start your sequence counting at a point above the highest current ticket id. The same would be true for TICKET DETAILS. Even in the USERS table, where you have a unique single column natural key (the User Name), it behooves you to implement a surrogate key to be able to take advantage of the built-in APEX code paths.

Business Logic vs. User Interface Logic

Being written in PL/SQL, APEX takes full advantage of everything that PL/SQL has to offer. The APEX development team has made thorough use of stored PL/SQL program units for their business logic, and you can take a very important lesson from them.

Although it's arguably a valid development method to prototype your business logic by first coding it as an anonymous PL/SQL block inside of APEX, it is foolish to leave it there long term. By moving it out into stored program units, you gain in many different ways.

One very important gain is in the realm of performance. Anonymous PL/SQL blocks are stored in the APEX metadata as uncompiled PL/SQL code. Each time they are required to run they must first be extracted from the APEX metadata, parsed, compiled, and then run. This carries quite an overhead if the PL/SQL in question is part of a page that gets thousands or even hundreds of thousands of hits a day. By moving that code into a stored program unit within the database, the retrieval, parse, and compile steps are all skipped and the code is run directly.

Another benefit is that of reusability. If the same logic is used in more than one place, it can simply be called instead of duplicated in two anonymous blocks. Therefore, any change to the business logic need only happen in one place. Another reusability benefit might occur if multiple systems (some being non-APEX) need access to the same business logic. When stored in a PL/SQL program unit, it doesn't matter whether the calling system is APEX, .NET, Java, or PHP. They can all use the same logic.

Finally, by moving business logic code into stored program units you gain the ability to code, debug, and test these program units outside of the restrictions of APEX, using your favorite PL/SQL coding tool instead. However, not all code needs to be moved out into the database. User interface logic that manages and manipulates items on the page, such as computations, validations, and processes, is often best kept as part of the page itself. Such logic is often so page specific and so small in footprint that the gain by moving it out to the database is not worth the extra management overhead. As a general rule of thumb, logic that controls or manipulates the UI is best placed within APEX and logic that implements business rules or controls the data is best placed in stored program units within the database.

Placement of Database Objects

The Oracle database is very flexible, allowing data from multiple schemas to be granted to and queried by other schemas, even across database links. The APEX wizards have been coded to work best when the database objects reside in a "parse as" schema assigned directly to the workspace.

The APEX wizards make heavy use of database metadata for the objects in its "parse as" schema. If you are trying to create applications against synonyms from another schema or across a database link to another database, in many cases the wizards won't be functional because the metadata for these objects is unavailable. There are some features that won't work at all, such as the management of BLOB data across database links.

In general, reports are much easier to deal with when it comes to disparate data, as you can merely supply a working query and create a report. Forms, however, become much more difficult as the insert, update, and delete logic must be coded manually instead of relying on the APEX-supplied automated DML processes.

While it's not always possible, the best practice is to create the underlying database objects in the "parse as" schema for the application. This is how you will be architecting your helpdesk system.

Translating Theory to Practice

Now that you have a reasonable understanding of the things you need to think about when designing the database objects for your system, you can translate your text-based tables into a real schema definition. Although it is very easy to take the previously described objects and attributes straight to SQL Workshop and start entering their definition, it's usually a good idea to go through the steps of creating an Entity Relationship Diagram (ERD). I often find the action of doing this can bring other design considerations to light.

There are dozens of ways to draw ERDs—from pen and paper to high-end database design tools. However, I tend to opt to take the middle grounds and use Oracle's SQL Developer Data Modeler, a robust and free tool from Oracle.

Figure 3–1 shows the results of using the Data Modeler to create the entity relationship diagram from the information in the initial definitions.

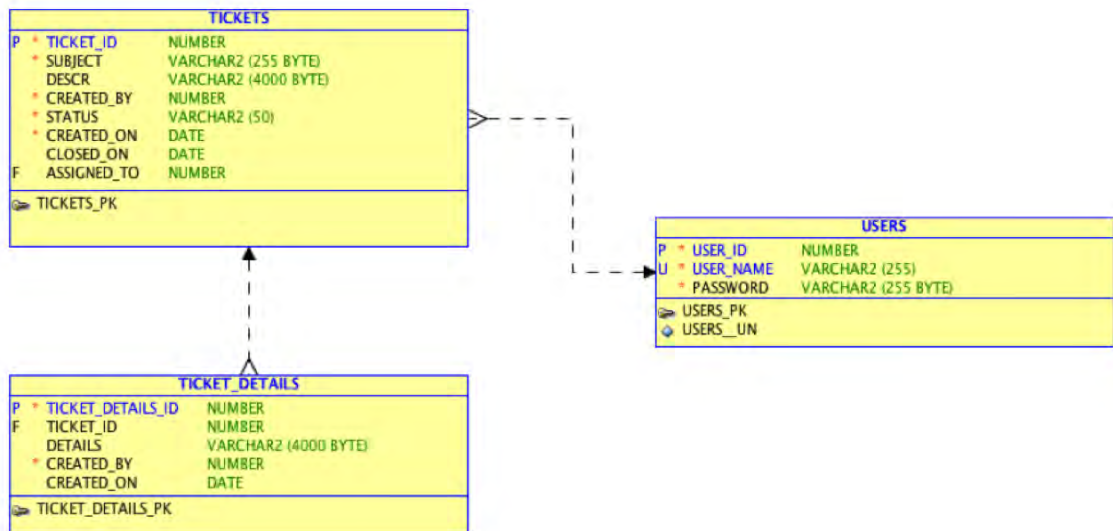


Figure 3–1. First draft of database design

The diagram shows each table having a surrogate primary key that will uniquely identify the records. As discussed in the previous section, this allows the APEX wizards to work more seamlessly and generate more complete objects.

There is a foreign key in place between TICKETS and USERS to identify the person to whom the ticket is currently assigned.

There has also been a unique constraint placed on the USER_NAME column of the USER table to make sure someone doesn't accidentally create two users with the same USER_NAME.

Although this is not likely to be the final version of the data model, it's probably complete enough for a start. Using your ERD tool, you could go ahead and generate the database object creations scripts, then upload and run them through APEX SQL Workshop's SQL Script interface. However, since your data model is so small, in the next chapter you'll use the Object Browser tool to create the objects from scratch.

Summary

Identifying the problems your APEX application is only half the battle. Good database design—and designing specifically with APEX in mind—is the key to creating a successful APEX application. Taking the time to make sure you have a solid foundation means you can take full advantage of everything APEX gives you so that there will be less work to do later.



SQL Workshop

Now that you have a graphical representation of what your underlying tables should look like, in the form of an ERD, it's time to dig in and start creating the objects. As mentioned before, you could use your ERD tool to generate the scripts, but to get you used to using the SQL Workshop, you're going to be creating these objects from scratch.

■ **Note** For this and many of the following chapters, you'll need to have downloaded the code that accompanies the book. If you have not already done so, download the code .zip file from this book's home page at www.apress.com. Once the .zip file is downloaded, unzip it to a directory where you can retrieve the files easily.

Creating Objects with the Object Browser

SQL Workshop's Object Browser is somewhat misnamed, as it not only allows you to view database objects, but also allows you to create and edit them as well. For now, you're going to skip your USERS table; you'll come back to it later in the book. Right now, you're going to focus on your TICKETS and TICKET DETAILS tables. From this point forward, you'll be following step-by-step instructions that will be interspersed with figures and discussions about what you're trying to achieve and why you're doing it the way you are. So let's get started:

1. Start by logging into your APEX workspace. You'll be presented with the workspace's home page, which, unless you've been doing other work in this workspace, will probably look a little sparse.
2. Once you're at your workspace home page, use the tabbed navigation bar across the top of the page and pull down the **SQL Workshop** sub-menu by clicking on the arrow on the right side of the tab (see Figure 4-1).
3. Click on the Object Browser option.



Figure 4–1. Navigate to the Object Browser

4. In the Object Browser, click on the **Create** button in the upper right hand corner and select **Table** from the drop-down menu.
5. The Create Table Wizard will open. The first screen (Figure 4–2) allows you to name the table and enter the details for each of the columns of the table.

The following wizard will walk you through the steps of creating a table. The first screen allows you to name the table and enter the name and details for each of the columns of the table.

Using the two arrows under the Move column, you can move the columns to what ever order you like. This affects the order in which they are defined and stored in the table.

If you run out of empty rows to enter columns into, you can use the Add Column button to add a new empty column definition row to the form.

Column Name	Type	Precision	Scale	Not Null	Move
TICKET_ID	NUMBER			<input type="checkbox"/>	▼ ▲
SUBJECT	VARCHAR2		255	<input checked="" type="checkbox"/>	▼ ▲
DESCRIPTION	VARCHAR2		4000	<input type="checkbox"/>	▼ ▲
STATUS	VARCHAR2		20	<input checked="" type="checkbox"/>	▼ ▲
ASSIGNED_TO	VARCHAR2		50	<input type="checkbox"/>	▼ ▲
CREATED_ON	DATE			<input checked="" type="checkbox"/>	▼ ▲
CLOSED_ON	DATE			<input type="checkbox"/>	▼ ▲
CREATED_BY	VARCHAR2		50	<input type="checkbox"/>	▼ ▲

Figure 4–2. Defining the table and its columns

6. Enter the details for the TICKET table as indicated in the ERD from the end of Chapter 3 and in Figure 4–2. Then click **Next**.

The next page (Figure 4–3) lets you choose how you would like the primary key to be populated and which column to use as the primary key. The four options for primary key are fairly self-explanatory, but the two in the middle are probably the most common. You’re starting from scratch and therefore don’t have any existing sequences defined in your database. By selecting “Populate from a new sequence,” APEX will create a sequence for you and will create a database trigger on the table that will populate the selected Primary Key column with the next value from the sequence, unless the field already has a value. You’re given the chance to name the sequence within this step as well. In this instance, you’ll use the default name given.

Figure 4–3. Defining the table’s primary key

7. Select the **Populated from a new sequence** radio button and after the screen changes, select **TICKET_ID (NUMBER)** for primary key. Click **Next**.
8. You’re not going to create any foreign keys in this table just yet, so simply leave the defaults and click **Next**.

The Constraints screen in Figure 4–4 allows you to add either Unique or Check constraints to the table definition. Add a constraint by defining the constraint in the Add Constraints region and clicking the Add button to add it to the list. Below the Add Constraints region are two help regions. Clicking on the gray arrow to the left of the region title will expand the help and show the columns you defined in the table and examples of how to code various check constraints.

Once you’ve clicked the Add button, the definition of the constraint is added to the list of constraints at the top of the page. You can define as many constraints on a given table as necessary. Once you’re done, simply continue on with the wizard.

9. You’re not going to create any Unique or Check constraints here, so stick with the defaults and click **Next**.

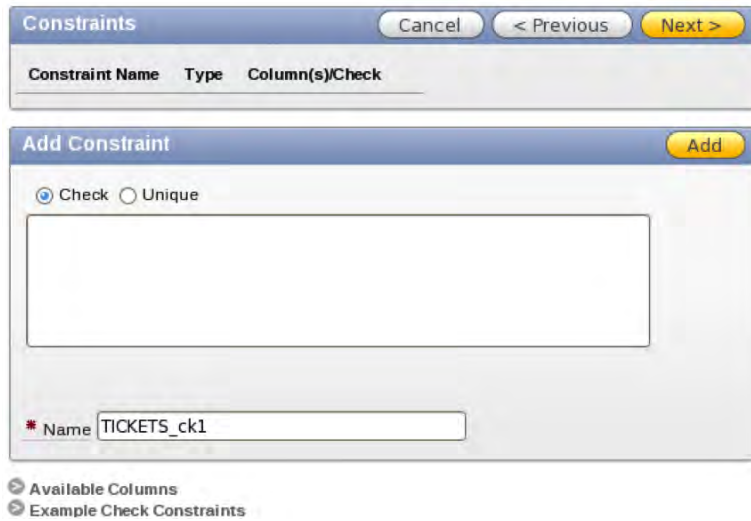


Figure 4–4. *The Constraints definition step*

The final step of the Create Table wizard shows you the code that will be executed and allows you to make any changes. For instance, the Table wizard doesn't allow you to add column level comments, but you could easily add them by editing the text displayed in the SQL region. You can see the code for the USERS table in Figure 4–5.

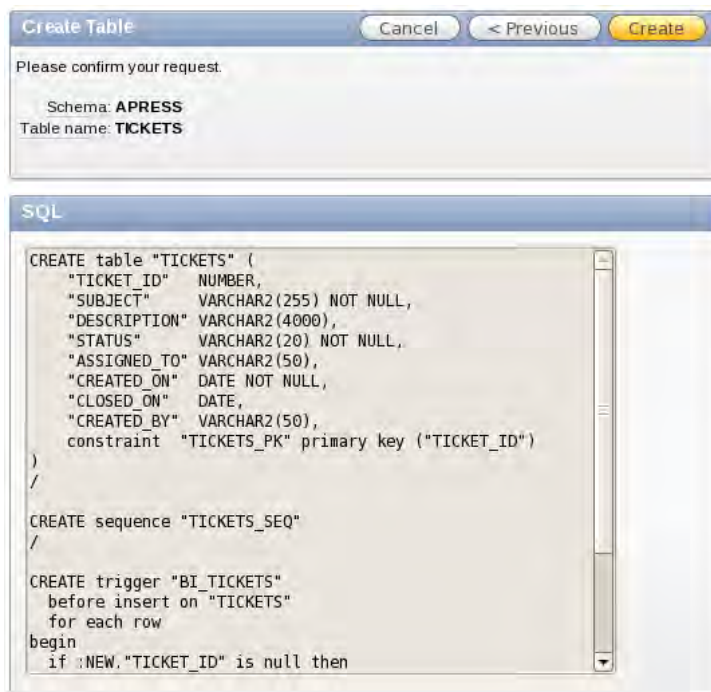


Figure 4–5. Review the Create Table wizards SQL

10. Review the text in the SQL region presented by the Create Table Wizard. Click **Create** to complete the wizard.

Once you’ve successfully completed the wizard, you’ll be taken back to the Object Browser and the definition of the TICKETS table will be displayed. Take a moment to examine the definition of the table. You should see all the columns that you defined listed. If you click on the Constraints tab across the top of the definition region, you’ll see a number of different constraints including the primary key constraint on TICKET_ID.

In the upper left hand corner of the object browser, there is a select list that defines the object type being browsed. Use this select list and choose Sequences. You’ll see that APEX created a sequence called TICKETS_SEQ that will be used fill the TICKET_ID.

Once again, use the Object Type select list and chose Triggers. You’ll see a trigger named BI_TICKETS (BI stands for “before insert”). Clicking on the Code tab above the trigger details will show the code for the trigger that is using the TICKETS_SEQ sequence to fill the TICKET_ID if it’s null. You should see code similar to the following:

```
create or replace trigger "BI_TICKETS"
  before insert on "TICKETS"
  for each row
begin
  if :NEW."TICKET_ID" is null then
    select "TICKETS_SEQ".nextval into :NEW."TICKET_ID" from dual;
  end if;
end;
```

Now that you have your Tickets table defined, let's go back and create your TICKET_DETAILS tables. This time you'll be creating a foreign key to the TICKETS table. You'll be creating the foreign key to the TICKETS table as a CASCADE DELETE. This means that if you delete the ticket, the ticket details will automatically be deleted as well.

11. Start the Create Table wizard using the **Create** button.
12. Enter the table name and column definitions based on the ERD and Figure 4–6, and click **Next**.

The next set of steps is purposefully a bit more vague than the previous ones. You should be used to using the Create Table wizard by now, but if you need a refresher, just look at the previous steps.

Column Name	Type	Precision	Scale	Not Null	Move
TICKET_DETAILS_ID	NUMBER			<input type="checkbox"/>	▼ ▲
TICKET_ID	NUMBER			<input checked="" type="checkbox"/>	▼ ▲
DETAILS	VARCHAR2		4000	<input type="checkbox"/>	▼ ▲
CREATED_BY	VARCHAR2		50	<input checked="" type="checkbox"/>	▼ ▲
CREATED_ON	DATE			<input type="checkbox"/>	▼ ▲
	- Select Datatype -				▼ ▲
	- Select Datatype -				▼ ▲
	- Select Datatype -				▼ ▲

Figure 4–6. Defining the *TICKET_DETAILS* table

13. Choose to populate the primary key from a new sequence, select **TICKET_DETAILS_ID(NUMBER)** as the primary key, and click **Next**.
14. Add a foreign key between the **TICKET_ID** in the **TICKET_DETAILS** table and the **TICKET_ID** in the **TICKETS** TABLE. Make sure that the Delete action is set to **Cascade Delete**. Your screen should look similar to that in Figure 4–7.

Foreign Keys [Cancel] [< Previous] [Next >]

Foreign Key **Columns** **Referenced Table** **Referenced Columns** **Action**

Add Foreign Key [Add]

Name:

☐ Disallow Delete
☒ Cascade Delete
☐ Set Null on Delete

Select Key Column(s):
 TICKET_DETAILS_ID
 DETAILS
 CREATED_BY
 CREATED_ON

Key Column(s):
 TICKET_ID

References Table:

Select Reference Column(s):
 SUBJECT
 DESCR
 PRIORITY
 CREATED_BY
 CREATED_ON

Referenced Column(s):
 TICKET_ID

Figure 4–7. Defining Cascade Delete foreign key for the `TICKET_ID`

Foreign Key	Columns	Referenced Table	Referenced Columns	Action
TICKET_DETAILS_FK	TICKET_ID	TICKETS	TICKET_ID	cascade

Figure 4–8. Foreign keys as defined in the table wizard.

15. Once the foreign key is defined, click **Next** (see Figure 4–8).
16. No constraints are required for this table. Click **Next**.
17. Review the SQL and click **Create** to complete the wizard.

Loading Data with the Data Workshop Utility

Now that you have your two base tables defined, you can begin working to migrate the old data into your shiny new data structure. SQL Workshop's Data Workshop utility can be used to load and unload data from an Oracle schema in a number of ways, as shown in Figure 4–9. The Data Load option allows you to choose Text Data, XML Data, and Spreadsheet Data.

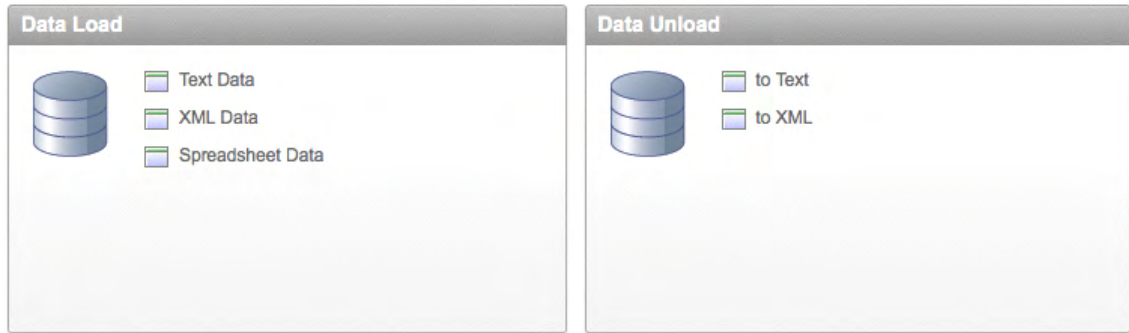


Figure 4-9. Data Load and Unload methods provided by the Data Workshop Utility

Although three separate options are presented, the Text Data and Spreadsheet Data options actually utilize the same Data Load wizard. There is little or no discernable difference in the actions of the wizard no matter which option you selected.

The third option (XML Data) allows you to load data that has been exported in Oracle's proprietary XML Data Transport format. The format looks like this:

```
<ROWSET>
<ROW>
  <USER_ID>2</USER_ID>
  <USER_NAME>DOUG</USER_NAME>
  <PASSWORD>A69856770A9AB9CBB0479573FCB3E2A5</PASSWORD>
</ROW>
<ROW>
  <USER_ID>3</USER_ID>
  <USER_NAME>DAVID</USER_NAME>
  <PASSWORD>E2E89134B8AC6E1FFC14139A6FB2C10B</PASSWORD>
</ROW>
</ROWSET>
```

In your imaginary company, your help desk technicians have been using Microsoft Excel to track tickets, so you're going to load the data using the Spreadsheet option. A quick glance at the spreadsheet that your technicians use shows you that they have two separate sheets in the Excel workbook: TICKETS and TICKET_DETAILS.

Knowing that you're going to be loading your data into pre-existing tables that already have primary and foreign keys in place, you need to be careful about how you load the data. TICKET_DETAILS depend upon the Tickets for their parentage so you need to load the TICKETS data first. Your spreadsheet should look like Figure 4-10.

	A	B	C	D	E	F	G	H
	ticket_id	subject	descr	status	assigned_to	created_on	closed_on	created_by
1	1	Cannot log into E-Mail	User called and cannot log into his MS Outlook e-mail Account.	OPEN	SCOTT	1-Jan-07	3-Jan-07	PAUL
2	2	PC will not turn on	The user's PC will not turn on when the power button is pressed.	CLOSED	JOHN	1-Jan-07		RINGO
3	3	Need more memory	User needs more memory installed	OPEN	DOUG	1-Jan-07		GEORGE
4	4	MSIE Crashed 4 times	MSIE keeps on crashing for any site	CLOSED	SCOTT	1-Jan-07		JOHN
5	5	Need to install SP2	SP2 Upgrade needed in order to be compliant	OPEN	DIMITRI	1-Jan-07		ALEX
6	6	Network drive not being mapped	X: drive not being mapped to \\corpshare	OPEN	DIMITRI	1-Jan-07		GEDDY
7	7	BSOD after rebooting	Blue Screen of Death every time system is rebooted	OPEN	DOUG	2-Jan-07		NEAL
8	8	Wireless signal not strong enough	Wi-Fi signal not as strong as it was last week	CLOSED	SCOTT	2-Jan-07	3-Jan-07	JOHN
9	9	I think I have a virus	Something is not right - PC is slow	OPEN	JOHN	2-Jan-07		ROBERT
10	10	Virus Definitions Dates	Message stating that virus updates are needed keeps appearing	CLOSED	SCOTT	2-Jan-07		JOHN
11	11	Funny smell coming from PC	There is an odd odor emanating from my PC...	OPEN	DIMITRI	3-Jan-07		JIMMY
12	12	Accidentally deleted Q2.ppt	File Q2.ppt placed in Recycle Bin; bin emptied	OPEN	JOHN	3-Jan-07		EDDIE
13	13	Several dead pixels on screen	There are at least 4 dead pixels on the display	PENDING	DOUG	3-Jan-07		ALEX
14	14	Smartphone will not sync with Outlook	Motorola Q does not sync with Outlook contacts and calendar events	OPEN	SCOTT	3-Jan-07		MICHAEL
15	15	Getting Out of Memory errors	Same Out of Memory error occurs when Office starts	PENDING	JOHN	3-Jan-07		DAVID
16	16	VPN Client Install Issues	Cannot install VPN client - installer errors out each time	OPEN	DOUG	4-Jan-07		JACKIE
17	17	Mouse is not working	Mouse does not move the pointer anymore	OPEN	DIMITRI	4-Jan-07		TITO
18	18	Speakers are too soft	Cannot get good quality of sound from built-in speakers	OPEN	SCOTT	4-Jan-07		JERMAINE
19	19	Keyboard busted	None of the keys work (I had to use someone else's PC to enter this)	PENDING	JOHN	5-Jan-07		MICHAEL
20	20	Disk is Full	No more space error keep coming up	OPEN	DOUG	5-Jan-07		MARLON

Figure 4–10. Spreadsheet data from the TICKETS tab of your Excel workbook

Now that you have the TICKET data in the clipboard, you can switch back to APEX and use the Data Load Wizard to insert this data into your TICKETS table. Here are the steps to follow to load your data from spreadsheet into database:

1. Locate the **helpdesk_spreadsheet.xls** file where you downloaded the supporting files for this book and open it with Microsoft Excel. Once it's open, navigate to the TICKETS tab. You'll notice that you have a row for each TICKET and a header row that contains the column headings for each of the columns.
2. Select all the data, including the column headings, and copy it to the clipboard.
3. Switch back to your web browser and, using the pull-down menu on the SQL Workshop tab, select **Data Workshop**.
4. Under the Data Load region, click **Spreadsheet Data**. You should see the Load Data dialog in Figure 4–11.
5. In the wizard, select **Existing table** for Load To and **Copy and paste** for Load From and click **Next**.

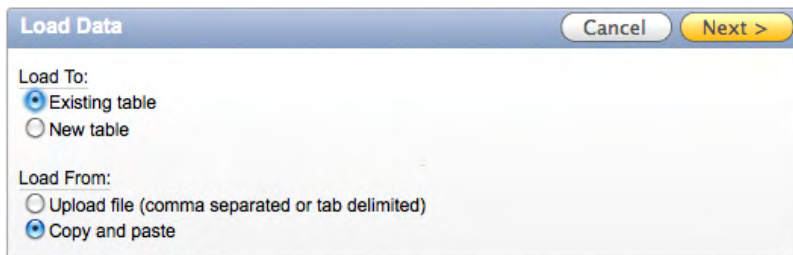


Figure 4–11. Preparing to copy and paste the spreadsheet data and load it into the existing USERS table

6. Select your parse as **schema** from the Schema select list and click **Next**. This will be the same schema in which you created your tables in the Object Browser.
7. Enter **TICKETS** for the Table Name, as shown in Figure 4–12, and click **Next**. This is the table into which you're going to be loading the TICKET data.

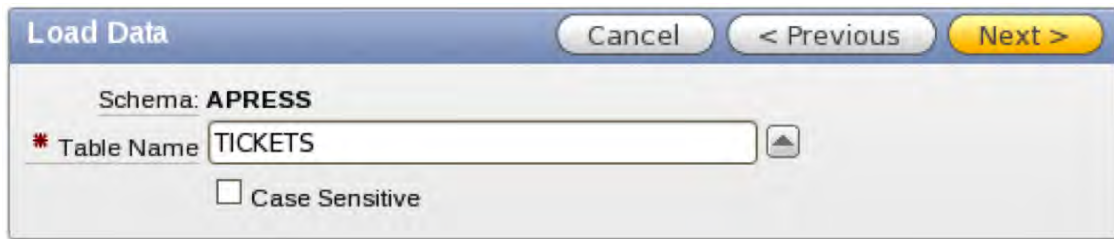


Figure 4-12. Enter the name of the table into which you're going to load the data.

8. Paste the data that you copied to the clipboard in step 2 into the Data Text area and ensure that **First row contains column names** box is checked, as in Figure 4-13, and click Next.

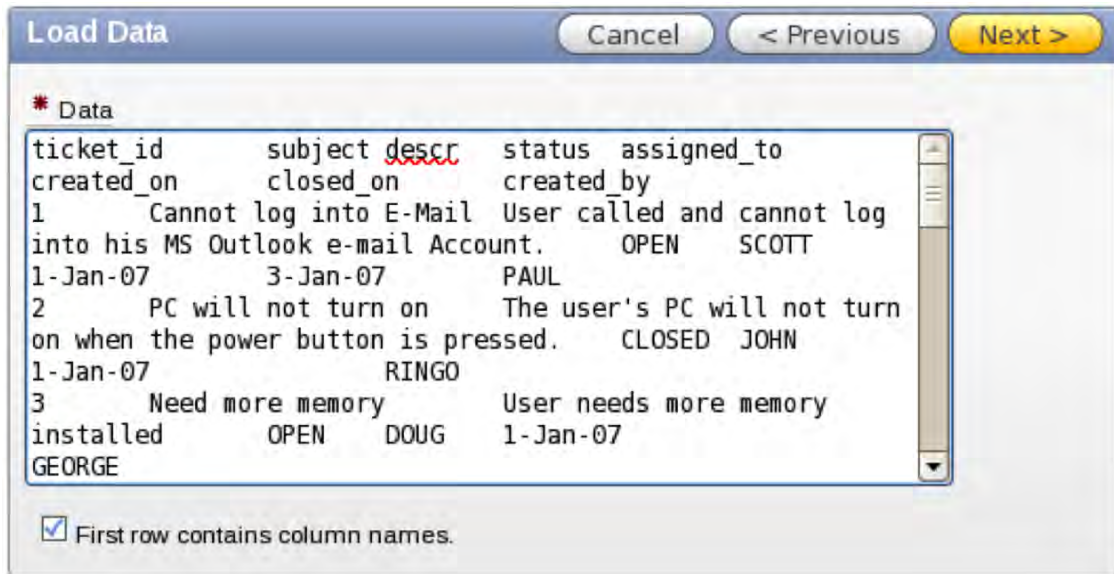


Figure 4-13. Pasting the spreadsheet data into the Data Text box

Once you click Next, APEX will parse the data you've pasted in and do its best to match the column names in the first row of the spreadsheet data to the column names of the table you're loading the data into. On the next screen, you'll be presented with the column-mapping screen so that you can check its accuracy and, if necessary, make some alterations and corrections.

APEX is very good about matching column names as defined in the spreadsheet with those that have the same name in the table. However, if the names differ, it won't try to guess but instead leaves the mapping to you.

If you scroll to the right, you'll see that APEX didn't map the e-mail address or the phone number in your example to a table column. You'll need to map these data elements to the correct table columns manually. You can do that using the drop-downs shown in Figure 4-14.

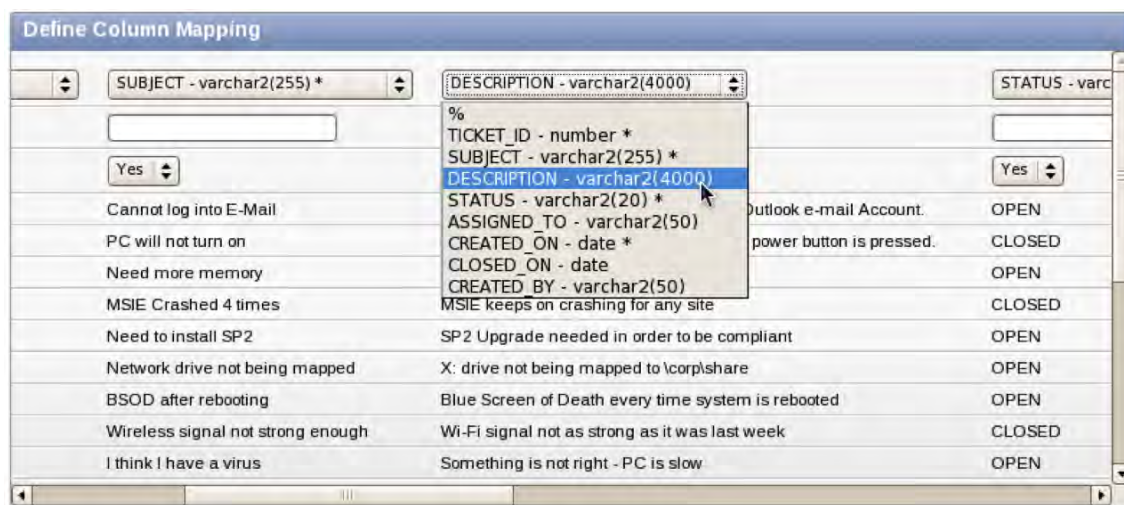


Figure 4-14. Manually mapping the data columns to the table

9. Scroll to the right in the Define Column Mapping region and map the third column to DESCRIPTION using the select list above the column data. Examine the other mappings to make sure that APEX made the correct selections.
10. When you're sure all the mappings are correct, click the **Load Data** button to load the data into the USERS table.

After the data is loaded, you'll be presented the Spreadsheet Repository screen in Figure 4-15. That screen shows you that 20 rows were loaded into the database and zero errors occurred during loading.

Details	Imported By	Imported On	Type	Schema	Table	Succeeded	Failed
<input type="checkbox"/>	APRESS	1 seconds ago	Spreadsheet Import	APRESS	<u>TICKETS</u>	20	0
							1-1

Figure 4-15. Data has now been loaded into the users table

If you navigate to the Object Browser, select your TICKETS table, and look at the data in that table, you can see that the records that were in your spreadsheet have now been loaded into the database. To finish the job, you'll need to load the data for the TICKET_DETAILS. Here's what to do:

18. Navigate to the Data Workshop and click on the **Spreadsheet Data** link in the Data Load region and click **Next**.
11. In the wizard, select **Existing Table** for Load To and **Copy and paste** for Load From and click **Next**.
12. Select your parse as **schema** from the Schema select list and click **Next**. This will be the same schema in which you created your tables in the Object Browser.
13. Enter **TICKET_DETAILS** for the table name and click **Next**.

14. In Microsoft Excel, navigate to the `TICKET_DETAILS` tab and copy all the data, including the column headings, in that spreadsheet to the clipboard.
15. In your browser, paste the data that you copied to the clipboard into the Data Text area and ensure that **First row contains column names** is checked and click **Next**.
16. Review the mappings made by APEX in the Define Column Mapping region. This time it should have mapped everything correctly. Click **Load Data** to complete the data load. The summary should say that 22 records were loaded into the `TICKET_DETAILS` table with zero errors.

You now have both of the main tables created and loaded with the legacy data. This alone is enough to start developing an application, but you're not quite ready to begin yet.

Creating a Lookup Table

Have a look at the definitions and data of the tables you just created. They are basically just mirror images of the spreadsheet tabs the technicians were using before. If you examine the data closely, you'll notice that there are still some areas where the data is still not quite normalized the best that it could be.

For instance, in the `TICKETS` table, you'll notice that the `STATUS` column only has three values—`OPEN`, `CLOSED`, and `PENDING`—which repeat over and over. The data values in this column indicate that it's a perfect candidate for creating a lookup table. While it's tempting to go off and create the table manually with the Create Table wizard and then manually migrate the data, APEX can actually create lookup tables—complete with their own sequence, trigger, and foreign key—and modify the original table so that it points to the new lookup table, all without you writing a line of code. Here's how:

1. Navigate to the Object Browser and select the `TICKETS` table in the Object List on the left hand side of the screen. You should see results similar to those in Figure 4–16.
2. Make sure that the **Table** tab is selected.
3. Below the Tab bar is a set of button-like links. Click the **Create Lookup Table** button, as shown by the mouse arrow in Figure 4–16; it starts the Create Lookup Table wizard.

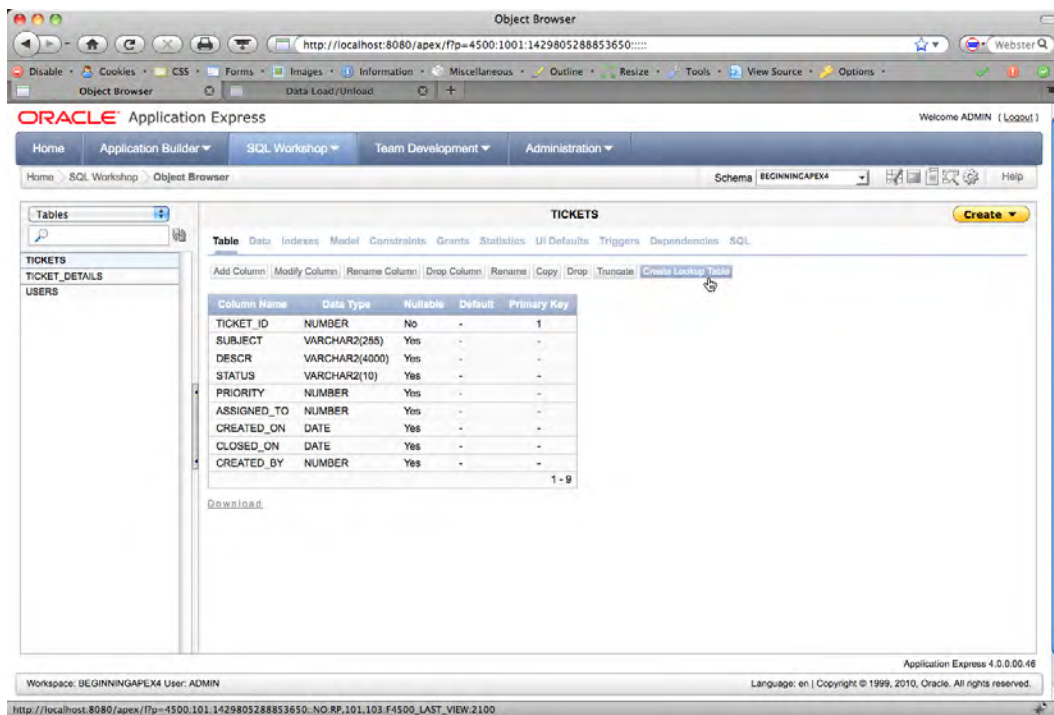


Figure 4–16. Clicking the Create Lookup Table button starts the Lookup Table wizard

The first page of the Create Lookup Table wizard (Figure 4–17) gives you the option to show only VARCHAR column types or all column types. It defaults to VARCHAR because it is the most likely to be the candidate for lookup tables. Looking at the columns that are presented in the wizard, you see that one of the VARCHAR columns is your STATUS column.

4. Select **Status** as the column from which you want to create the lookup table and click **Next**.

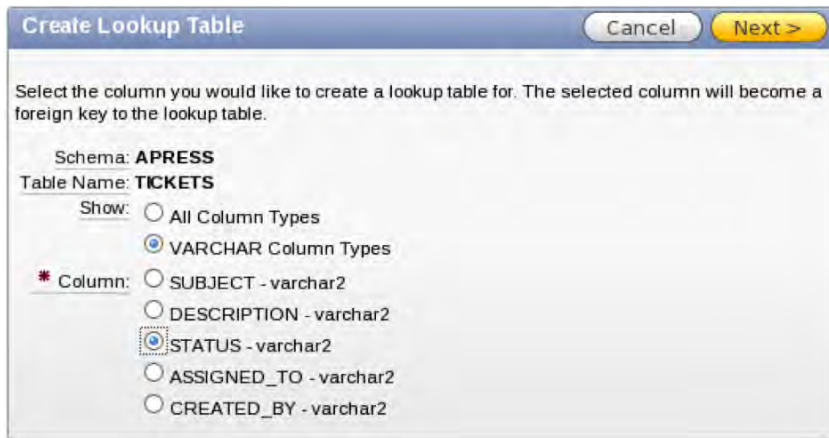


Figure 4–17. Selecting the Status column as the source of your lookup table

5. On this screen you're able to name your lookup table and the sequence that will be related to it. APEX has chosen a reasonable name for your new table and sequence so take the defaults and click **Next**.
6. The final screen of the wizard (Figure 4–18) provides you with information about the choices made and the action that is about to be performed. It's easy to miss the SQL Syntax link just below the wizard region. Click on the **SQL Syntax** link to show the SQL.

Examining the SQL will show the steps that will be taken to create the new lookup table, associated sequence, and trigger; insert the data into the table; and update the data in the originating table so that it references your new lookup table. That's quite a lot of work saved.

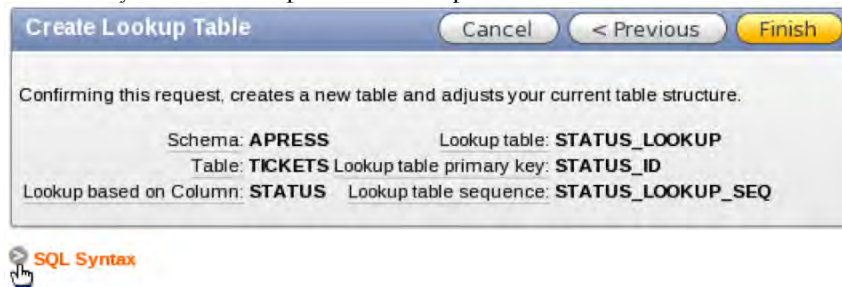


Figure 4–18. Clicking the SQL Syntax link will show the SQL about to be executed.

7. Click **Finish** to complete the wizard. You'll be taken back to the Object Browser. The STATUS_LOOKUP table will be highlighted and its details shown.

Use the Object Browser to examine the objects that were created by the wizard.

Loading and Running SQL Scripts

The SQL Scripts tool of SQL Workshop allows you to create, upload, manage, and run SQL Scripts. These scripts are similar to SQL*PLUS scripts in many ways. However, if you use scripts written for SQL*PLUS, any SQL*PLUS-specific syntax will be ignored by APEX.

Once a script is created or loaded, it is moved into the Script Repository where it will remain until you decide to remove it. From the script repository, you can decide to edit or run the script. Once you run a script, APEX stores the results for you to view later. For example, you can come back later to review it for possible error messages.

You're going to load and run a script that will modify your underlying data just a bit, and here's why.

In the real world, the spreadsheet you received from your help desk team would have current dates and data in it; however, the ticket dates in the spreadsheet that is downloaded with the .zip file accompanying this book are very likely not current. This would cause you to have to search back in history for the tickets if you were searching by date. This script will update these dates so that they're recent.

Another thing that you need to take into consideration is that you loaded a bunch of data into your tables that already had IDs assigned to them. Because the IDs were loaded with the data, you didn't use your database sequences. Therefore, your sequences are out of synch with your data. You need to drop and recreate your sequences so that the next sequence number is greater than the largest ID used in the associated table.

You're also going to alter the Before Insert trigger that was automatically created on the TICKETS table so that it automatically fills in the CREATED_ON column.

Lastly, you're going to create a function that, when passed a status name such as OPEN, will pass you back the ID for that status. This function will be used in a number of places, as you can't guarantee you know the ID value of a given status. Therefore, this function is the only safe way to get the associated ID for a given status.

When you're in any of the SQL Workshop tools, there is an icon menu in the upper right corner of the screen that provides quick links to each of the other tools. In Figure 4–19, the hand-shaped pointer is pointing to the SQL Scripts icon.

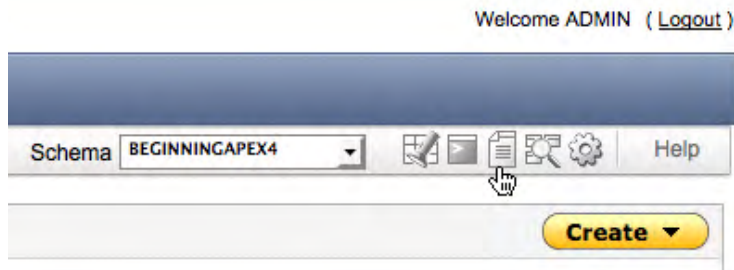


Figure 4–19. The icon based Quick Menu in SQL Workshop

Here's what to do to create the Lookup function to return status ID codes:

1. Click on the **SQL Scripts** icon in the SQL Workshop's Quick Menu. If you're not already in the SQL Workshop, use the pull-down menu from the SQL Workshop tab and chose **SQL Scripts**.
2. Click the **Upload** button in the upper right section of the screen.
3. Click on **Browse** to search for the SQL file to upload.

4. In the pop-up file finder window, locate and select the **ch4_schema_changes.sql** file and click **Upload**. You do not need to give the script a name; it will default to the name of the script as it appears at the OS level.

Once the file has been uploaded, you'll be presented with the SQL Scripts report showing the script that you just uploaded. From this point, you can either edit or run the script. If you want to see what the script contains, feel free to edit it. You are able to run the script from the edit screen as well.

Run the script by clicking either the Run button (if you're editing the script) or the Run icon (if you're still viewing the SQL Script report).

5. As shown in Figure 4–20, you'll be asked to make a selection between Run in Batch and Run Now. Select **Run Now**.

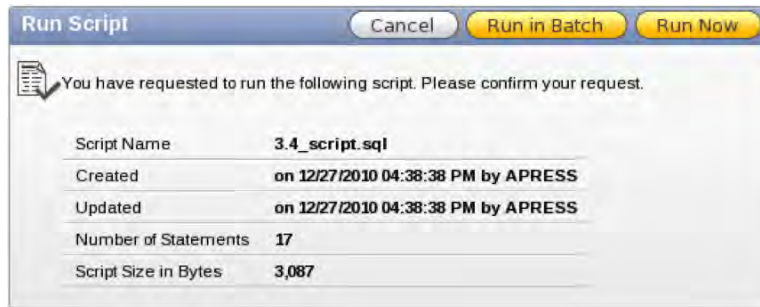


Figure 4–20. Choose whether to Run in Batch or Run Now

The script will be run and you will immediately be taken to the Manage Script Results page. You'll most likely see that your script status is COMPLETE.

6. Click on the **View Results** icon at the very right of the report row to see the results of the script. Figure 4–21 shows the button to click.

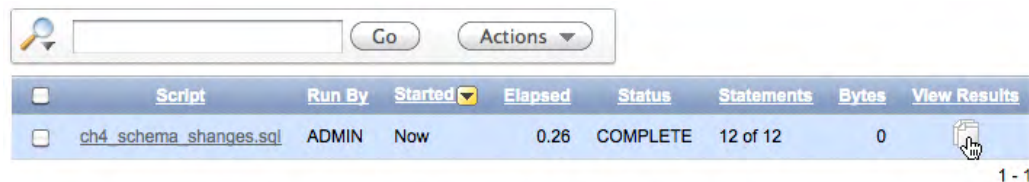
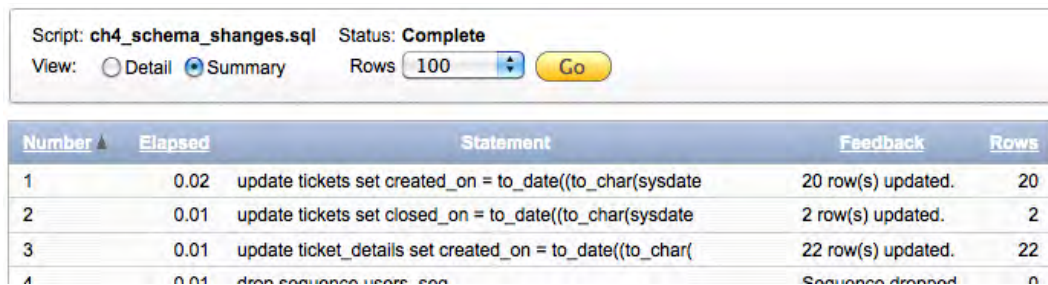


Figure 4–21. Click the View Results icon to view the results of running the script.

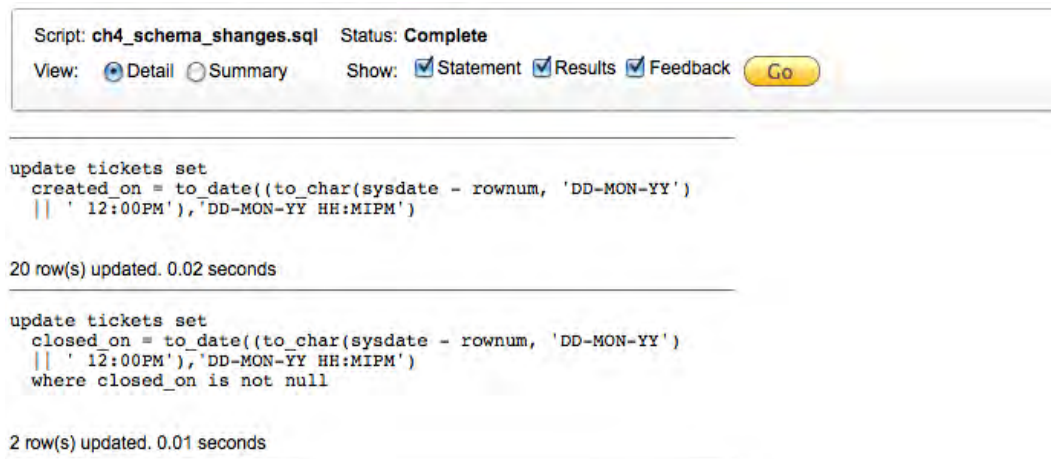
The View Results page allows you to see what happened when the script ran. The default view shows you an overview by displaying the first 50 or so characters of each statement along with some brief feedback and the number of rows affected by the statement. Figure 4–22 shows the results from your run of the script.



Number	Elapsed	Statement	Feedback	Rows
1	0.02	update tickets set created_on = to_date((to_char(sysdate	20 row(s) updated.	20
2	0.01	update tickets set closed_on = to_date((to_char(sysdate	2 row(s) updated.	2
3	0.01	update ticket_details set created_on = to_date((to_char(22 row(s) updated.	22

Figure 4–22. The Summary view of the script results.

You can, however, get more detailed feedback by changing the report view to Detailed. Doing so will give you far more insight, especially if you have a script that has errors during execution. Figure 4–23 shows a detailed view.



Script	Status	View	Show	Go
ch4_schema_shanges.sql	Complete	Detail	Statement, Results, Feedback	Go

<pre>update tickets set created_on = to_date((to_char(sysdate - rownum, 'DD-MON-YY') ' 12:00PM'),'DD-MON-YY HH:MIPM')</pre>
20 row(s) updated. 0.02 seconds
<pre>update tickets set closed_on = to_date((to_char(sysdate - rownum, 'DD-MON-YY') ' 12:00PM'),'DD-MON-YY HH:MIPM') where closed_on is not null</pre>
2 row(s) updated. 0.01 seconds

Figure 4–23. The Detail view of the script results

In either view you can quickly see if the script encountered any errors by scrolling to the bottom of the page and looking at the report footer where the report will display the total number of statements processed, the number of those that were successful, and the number that generated errors. Figure 4–24 shows the number of statements process from your run of the script.

```
Statements Processed 12
Successful          12
With Errors         0
```

Figure 4–24. In the footer of either report is the success summary for the script.

User Interface Defaults

Before you start to write your application, one last thing you can do that will make your life easier along the way is to create some User Interface (UI) Defaults. This, in my opinion, is one of the most underutilized features of APEX.

UI Defaults allow you to customize the default display attributes for tables, views, and their columns. They can be used to control many properties including alignment, searchability, display sequence, what type of item is created for a column, default values, and many more.

For instance, when you're creating a new form or report via a wizard (which is most of the time), APEX will ask you if you wish to use UI Defaults. If you select yes and defaults are available, APEX will apply them to the appropriate regions or items based on the tables or columns for which the attributes are defined. UI Defaults are divided into two categories: Attribute Dictionary and Table Dictionary.

The Attribute Dictionary allows you to create more generic UI Defaults based on attribute names. Consider this a more macro level definition.

Let's say you create an Attribute level default for any attribute named PHONE_NUMBER. If a column named PHONE_NUMBER appeared in a table and didn't have Table Dictionary default assigned, the Attribute Dictionary default would take effect.

Dictionary Attribute definitions can also be assigned synonyms, allowing more than one attribute name to share the same actual definition. So, for instance, you could create the synonyms PHONE, TELEPHONE, PHONENUMBER, etc. for the original PHONE_NUMBER definition. If the wizard ran into a column with any of those names, it would apply the PHONE_NUMBER defaults to the APEX item that is created.

The Table Dictionary allows you to define defaults for a specific table or column and those defaults will only be applied to APEX regions or items created for those specific items.

Some things to note about UI Defaults:

- Table Dictionary defaults always override those of Attribute Dictionary defaults.
- When an item is created using UI Defaults, there is no relationship established with the UI Default. Therefore, if you later change the definition of the UI Default, the changes will not be propagated to previously created items.
- Items created before UI Defaults have been established will not inherit properties of the UI Default.
- Developers can choose not to use UI Defaults, and even if they are used, can override them after the component is created.

Having said that, UI Defaults do help ensure consistency across your application and make your job much easier as a developer.

Defining UI Defaults for Your Tables

UI Defaults can be managed either from the SQL Workshop's Object Browser or from SQL Workshop's Utilities page. Here's what to do:

1. Navigate to SQL Workshop's UI Defaults page by clicking on the arrow on the SQL Workshop tab and selecting **UI Defaults** from the drop-down menu.

You'll be taken to the UI Defaults dashboard where things will likely look pretty sparse. This is because you haven't actually created any UI Defaults yet. The first step to creating UI Defaults is to synchronize the Table Dictionary with the database so it knows what tables are in your schema.

2. Click the **Manage Table Dictionary** button and then click the **Synchronize** button on the screen that appears.

This will initiate the Synchronization wizard. This wizard shows you the number of tables with defaults defined, and the number without. In your case, you should have zero tables with defaults and four tables without.

3. Click the **Synchronize Defaults** button to begin the synchronization with the database. This may take a little bit of time.

Once the Table Dictionary is synchronized with the definitions in the database, you're presented with the report in Figure 4–25 that shows you each table that now has base UI Defaults.



Object Name	Type	Defaults Exist
STATUS_LOOKUP	TABLE	Yes
TICKETS	TABLE	Yes
TICKET_DETAILS	TABLE	Yes

1 - 3

Figure 4–25. List of tables with UI Defaults defined

You can now go and view or edit the UI Defaults for each of these tables. Start by viewing the UI Defaults for the TICKETS table.

4. Click on the **TICKETS** link in the report. You should see the results in Figure 4–26.

On the page in Figure 4–26 you can see an overview of the UI Defaults for the TICKETS table. In the upper portion of the report you see the Table level definitions, including what the Form and Report regions based on this table will be called. In the lower portion is a list of the table's columns, the labels that will be used, how they will be aligned when used in a report, whether they will be displayed in a report or form, whether their Required attribute will be set in a form, and whether or not they have any help text.

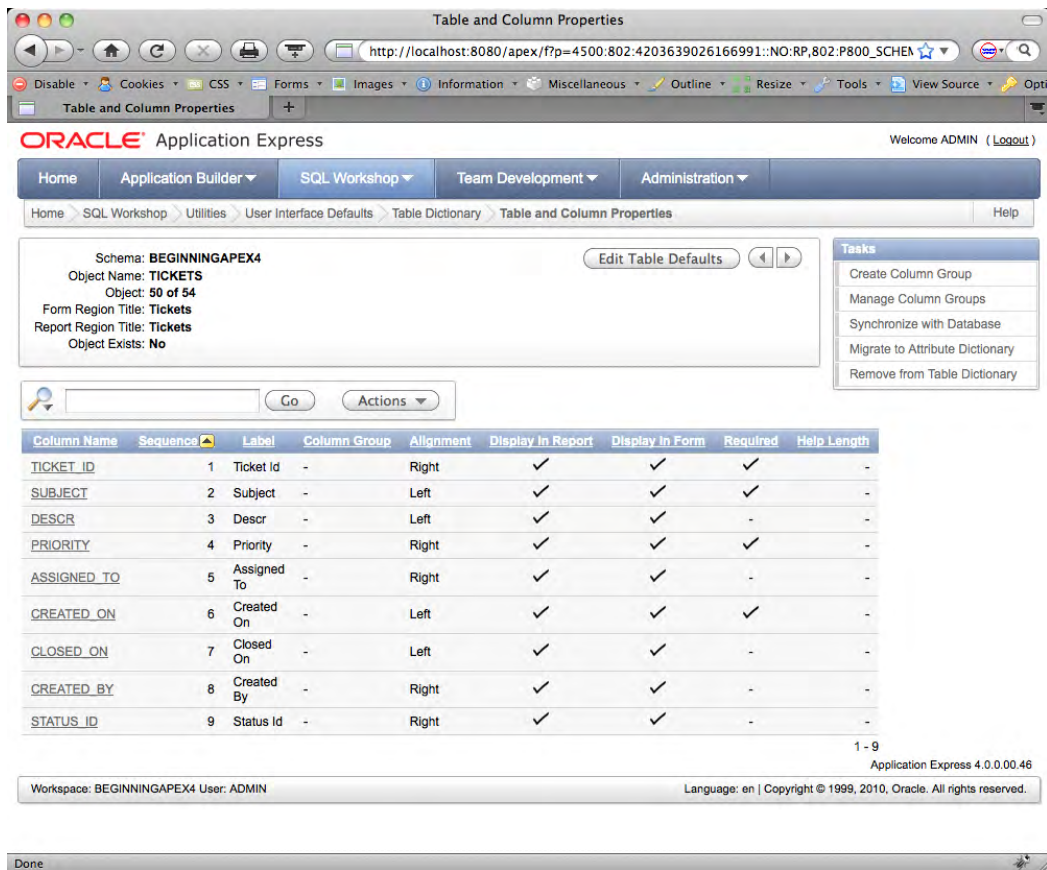


Figure 4-26. The Table and Column UI Defaults overview

Next you'll edit both the Table level and Column level attributes.

- Click on the **Edit Table Defaults** button in the upper portion of the report. This will allow you to edit how Form and Report regions based on this table will be named.
- Enter **Manage Tickets** for the Form Region Title, leave the Report Region Title as it is, and click **Apply Changes**.

Clicking on any of the column names will take you to a page that allows you to set UI Defaults for that specific column. As you peruse the column UI Defaults, you'll notice that several things have been set for you, including the **REQUIRED** attribute. When APEX synchronized with the database, it saw that certain fields were marked as **NOT NULL** at the database level and translated those constraints into UI Defaults for you.

APEX also makes some decisions based on the column's data type, such as how to align the column when it's displayed in a report. Use the following information to alter the UI Defaults for the indicated columns:

Column: SUBJECT

Label: Subject

Help Text: A brief title for the issue.

Column: DESCR

Label: Description

Help Text: Describes the ticket in detail. Please be as complete as you can.

Resizable: YES

Width: 50

Height: 5

Column: STATUS_ID

Label: Status

If you wish, you can go ahead and set the UI Defaults for any of the other columns and/or tables. Just remember, what you do now will effect what the wizards create for you later, so if something doesn't look exactly like what we're showing in the book, check what you set for UI Defaults.

Summary

While SQL Workshop may not measure up to some of the more popular GUI tools, it certainly has the power to do most things you'll need to do with relation to creating and managing tables and data. You've also seen that SQL Workshop has a few built-in but hidden gems like the Lookup Table wizard. Finally, among the many useful utilities is the UI defaults manager, making your job as a developer just a little bit easier.

Sure, you haven't covered SQL Workshop in its entirety, but you've definitely gained a fair amount of insight as to what it's capable one. You'll be using SQL Workshop for a number of other things throughout the book, but don't wait. Go poke around into some of the dark nooks and crannies and see what you find!



Application and Navigation

With some basic data created, you can now create the shell for your application. APEX provides a wizard for creating applications. Inside, several options are available to assist with generating a starting application. Based on how much prior planning has been done, the result of running the initial application wizard may vary. You will start this chapter by walking through the steps of the wizard, highlighting the most common features.

For your example application, you are going to create the most basic shell of the application with only one page. In other scenarios, you could create an initial draft of all of your pages. In order to illustrate the individual wizards for creating pages, you will cover them with more detail in later chapters.

After the application has been created, you will add shared components to your example application. Shared components are items and structures that are common across all of the pages in the application. You will prepare breadcrumbs, lists, and Lists of Values (LOVs) for use; you'll also learn how the Page Zero concept works. By the end of the chapter, you will have some basic components for the application and a starting outline for the remaining pages.

The Create Application Wizard

Applications in APEX are created through application imports, copying an existing application, or running the Application wizard. The Application wizard is the first step in creating an application from scratch. We'll first have a look at the Application wizard and its various options before we actually walk through the process or creating our Help Desk application.

To begin, navigate to the Application Builder in APEX. This can be done from the APEX home page either by clicking on the Application Builder menu item or the Application Builder icon shown in Figure 5-1. The Application Builder will show a list of the current applications. At the top of the list of applications, there is a highlighted button entitled Create, shown in Figure 5-2. Click the button and the wizard will start. You will be presented with three choices for application type: database, websheet, and sample application. The Application Builder will quickly become very familiar to you when working with APEX. Because of this, the shortcut menu in Figure 5-3 is also available to assist with quick navigation even when in other sections of APEX.



Figure 5–1. The Application Builder icon on the APEX home page

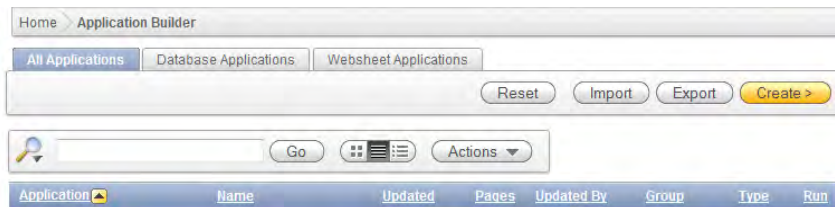


Figure 5–2. The Create button



Figure 5–3. The shortcut to creating an application

Sample Applications

If this is a new workspace, there may or may not be a sample application that was created automatically when the workspace was provisioned. The automatic installation of sample applications is a feature setting that is configurable by the APEX administrator. If a sample application is not installed, you can install one manually by selecting the Sample Applications icon on the first step of the Create Application wizard, as shown in Figure 5–4. You'll then see the dialog in Figure 5–5. The dialog lists several sample applications. Selecting the Install link on the right and following the default prompts for schema selection will automatically install a sample application and the database objects that support it. If a sample application is already installed, you'll see options for removal and re-install (intended for damage or repair) as shown in Figure 5–6. The sample applications are good learning tools for seeing a variety of features implemented very quickly.

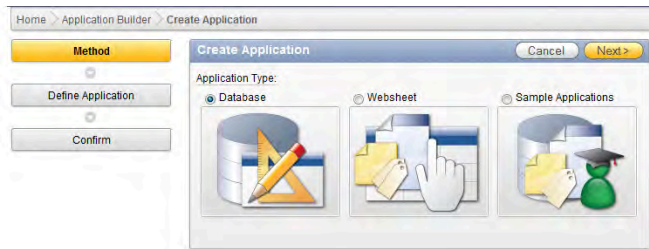


Figure 5-4. Choosing the type of application



Figure 5-5. List of sample applications

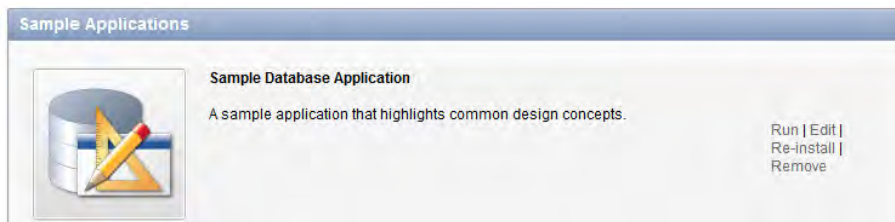


Figure 5-6. Re-install and Remove buttons for an installed sample application

Websheet Applications

You will cover the websheet application features in a separate chapter. The starting point for creating a websheet application is the same as for a database application. One of the primary differences is the creation of predefined database objects that support the websheet applications. Another difference is the segregation in the application builder between database applications and websheet applications in the form of tabs (seen in Figure 5-7) that quickly display the different types of applications separately.

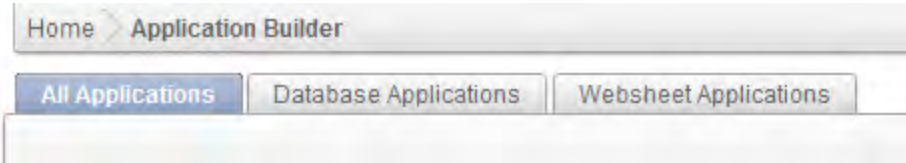


Figure 5–7. Choosing between database and websheet applications

Database Applications from Spreadsheets

When selecting a database application from the wizard, you are quickly faced with the question: Where is your data coming from? Figure 5–8 shows the dialog that you’ll see. One option is to use a spreadsheet of data as the basis for a table. The Create Application wizard will provide steps for loading data from a single table to create an application at the same time from that data. The application will be very simple, using a report and form combination such as in Figure 5–9. Creating a database application from a spreadsheet is a fast and easy way to get from a single page spreadsheet to a working online application that can be expanded upon with additional tools and functionality.

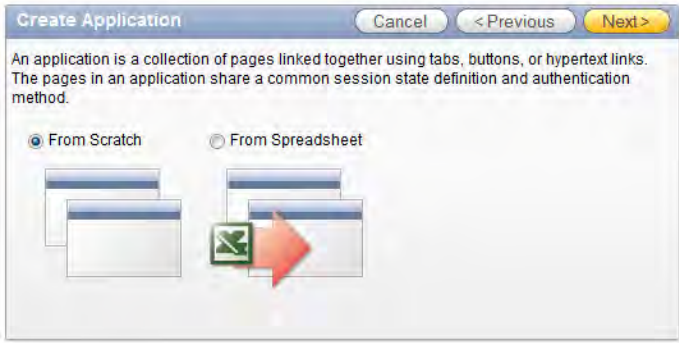


Figure 5–8. The From Scratch and From Spreadsheet options

Page	Name	Updated	Page Type	Group	Lock	Run
1	Report Page	9 minutes ago	Home	Unassigned		
2	Insert Form	9 minutes ago	DML Form	Unassigned		
3	Update Form	9 minutes ago	DML Form	Unassigned		
4	Success Page	9 minutes ago	Static HTML	Unassigned		
101	Login	9 minutes ago	Login	Unassigned		

1 - 5

Figure 5–9. The application page from a spreadsheet application

Database Applications from Scratch

When you create a database application from scratch, the wizard offers many interesting options. You can create any number of pages and link pages to different tables of data. Additional steps give advanced options that, when planned for, are very powerful. Creating an application from scratch is the method you will be using in your ticketing application exercise.

Naming the Application

After selecting the From Scratch option, you are prompted for an application name and ID, as in Figure 5–10. The name value is what will be used to identify the application inside the builder and is used as the title of the application. The application ID, although automatically created, can be manually selected here. However, there is no need to modify the application ID. Simply accept the ID that is given. Leave the last setting, the schema setting, alone. This setting exists for workspaces that have been granted access to more than one database schema. You won't be granting such access in your example.

As mentioned, you're creating an application from scratch in your example. Note that Figure 5–10 shows a second option involving creating an application from an existing design model. Use that option only when you've previously saved a series of settings after having created an earlier application. Figure 5–11 shows the prompt for a design model that you'll get if you ever decide to use the design model approach.

Figure 5–10. *Entering the application name and ID*

Figure 5–11. *Application design models option*

Laying Out Pages

The next step in the wizard is to lay out pages for the application. The wizard requires at least one page to be created. Figure 5–12 shows that you have the option to create as many pages as desired. Figure 5–12 also shows that pages have types. Different types of pages call for different types of information. Add a report page and you'll be prompted for a table name or query upon which to base the report; see Figure 5–13 for an example of such a prompt.

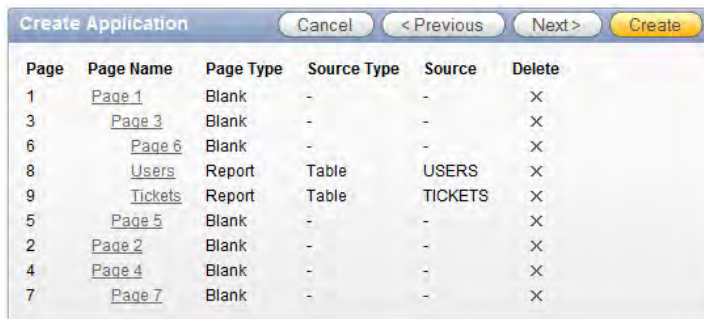


Figure 5–12. Multiple pages defined in the Application Creation wizard

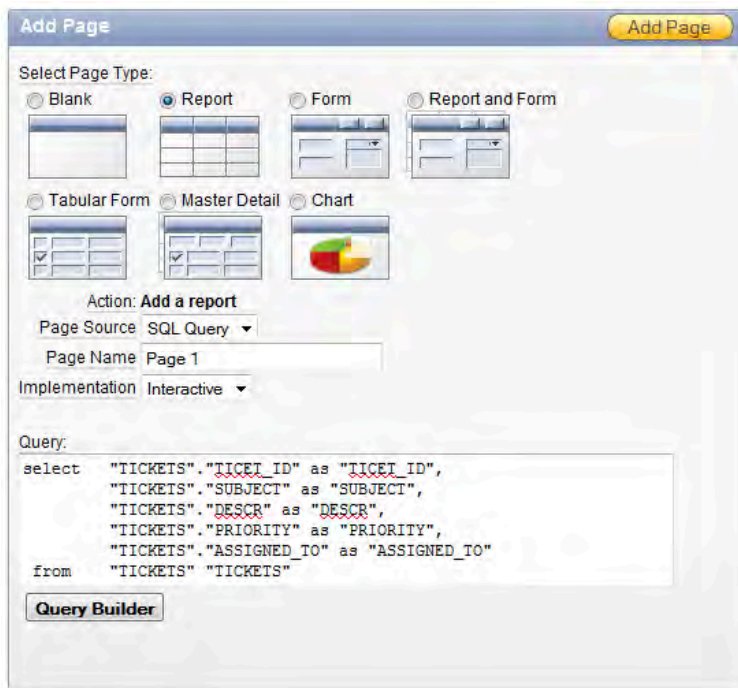


Figure 5–13. Report page definition with SQL Query prompts

Choosing an Approach to Tabs

Tabs are a common navigational structure for web applications. They provide an intuitive interface for switching subjects or general areas in an application. Three options are available:

- **No Tabs:** This is a basic page style where no tabs will be generated by the wizard, and no tabs will be displayed by the page template. This is often selected for small applications or applications where navigation is managed by a different method such as lists, buttons, or other template constructs.
- **One Level Tabs:** This is the most common style of tab layout; it's useful for small to mid-sized applications where functionality needs to be separated yet easily accessible. This is also the easiest type of tab style to manage.
- **Two Level Tabs:** The construction of the two level tab uses a parent tab construct and breaks the standard tabs into tab sets. It is similar to having a controlling tab

APEX supports up to two level tabs in the display templates provided, and the wizard will build the shared components for the tab set as part of the wizard. Figure 5–14 shows the page from which you choose a tabbing approach. If you know your application's page outline and can lay it out during the creation of the application, the wizard will do a most of the tab setups. Designing the page at creation time can be a big time saver if the application design calls for a significant number of tabs. In any case, you can create and modify the shared component after the initial run of the application creation wizard.

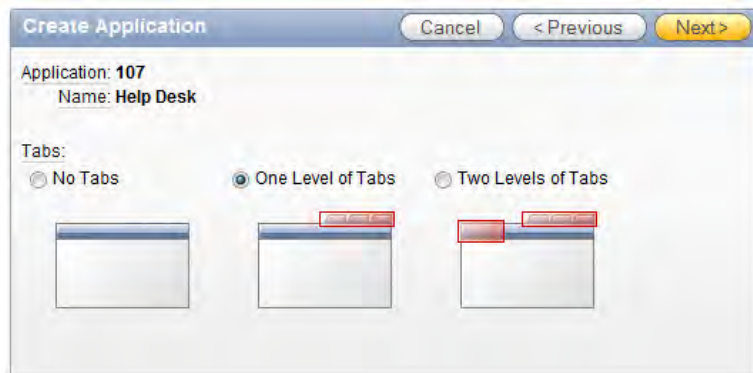


Figure 5–14. Application tab choices

Copying Shared Components

The Application Creation wizard contains a step (shown in Figure 5–15) in the process that allows shared components that have been set up in other applications to be copied during creation of a new application. This allows for easy generation of new applications when a standard has been defined. Copying shared components is not an advanced procedure, but it does lend itself to a controlled and mature development process. This feature step in the wizard is a convenience as the same objects can be copied in other ways after the application has been created.

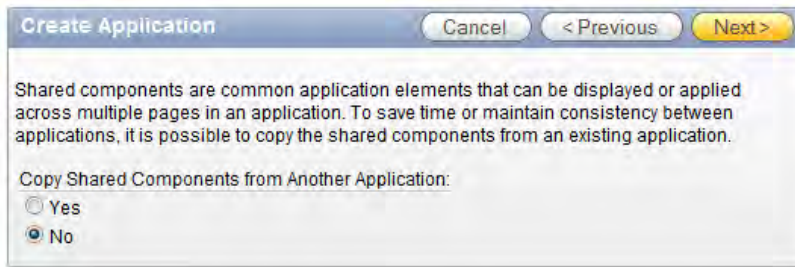


Figure 5–15. Choosing whether to copy shared components

Selecting an Authentication Method

With every application, you need to make a choice about authentication, even if that choice is no authentication. This topic will be discussed further in Chapter 9. By default, the APEX Application Creation wizard provides three options for authentication.

- **Application Express:** Users and passwords are local to the APEX workspace. These users are managed in the same way the developer accounts are inside the APEX workspace. These users only work inside the current workspace.
- **No Authentication:** This is like a public web site. The users will not be prompted for any type of authentication. This is useful for informational applications where the question of “Who are you” is not important.
- **Database Account:** This option uses the Oracle Database schema user names and passwords for credentials. Some organizations use this type of database driven authentication to keep track of users. The application still executes as the APEX schema, not as the individual user in the database.

For simplicity, the default is to use the Application Express authentication scheme. This is the one setting that will provide login security; by default, the developer writing the application will be able to log in without any additional work.

■ **Note** Many organizations will have an existing method of authenticating users. If there is currently an LDAP server available (such as Oracle Internet directory, Microsoft Active Directory for network domain authentication, or even an Oracle eBusiness suite), it may be desirable to leverage these systems for APEX authentication. The number of options and methods are beyond the scope of this book. Simply know that with the Oracle database technology and the technology of your application server, it's possible to leverage many of the most common authentication infrastructures.

The authentication step in the wizard also allows for three additional settings, as seen in Figure 5–16:

- **Language:** Used for internationalization and translation.
- **User Language Preference:** For multi-lingual applications, this setting determines the translation needed.
- **Date Format:** Defaults how date elements will be formatted. Different regions of the world have assumptions on how dates are formatted, especially when they are strictly numeric values. A common format that is used to try to alleviate this issue is the DD-MON-YYYY format. This style of format makes it clear which portion represents the day, month, and year clearly (ie. 01-JAN-2010).

The wizard will use these setting as starting values. You can alter them as needed in the shared components of the application. The language, date format settings, and time zone handling are classified as globalization settings. One of the settings that can be turned on after the application is created is automatic time zone detection. Automated time zone detection is especially useful for applications whose users span different time zones.

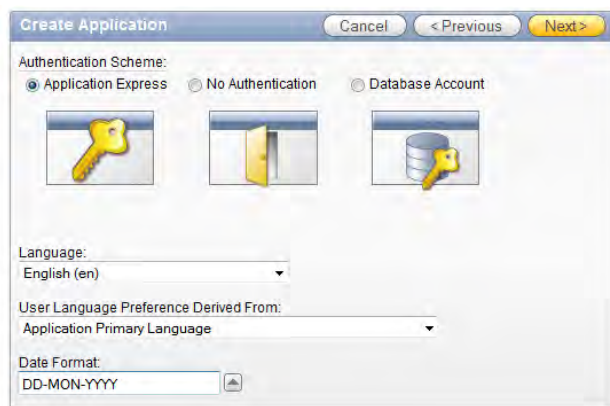


Figure 5–16. Authentication scheme selection

Selecting a Theme

APEX themes are the groupings of templates that are used to establish look and feel of pages, reports, buttons, and other graphical components. Figure 5–17 shows the APEX theme-picker. As APEX evolves and web standards evolve, so do the pre-made themes of APEX. Version 4.0 offers some new themes. Although APEX currently comes with twenty themes of varying looks, it is always possible to customize an existing theme or to create a completely new theme. The APEX administrator also has the ability to create themes that are specific to their instance of APEX. Choosing a theme as part of the Application Creation wizard is an easy way to apply a default theme. As you might expect, you can change your mind later and apply a different theme. Additional themes can be added, modified, and tested as part of the shared components of APEX.

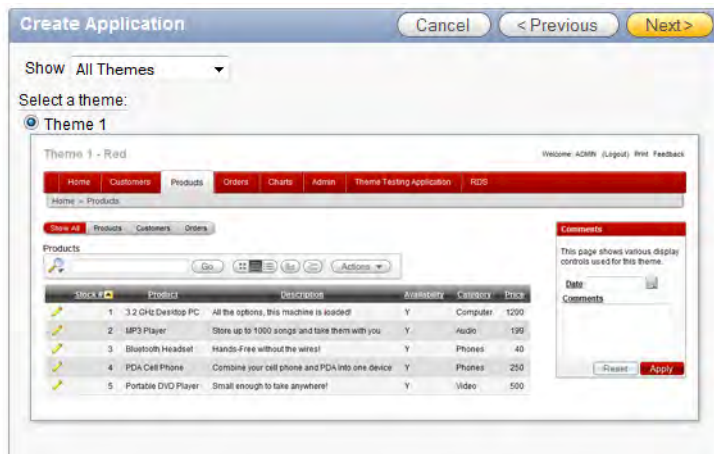


Figure 5-17. Theme selection

Completing the Application Creation Wizard

The last step of the wizard is a simple confirmation dialog. Clicking the Create button in Figure 5-18 will commit all of the settings and generate the application. The Previous button can walk backward through the wizard to make any additional changes before completing.

One additional option is available on the last page, at the very bottom: Save this definition as a design model for reuse. At the beginning of the wizard there was an option to re-use a design model. This is the point where those design models get created. They are sepecific to the workspace and can be very useful when running the wizard to set application defaults quickly.

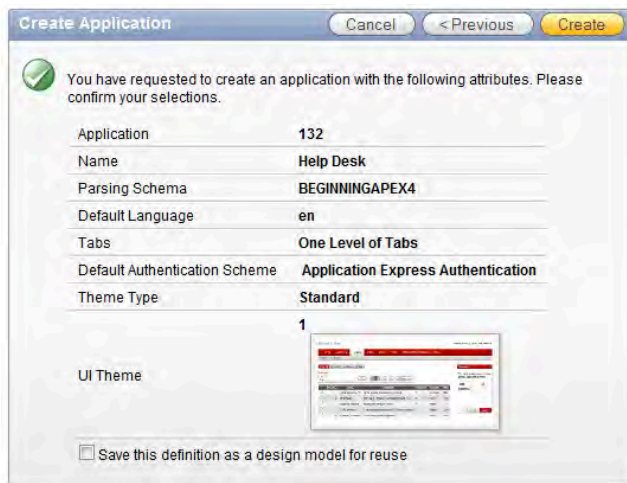


Figure 5-18. Completing the Application Creation wizard

Now that you have reviewed the steps of the wizard, let's create the application for the Help Desk.

1. Navigate to the Application Builder. Start the Application Creation wizard by clicking the **Create** button.
2. Select **Database** as the type of application. Click **Next**.
3. Select **From Scratch**. Click **Next**.
4. Enter **Help Desk** for the name of the application. Everything else can stay default values. Click **Next**.
5. Select **Blank** for the page type. Enter **Home** for the page name. Click **Add Page**.
6. Check your result. There should be one page listed, as shown in Figure 5–19.

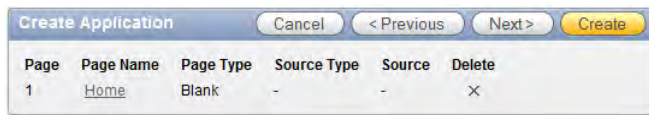


Figure 5–19. Application page creation

1. Click **Next**.
2. Select **One Level of Tabs**. Click **Next**
3. Select **NO** for Copy shared components. Click **Next**.
4. Select **Application Express** as the authentication scheme.
5. Set the date format to **DD-MON-YYYY**.
6. Click **Next**.
7. Select **Theme 1**. Click **Next**
8. Click **Create** to finish.

You will now have a simple application with only two pages, as shown in Figure 5–20. Run that application, and you should receive the login page shown in Figure 5–21. That login page will take your normal APEX developer username and password, as shown in Figure 5–22.



Figure 5–20. Resulting pages for the help desk application

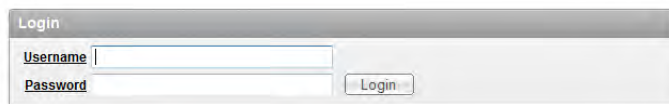


Figure 5–21. Login prompt when running application



Figure 5-22. The application after you've logged in

Static HTML Regions

The Static HTML region is a very basic type of region. There are three sub types of HTML region, as shown in Figure 5-23:

- **HTML:** Any HTML tags will render as their interpreted value; JavaScript will execute as if part of the page source.
- **HTML Text (with shortcuts):** Like the HTML region, with the addition of support for shortcut technology. This technology includes a shared component object that can be used for managing a type of variable using “SHORTCUT_NAME” syntax.
- **HTML Text (escape special characters):** This will show HTML code as the source value. Example: `
` will show up exactly as the code `
` rather than being interpreted as a break / return.

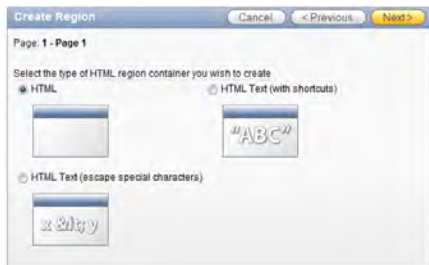


Figure 5-23. HTML region sub types

With the HTML region's simplicity comes a wide variety of uses. An HTML region is a container that can have its own value, embedded JavaScript or CSS definitions, or it can contain other page items. Any valid HTML entered in the source will render on an APEX page. Substitution string syntax, such as `&PX_ITEM_NAME.`, can also be used to display item values in the source text.

Continuing with your help desk application, you will add some content to your first page, like so:

1. From the Application Builder for the help desk application, click on the page with the house icon with a title of **Home**.

2. In the Page Rendering section, double click on the **Home** region as indicated in Figure 5–24. (Using a right click brings up the Edit Options menu).

If this is the first time you have visited the page edit screen, you will be presented with a pop-up dialog that introduces you to the functionality available on this screen. If presented with this dialog, simply press OK to dismiss it.

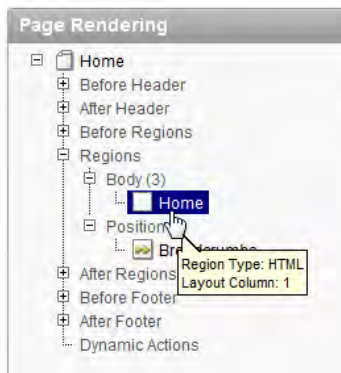


Figure 5–24. Home region location on the Page Rendering tree

3. Change the title of the region under identification title to **APEX Issue Tracker**.
4. Enter the following in the **Source - Region Source** section:


```
<h1>Welcome to the APEX Issue Tracking System</h1>
<br />Select an option from the list
```
5. Click on **Apply Changes**.

■ **Caution** A common mistake is to click the RUN PAGE shortcut in the corner BEFORE clicking Apply Changes.

Run the page. You should see the changes you just made indicated by a region with a friendly welcome message. Your results should be similar to those in Figure 5–25.

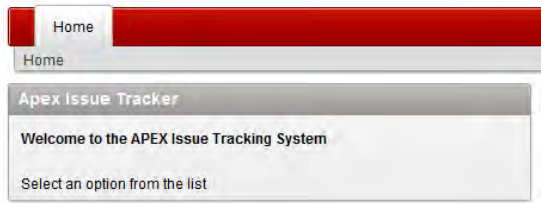


Figure 5–25. Results after adding the Static HTML page

Public Pages

As mentioned, it's possible to allow the entire application to use a no authentication scheme. But what if you want some of the pages to require authentication, and the rest can be public? How can you make a page that does not require a login to view?

If any of the pages within an application require authentication, an appropriate authentication scheme must be applied to the whole application. APEX allows individual pages to be defined as “public” or “requires authentication” using a defining property of the page. Each page can have different security requirements (authorization), but only one authentication mechanism can be applied to an application. Public pages are useful for introductory landing pages, login pages, and information pages.

In your help desk project, you want to have the main page available to all visitors. To accomplish this, you will modify the first page of your application to allow it to be seen by anyone without requiring authentication via the following steps:

1. For the help desk application, navigate to the page edit view for page 1.
2. In the Page Rendering section, the title of the page is the top of the tree. Double-click on the title **Home**.
3. Scroll to the Security section shown in Figure 5–26. In this section change the authentication to **Page is Public**.
4. Click **Apply Changes**.

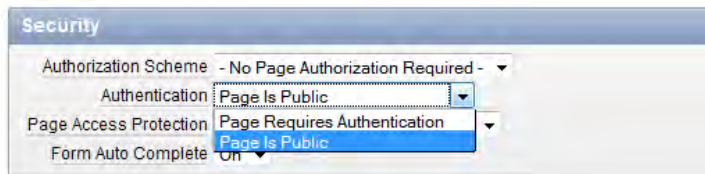


Figure 5–26. Change a page's authentication setting

Now when the page is run, the authentication screen will not show when page 1 is requested. You will learn more about authentication and authorization in Chapter 9. For now, just know that the change you've made allows users to see the first page of the application without being logged in.

Navigation Bar Entries

Each APEX application has one navigation bar that may contain multiple entries. Examples of links typically displayed on every page are login, log out, help, and my account. As a developer, you can create and modify navigation bar entries depending on the application and need. The navigation bar can also go beyond standard link text; it can be modified to include images. Entries can be based on conditions, authorization schemes, and build options. Placement of navigation bars is dictated by the page template substitution variable #NAVIGATION_BAR#. In most applications, the navigation bar is placed in either the upper right or upper left of the page. An example of a navigation bar can be seen in Figure 5–27. The icons located in the upper right are a navigation bar; they provide quick access to run page, shared components, and other common utilities.

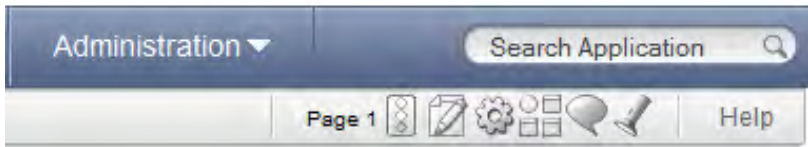


Figure 5–27. Icons in a navigation bar

For the Help Desk application, you will add entries to support the ability of users to log in and logout.

1. Navigate to the **Shared Components** section, either from the application page list or via the gear icon in the builder navigation bar.
2. Under Navigation, click **Navigation Bar Entries**, as shown in Figure 5–28. You'll see that APEX has already created a Logout entry for you, but you need to create your own Login link.

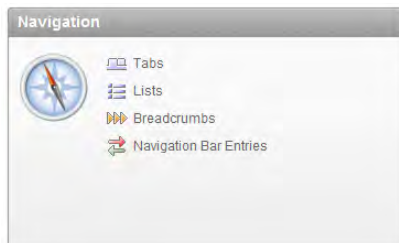


Figure 5–28. Navigation items in the Shared Components screen

3. Click **Create**.
4. Select the **From Scratch** option shown in Figure 5–29 and click **Next**.

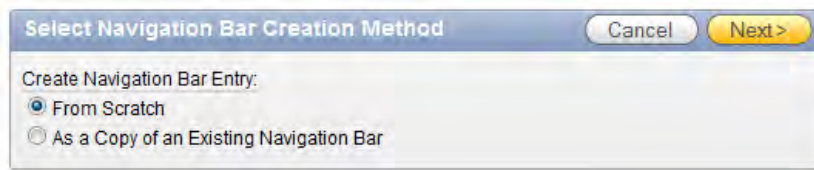


Figure 5-29. Navigation bar creation

5. Select **Navigation to URL** as shown in Figure 5-30. Then click **Next**.

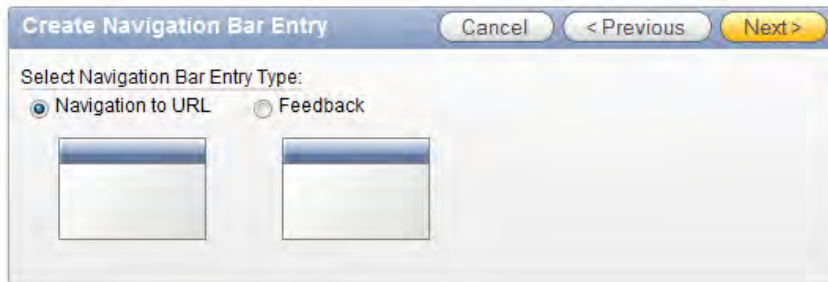


Figure 5-30. Navigation bar entry creation

6. Enter **Login** in the entry label field and **Login** in the image ALT field, as shown in Figure 5-31.
7. Click **Next**.

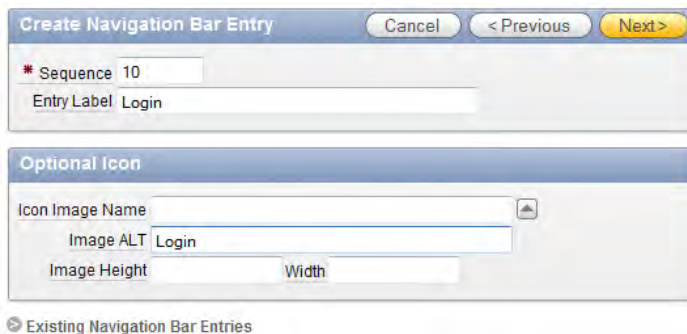


Figure 5-31. Navigation bar settings

8. Set the target to **Page in this Application**, as shown in Figure 5-32.
9. For page, enter **101**. This will send the user back to the login page after they've logged out.
10. Click **Next**.

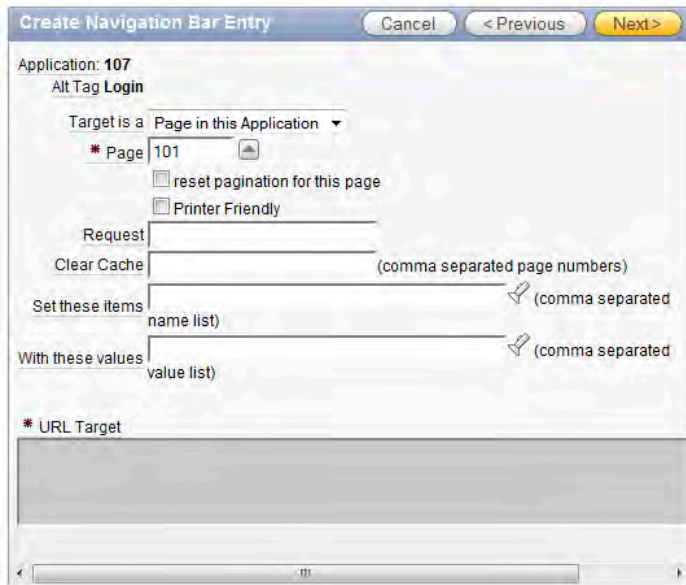


Figure 5-32. Specifying the target page

11. Set the Condition Type to **User is the Public User (user has not authenticated)**, as shown in Figure 5-33.
12. Click **Create**.



Figure 5-33. Navigation bar conditions

Run your application now. If you are logged in, you will only see the Logout navigation bar entry. Click the Logout navigation item. Once you are logged out, you will see the new navigation item as in Figure 5-34. This identifies a small problem: The Logout link can still be seen even though you've already logged out..



Figure 5-34. Logout and Login buttons

Clearly it's a problem to show the Login and Logout choices at the same time. After all, only one of those two choices can apply. Let's tackle that problem.

1. Navigate back to the Shared Components-Navigation Bar Entries and edit the **Logout** item.
2. Set the condition to **User is Authenticated (not public)**, as shown in Figure 5-35.
3. Click **Apply Changes**.

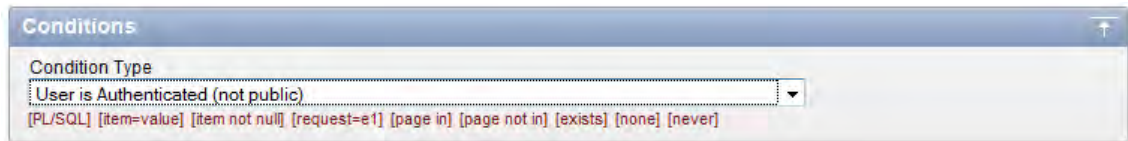


Figure 5-35. Navigation bar condition type

Run the application again. You should see that the Login and Logout navigation items are mutually exclusive.

When you created the new navigation item, you applied the condition to allow it to be seen only by the public user. The Logout navigation item was created as part of the application creation wizard. There was no condition placed on the Logout item by default. You will learn more about conditions in Chapter 8.

Page Zero

Page Zero is a special type of page. Items placed on Page Zero are rendered with every page. Page Zero is a useful construct when you find the need to copy regions to every page. Simply move a region to Page Zero, and it will be rendered with every page. A good example of usage would be a breadcrumb region or a region that contains custom JavaScript code that needs to be available to every page. Region contents from Page Zero will be included on every page even when a region does not render visibly.

You may notice when looking at the Page Zero in the APEX page editor (Figure 5-36) that there is no Page Processing section. Page Zero is used during Page Rendering only. Regions that are added to Page Zero will be included even on the Login page. The different page types in an application need to be considered when adding content on Page Zero.

■ **Note** The APEX Builder will allow you to add calculations and processes to Page Zero. These items will not execute.

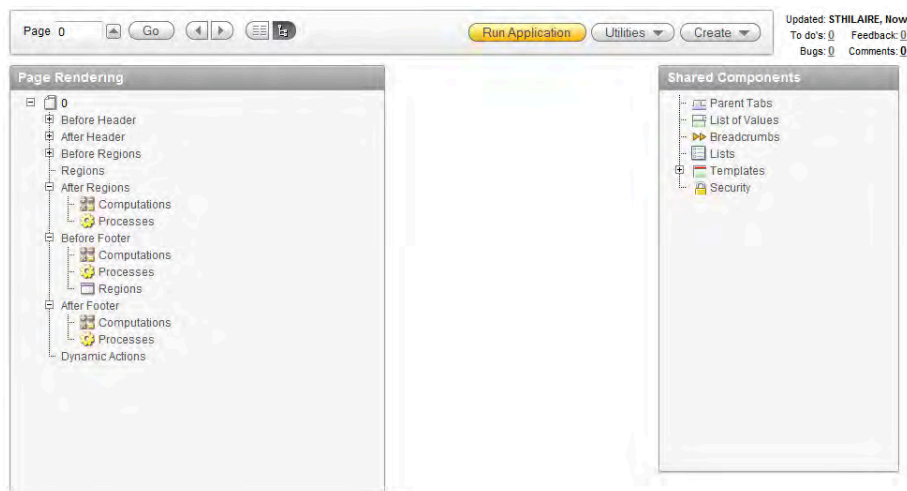


Figure 5–36. Page Zero in the APEX page editor

Creating a Page Zero is like creating any other page in an APEX application. Once a Page Zero is created, it will no longer be available in the Creation Options list. Let's create a Page Zero for your ticketing application.

1. From the Application Page List, click the **Create Page** button.
2. Select **Page Zero** from the Page Type list. Figure 5–37 shows the Page Zero option, which should be near the bottom of the list.

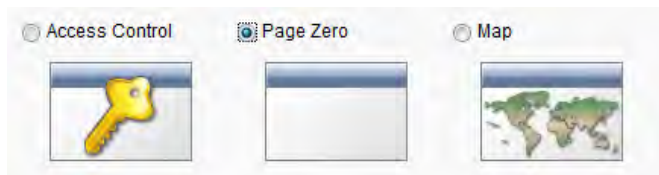


Figure 5–37. Choosing to create Page Zero

3. Click **Finish**.

You should now see Page Zero listed in the pages for the application. Currently there is no content on Page Zero. You will use Page Zero to contain and display the breadcrumb region.

Breadcrumb Region

Breadcrumbs are a popular navigation structure. They give a quick and intuitive representation of the current navigation path to the user with optional functionality to navigate back using the structure. Oracle Application Express uses the structure inside of the builder seen in Figure 5–38.

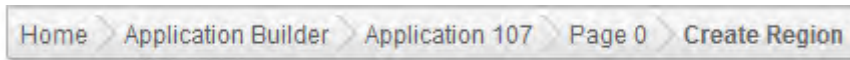


Figure 5-38. Example of breadcrumbs in Application Builder

In APEX, breadcrumbs are a declarative structure with built-in behavior. They are managed as shared components and have their own region type and template. When the Application Creation wizard was run, pages that were created with the wizard automatically included a region to contain the breadcrumbs. Figure 5-39 shows the Breadcrumbs region in Position 01 of the page.

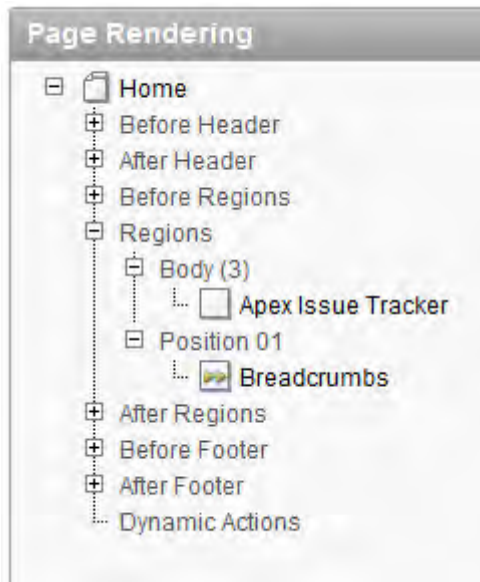


Figure 5-39. The breadcrumbs position in the page rendering hierarchy

When creating new pages of an application, the Create Page wizard has an option to assist in creating new breadcrumb entries. When the option is exercised, child pages will receive a copy of the breadcrumbs region from the parent as well as an automatic entry in the breadcrumb group. When a breadcrumb region does not exist, nothing will be copied, but the entry in the breadcrumbs shared component will still be created. An issue with this approach is if any changes to the region need to be made as far as display or other layout consideration, they will need to be done on every page which contains a breadcrumb region. Adding the region to Page Zero to make it show up on all pages can be helpful, as it gives you one point of change instead of many.

Continuing your help desk application, the design is to have a breadcrumb region show up on all pages. It is not necessary to re-create the region manually. Since the application wizard created the region for you, you will use the Copy Region feature in APEX to duplicate the region to Page Zero. Here you go:

1. Navigate to the Page 1 edit screen.
2. Use the right mouse button while on the **Breadcrumbs** region to show the context menu, as seen in Figure 5-40.

3. Select the **Copy** option.

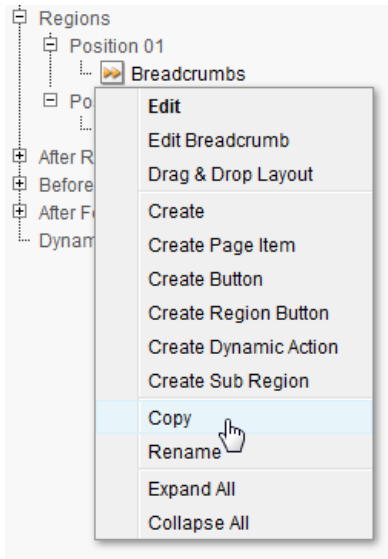


Figure 5-40. Context menu for the breadcrumbs region

4. Change the page number for the new region to 0, as shown in Figure 5-41 and click **Next**.

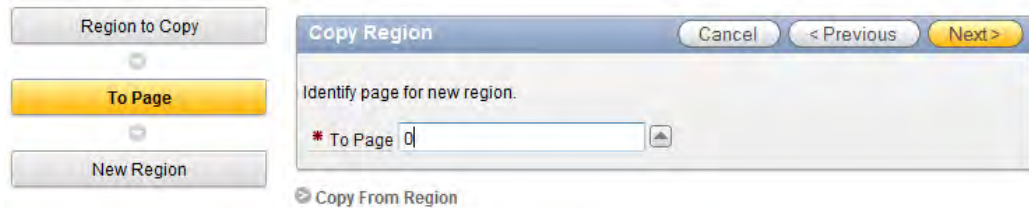


Figure 5-41. Setting the destination page

Part of the Copy wizard is to allow modification of what is copied in a limited fashion. Options that do not apply are disabled. In your current example, the region name and sequence can be modified as well as some display placement options. Leave them with their default values.

5. Confirm settings shown in Figure 5-42. Click the **Copy Region** to complete the wizard.

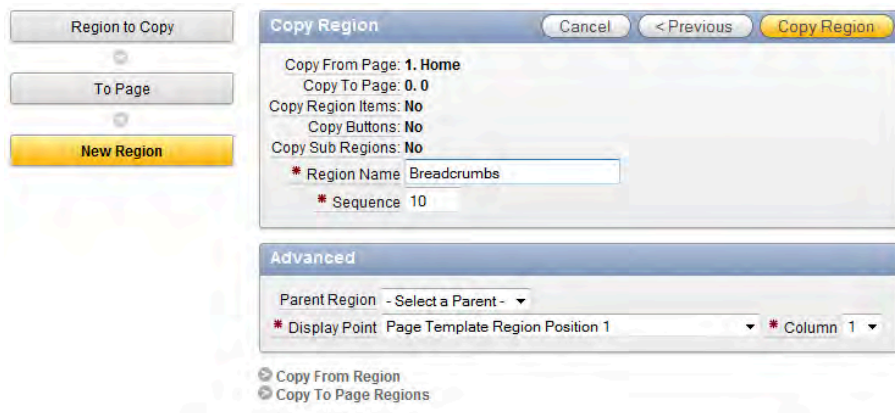


Figure 5–42. Confirming the copy operation

Reviewing the change in the editor, you notice that Page Zero now has the new breadcrumb region. Running your application, you will see the two breadcrumb regions shown in Figure 5–43. Note that the Copy feature does not remove the existing breadcrumb region.

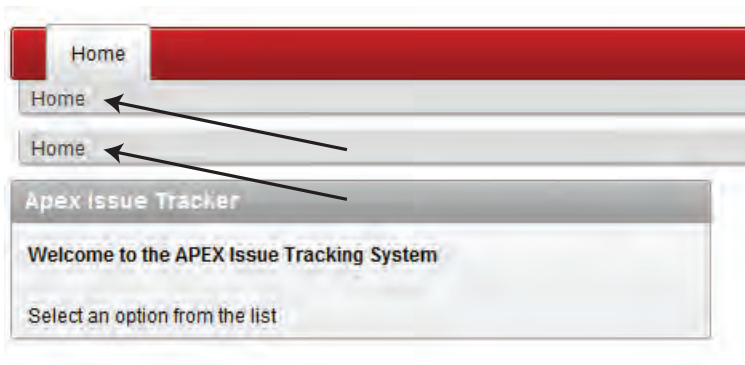


Figure 5–43. Redundant breadcrumb regions

To correct this duplication, do the following:

1. Edit **Page 1**.
2. Double-click on the **Breadcrumbs** region name.
3. Click on the **Delete** button at the top of the Region Definition, as shown in Figure 5–44.

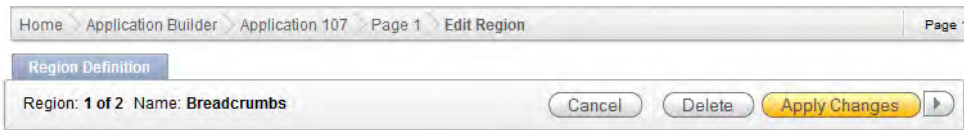


Figure 5–44. Preparing to delete redundant breadcrumb region

4. Click on the **Delete Region** button to confirm, as shown in Figure 5–45.



Figure 5–45. Confirming deletion

Now, re-test the application. You should just see the Page Zero version of the breadcrumbs region, as seen in Figure 5–46.

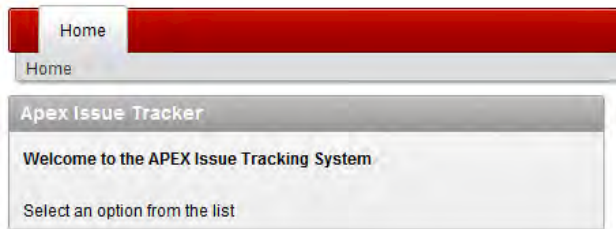


Figure 5–46. Completed migration of breadcrumb region to Page Zero

Effectively you have moved the management of the Breadcrumb region to Page Zero. Any settings changes to that region done on Page Zero will be seen on all pages of the application without requiring any additional work.

Breadcrumb Entries

As additional pages are added to the application, the Page Creation wizard will prompt for optional breadcrumb settings. If they were not set at the time the page was created or they need to be modified from their existing settings, the data that drives the breadcrumbs can be modified in the Shared Components section of the application.

It is possible to have several breadcrumbs within one application. A default breadcrumb with the name of “breadcrumb” is created as part of the APEX Application Creation wizard. This is the name of the grouping of breadcrumb entries. APEX provides some utilities to see where breadcrumbs are used and easy methods of editing entries.

To see the breadcrumb groups created, navigate to the Shared Components section and click on the option for breadcrumbs under the Navigation section. Figure 5–47 shows the main screen for listing the different breadcrumb groups.

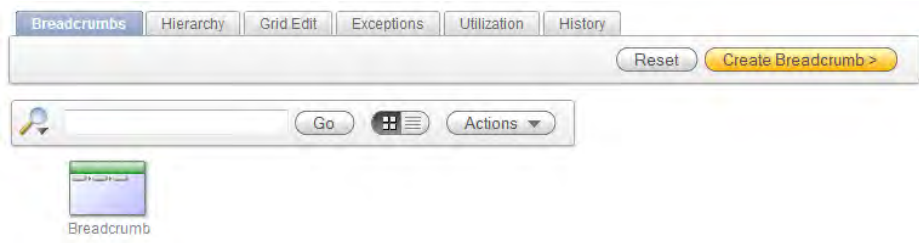


Figure 5–47. Breadcrumb groups available in the application

Clicking on the group name will display the detailed entries inside that group, as seen in Figure 5–48. The entries can be modified independently here. As an application becomes larger, it may be necessary to arrange the entries into different breadcrumb groups.

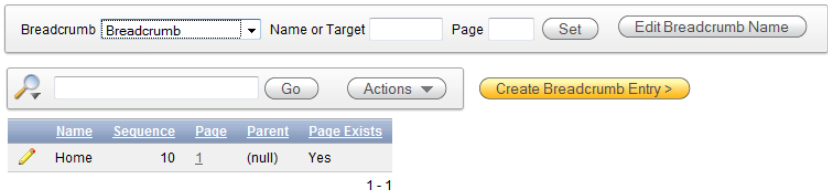


Figure 5–48. Detail of entries within a breadcrumb group

Lists

As the name implies, a list is a structure that APEX uses to keep a collection of data for links. The list structure allows menus to be displayed consistently across numerous application pages, with easy maintenance in the Shared Components area of an application. Do not confuse navigation lists with LOVs. Lists are a navigational structure with built-in templates for displaying the information in different ways. LOVs are used to support data entry, limiting the options a user can enter.

List templates have a lot of capability. They support hierarchical lists, graphical bullets, dynamic HTML, and highlighting for current page. Lists can contain data in a parent child relationship. Some list templates are specifically designed to display parent child data. APEX standard themes contain varying templates available for lists, but if the behavior you are looking for is not already available, it's possible to modify or create your own list template to display and behave as desired.

Your help desk application needs some lists to help with navigating to some key features. You are going to create some entries in the list for pages that do not exist just yet, but you will be creating them in the next few chapters.

1. Navigate to the Shared Components for your application.
2. Locate the **Lists** entry under Navigation.

3. No lists currently exist. Click on the **Create Button** shown in Figure 5–49.



Figure 5–49. The Lists maintenance screen

4. Enter **Home Page List** as the name, as shown in Figure 5–50.
5. Specify the list template as **Vertical Unordered List with Bullets**.

As with some of the other wizards, it is possible to complete the wizard or to continue with some additional steps. This is all you need to define for your example.

6. Click the **Create List** button.

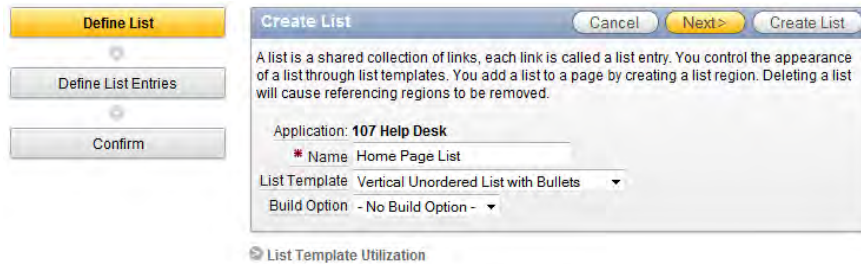


Figure 5–50. Selecting a name and template for a list

The wizard can optionally continue if you click the Next button. It will allow you to quickly create links and regions on pages. As your example application does not have pages created yet, you'll skip this step. You will be creating the links manually with additional options that are not available in the wizard.

Under the List Share component you should now see the Home Page List as an entry. In the next steps, you will add list entry values to support your application design.

1. Begin by navigating to the previously created Home Page List.
2. Click on the tab at the top of the region to view the **Entries by List** report shown in Figure 5–51.

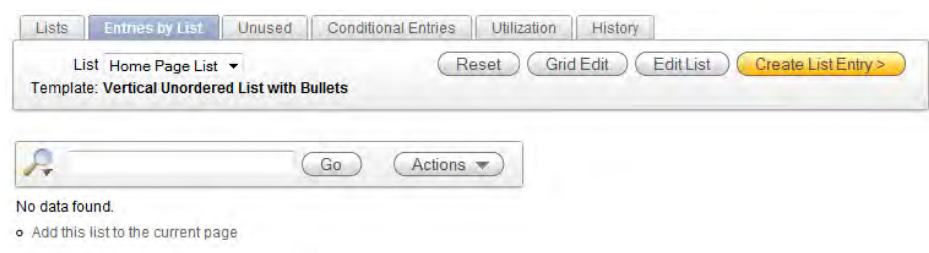


Figure 5–51. Entries by List view

- 3. Click the **Create List Entry** button.
- 4. The resulting dialogs will present all of the options available for a list entry. A lot of functionality is built into the Lists structure. The key items you are interested in are shown in Table 5–1. Fill out the dialogs shown in Figure 5–52 through 5–54 using the values from Table 5–1.

Table 5–1. Values To Use For The First List Entry

Section	Value	Entry
Entry	Sequence	10
	List Entry Table	Submit a Ticket
Target	Page	2
	Clear Cache	2
Current List Entry	List Entry Current for Pages Type	Colon Delimited Page List
	List Entry Current for Condition	2



Figure 5–52. Choosing a parent list entry

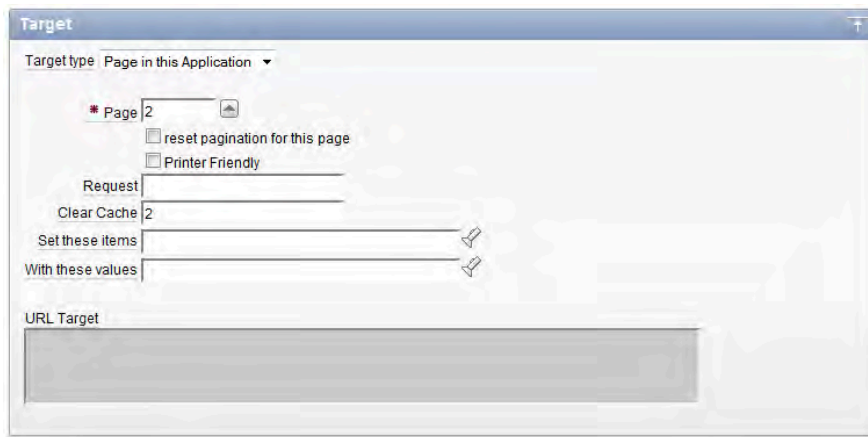


Figure 5-53. Target definition

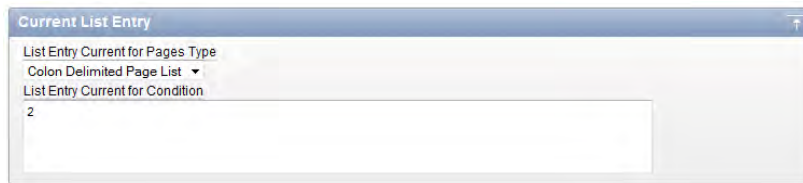


Figure 5-54. Current list entry

5. Once you have finished your entries for the first list item, scroll to the top of the page and click the **Create and Create Another** button. This brings you back to the same page and allows you to add another list entry. Use the information in Table 5-2 to create the second list entry.

Table 5-2. Values To Use In The Second List Entry

Section	Value	Entry
Entry	Sequence	20
	List Entry Table	Contact Us
Target	Page	3
	Clear Cache	3
Current List Entry	List Entry Current for Pages Type	Colon Delimited Page List
	List Entry Current for Condition	3

- 6. Click **Create** to save your changes.

You should now have a Shared Component list with two entries in it, as shown in Figure 5–55. The Entries by List tab shows some important information in a single view. The sequence value identifies the order in which the items are listed when using an un-ordered list type. Some list types are classified as ordered, in which case they will sort by name alphabetically. The target value is the construction of a URL that includes the page to navigate to as well as a clear cache instruction. Several of the declarative forms will construct a URL based on the inputs provided in the same way as the list entry.

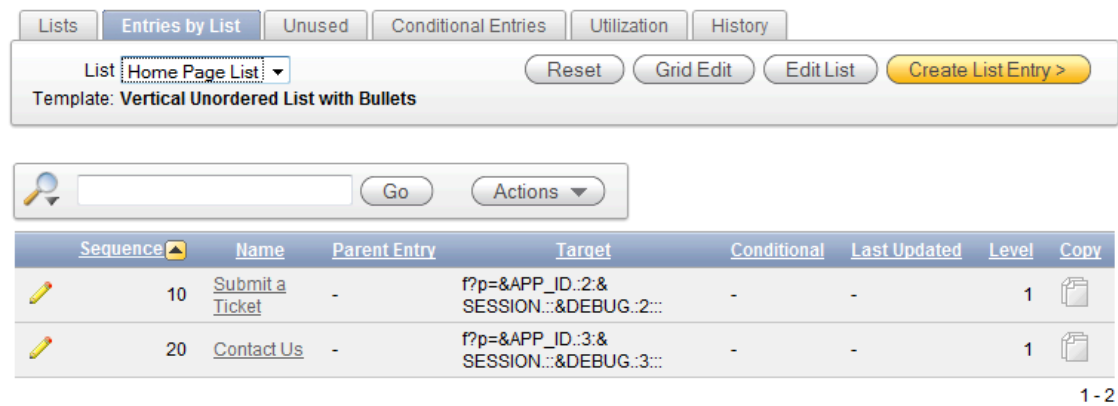


Figure 5–55. List entries at a glance

List Regions

A list as a shared component does not display in an application directly. A list needs a region to be configured on a page in order to be seen by the user. APEX has a template type defined specifically to support lists. The list templates contain all the intelligence required for dynamic lists and options for display. When creating a list region, the template choice can be set, and it can be modified through the region settings.

Now that you have created your list, you need to include it on your home page, and subsequent application pages. You will accomplish this by adding a region to your Page Zero, like so:

- 1. Navigate to the Page Zero edit page.
- 2. Use the **Create** button to create a **Region on this page** as shown in Figure 5–56.

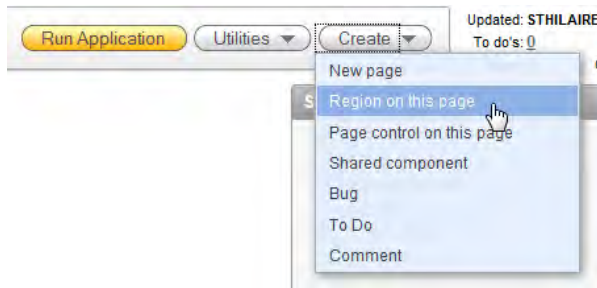


Figure 5-56. Creating a new region

3. Select the **LIST** region type.
4. In the Display Attributes step of the wizard shown in Figure 5-57, set the attributes as shown in Table 5-3.

Table 5-3. Display Attributes For The List Region

Attribute	Value
Title	Actions
Region Template	Sidebar Region, Alternative 1
Display Point	Page Template Region Position 3

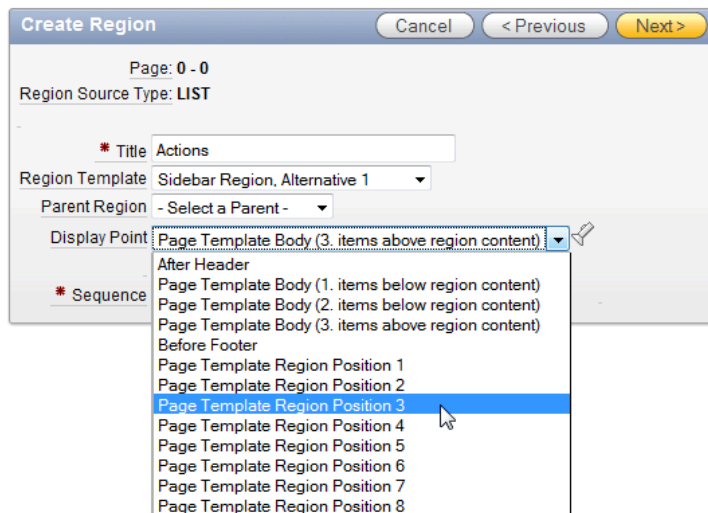


Figure 5-57. Creating region actions for lists

5. Click **Next**.
6. Select the list you created, **Home Page List**, as shown in Figure 5–58.

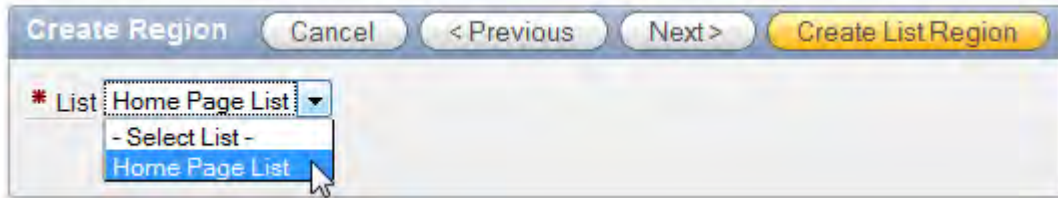


Figure 5–58. Selecting the list to be used in the region

7. Click **Next**. You will see the dialog in Figure 5–59.

You could stop now, end the wizard, and have your display region. But you really don't want your navigation links to display on each and every page. Specifically, you don't want them to display on the login page because the links won't work until after a user is logged in.

In this instance, you will create a Display Condition for the lists region. In your application, the login screen will be page number 101. The remainder of the public pages which we will create will have page numbers less than 100. So you setup a PL/SQL condition that evaluates the page number and will return TRUE when the page ID is less than 100. The page number can be referenced using the substitution variable `:APP_PAGE_ID`. The condition enables the display of your list region only on the non-login pages.

8. Set the condition type to **PL/SQL Function Body Returning a Boolean**
9. Set Expression 1 to the following code:

```
IF :APP_PAGE_ID <100 THEN
  RETURN TRUE;
ELSE
  RETURN FALSE;
END IF;
```

10. Click **Create Region**.

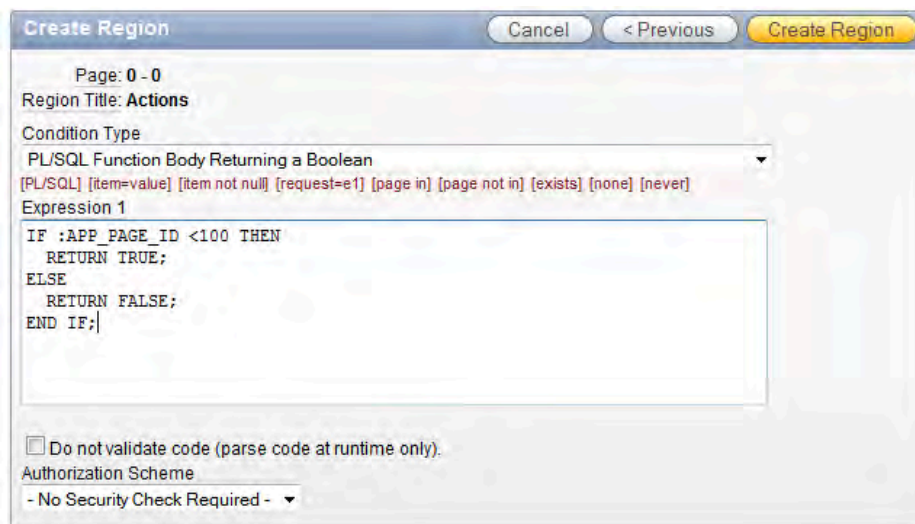


Figure 5–59. Specifying a conditional display option for a list region

Running your application now should result in the action list you created on the right side of the screen (Figure 5–60). Clicking on either link will cause the application to prompt for authentication. After you log in, clicking on a link will generate an application error. This is expected. You have asked the application to link to pages that do not exist yet.

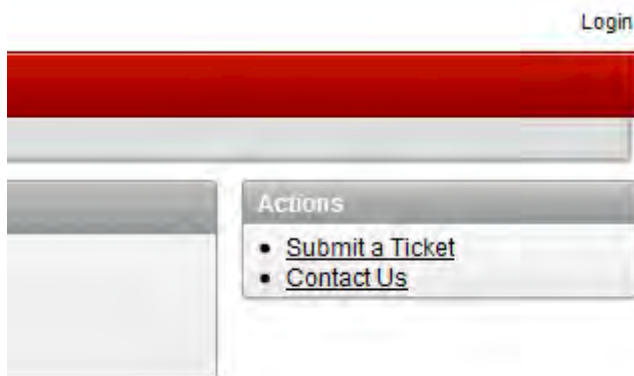


Figure 5–60. Application page with an action list

Limiting the display of the region to non-login pages is just a small example of what can be done with conditions. The full capabilities of the Oracle PL/SQL engine are available for conditional logic on many of the APEX components. Conditional logic can be very complex and comprehensive using combinations of functions, packages, and SQL. It is recommended that you spend some time

strategizing an approach to display conditions and security functions when designing a complex application. If the same condition logic is going to be used in many places, it may be a good idea to setup a utility package with functions to be called in several places. Thus, maintenance of the logic can be done in a single place.

Template Positions

In the previous section, you chose Region Position 3 when selecting the location of the region. This location was based on knowledge of the theme and template you chose at the beginning of the example application. When working with another theme, it may be desirable to see where the regions are placed. APEX has a utility that displays approximate region placement and allows quick selection of display points.

To get a preview of display points in a page template, follow these steps:

1. Navigate to the edit screen of any region.
2. Under the User Interface section, locate the **Display Point**.
3. Click on the **flashlight** to the right of the drop-down, as seen in Figure 5–61.

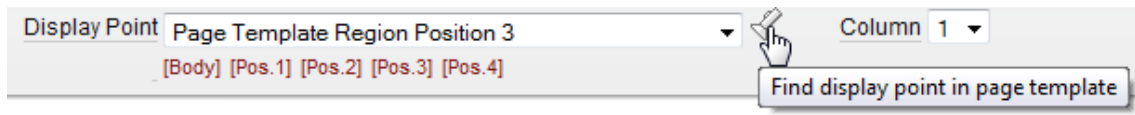


Figure 5–61. Location of the display point flashlight

4. In the resulting pop-up window shown in Figure 5–62, click on the name of a display position. Be aware that the positions are specific to the page template. There can be more than one page template per application.

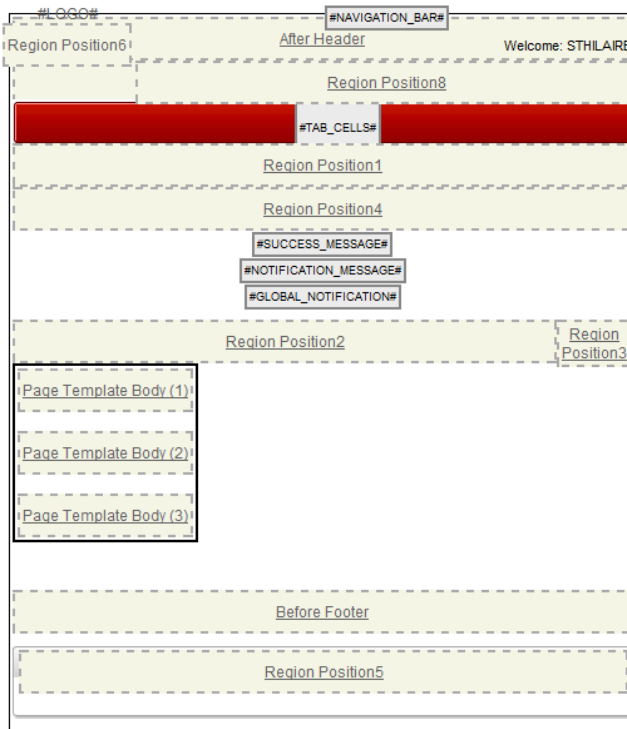


Figure 5–62. Resulting region positions available on the current page template

5. Click the **Cancel** button to prevent any changes from being saved.

This feature allows visual representation of region positions to better understand where your content will be displayed and how it will interact with other regions on the page. This feature is also very useful for understanding how a template may need to be modified in order to support the application.

Lists of Values

One of the fundamental benefits of writing an application on top of database architecture is the ability to enforce data quality. Lists of Values, or LOVs, are an APEX component that can be mapped to different item types including select lists, multiple select lists, checkboxes, and radio groups. These types of structures help assure data collected through transactions is consistent. There are two types of LOVs in APEX:

- **Static:** a set list of options within APEX.
- **Dynamic:** based on SQL against the database.

LOVs can be defined either as a shared component at the application level or at the item level. Figure 5–63 shows an item-level definition. LOVs used more than once should be written as a shared component. This allows the maintenance of that LOV to be centrally located with the shared

components. If an LOV was created at the item level, it is easy to convert it to a shared LOV by using a utility that APEX provides. When viewing a component with an item level LOV, the page will contain a Tasks menu with the Convert LOV option. This will make the LOV a shared component.



Figure 5–63. An item level LOV with static options

Static List of Values

A static list of values is simply a set of display and return value pairs. This type of list is normally short and un-changing. When defining a static list of values and item level, there are two types data options:

- **STATIC:** Entries are automatically alphabetized.
- **STATIC2:** Entries will render in the order which they are entered.

The syntax for specifying a static LOV is as follows:

```
TYPE:DISPLAY;RETURN, DISPLAY;RETURN,...
```

The TYPE may be either STATIC or STATIC2.

If you wish the display and return values for a given entry to be the same, then omit the semicolon and specify only one value. For example, the second item in the following example is a single value for both display and return:

```
TYPE: VALUE,VALUE,VALUE,...
```

The return value in a LOV is what is saved as the value of the associated form item. In static lists, using the semi-colon as the value of an entry may cause issues with parsing the list.

The following is an example of a static list. Commas separate the list items. Each list item is composed of a display and a return value, with a semicolon separating those two values.

```
STATIC:C;1,A;2,D;3,B;4,
```

When you display the values in this list, you'll see only the display values. Because the list of type STATIC, the values are displayed in alphabetical order:

```
A
B
C
D
```


Next is an example of a STATIC2 list. Notice the entries are specified in the same order as before.

STATIC2:C;1,A;2,D;3,B;4,

However, this time the values are displayed in their order of definition. They are *not* sorted alphabetically.

C
A
D
B

Shared component static LOVs have more options than item level static LOVs. Due to the shared nature, conditions per row and build option settings can be configured. These can be edited after the list has been created. Because the lists are stored differently as shared components, it's possible to use a semi-colon in the item value.

Dynamic List of Values

As with Static LOVs, Dynamic LOVs have a display and return value pair requirement. The difference is that the values are obtained through a SQL query. The SQL query written must have two columns returned. If the columns are the same, it will be necessary to use aliases to distinguish a display and return value. An alias will also have to be used if using a concatenated string as a column. Dynamic LOVs can also leverage session variables or values currently being used in the application. This gives dynamic LOVs flexibility to dynamically change what is offered during run time.

You will create two LOVs to support the selection of user names. In preparation to build your form pages, you are going to create a LOV to support the names of the Users and the Technicians in your Help desk system.

1. Navigate to the shared components of the help desk application, then to the User Interface section shown in Figure 5–64. Click the **List of Values** link.

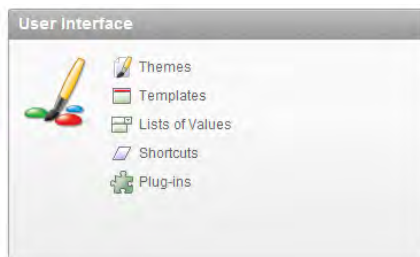


Figure 5–64. User Interface options

2. Click **Create** to create a new List of Values.
3. Create a LOV from scratch, as shown in Figure 5–65.

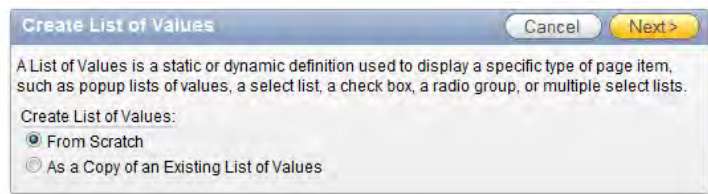


Figure 5-65. Creating a LOV from scratch

4. Click **Next**.
5. Enter **TECHS** as the LOV name. Select the radio button to make the list **Static**, as shown in Figure 5-66.

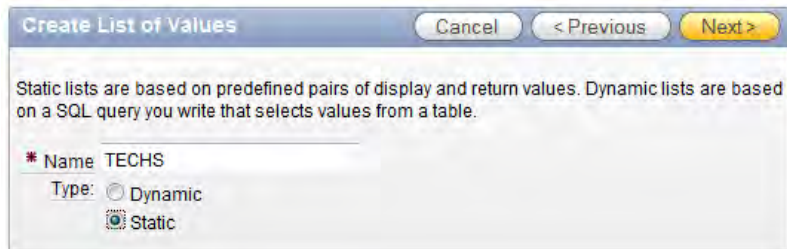


Figure 5-66. Specifying a list as static

6. Click **Next**.
7. Enter the values shown in table 5-4 into the form. Add your own name to the list!
8. Click **Create List of Values** when finished.

Table 5-4. Display Attributes For The List Region

Display Value	Return Value
Scott	SCOTT
Doug	DOUG
Karen	KAREN
Martin	MARTIN
Patrick	PATRICK
Tim	TIM
(Your Name)	(YOUR NAME)

Now that you have created a static LOV, let's include a second one that uses a SQL query to derive the list of values.

9. Repeat steps 1 through 4.
10. Create a second list named **USERS**, selecting the **Dynamic** option. Click **Next**.
11. Locate the book supplemental file named **ch5_lov.txt** that includes the SQL query text. Enter the SQL query for the LOV source.
12. Click **Create List of Values**

You should now have two LOVs created. Don't worry if you made a mistake. All of the settings can be modified. Simply double click on the name of the LOV you want to modify.

Summary

In this chapter, you created the basic shell of your application and several of the supporting objects that you will use in the upcoming chapters. These items have been created as a result of planning that was done prior to starting the application. Depending on your situation, the amount of planning you have done for your own application may vary. The shared components outlined here can be created at any time as needed during the development process. The next section will start utilizing some of these key structures outlined here.



Forms and Reports – The Basics

Now that you have database objects in place, you can get to the real work of building pages in your application. Most applications contain a series of forms, reports, charts, and other elements designed to display, edit, and collect data.

This chapter focuses on basic forms and reports. These are the simplest, most standard types of forms and reports in APEX. They are most often created by using the APEX Form and Report wizards, which create all of the elements that comprise a form or report for you.

In the sections that follow, you'll learn how to use the APEX Form and Report wizards to add pages to your Ticket application. You will create some basic forms and reports on your Tickets table; you'll also take a look at the elements created by the wizards that comprise your working forms and reports.

APEX Forms

Forms are used to display, edit, and collect data, which is then sent back to the database for processing. Forms can interface with tables, views (via “instead of” triggers), procedures, and web services.

An APEX form is actually a collection of APEX objects acting together as a single, cohesive unit to perform insert, update, and delete operations on data elements. An APEX form generally consists of a region, one or more items, one or more buttons, and one or more processes that handle interactions with the database. The APEX Form wizards create all the objects necessary for a fully-operational form.

■ **Note** Once a form is generated, the objects in it are not logically associated in any way except that they collectively make a complete working form. While it is possible to alter or delete individual elements, doing so may cause the form to not work properly if an error is introduced, thus it's not recommended.

The APEX Form wizards listed in Figure 6–1 are the fastest, most effective, and most accurate way to create APEX forms. The wizards guide you through a series of steps, collect the information required for the form type, and then generate all of the required items, processes, and buttons. Using the wizards frees you from the tedious and error-prone task of individually creating each component. After a wizard creates a form, you can, and likely will, make modifications and enhancements to the resulting components to tailor the form to your specific requirements.

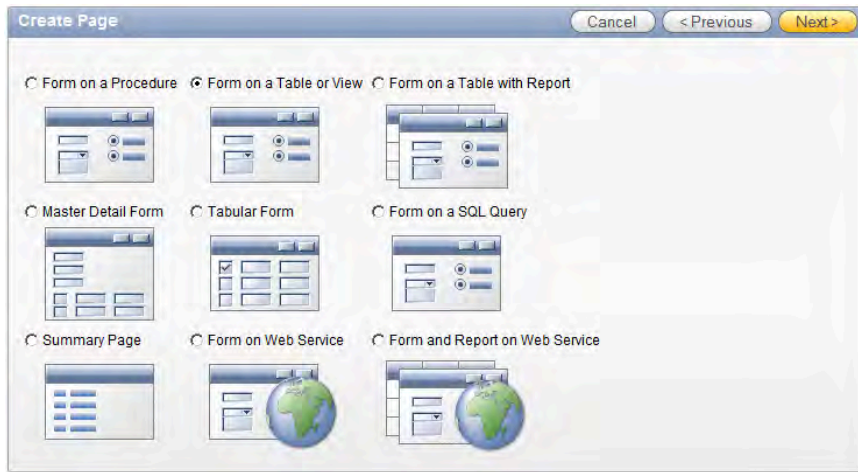


Figure 6–1. APEX Create Page form options

The following are some of the form types that you can create using the wizards listed in Figure 6–1:

- **Form on a Procedure:** A form based on the arguments of a procedure, typically to collect values for passing in to a procedure for subsequent processing.
- **Form on a Table or View:** A form built on the columns of a table or view, having one item for each table column and processing a single row of data at a time.
- **Form on a Table or View with Report:** A form built on the columns of a table or view, having one item for each table column and processing a single row of data at a time, plus a report on the contents of the table or view, with navigational elements between the report and form pages.
- **Master-Detail Form:** A form on a pair of tables having a master-detail relationship. The APEX Master-Detail Form wizard creates all the data, processing, and navigational elements required for managing master-detail data.
- **Tabular Form:** A multi-row, multi-column form (like a spreadsheet) that allows editing of multiple rows and columns of data at once.
- **Form on a SQL Query:** A form built on the results of a SQL query. This is a very powerful form construct due to its flexibility.
- **Summary Page:** A display-only form showing selected items from an existing input form page. A summary page is often used in building a confirmation page for a wizard.
- **Form on a Web Service:** A form on the arguments of a web service.
- **Form and Report on a Web Service:** A single-row form on the arguments of a web service with a corresponding report of all rows of data, including navigational elements for moving from report to form and back.

If you look at the available APEX Form wizards, you will see that several of them create accompanying reports (the Form on a Table with Report and Form and Report on a Web Service wizards). It is a common practice to use a report on a table, view, or web service to locate a particular row of data and then edit that data in a form on the same table, view, or web service. Some wizards simply create both the report and the form for you, including all navigation elements and database transaction processes required to make it all work.

Form on a Table

One of the most common types of Form in APEX is the Form on a Table. The APEX Form on a Table Wizard will automatically create and map APEX items to database columns, making it trivial to quickly create forms for database table entry and update. As a developer, you can then modify the different types of controls for each column. All of the supported HTML widgets (text fields, text areas, select lists, radio groups, checkboxes, etc.) are available, as well as several APEX-specific ones. The best way to understand just what the APEX Form on a Table wizard does is use it, so let's dive in and create a form on a table.

Creating a Form on a Table

You'll create page 2 of your Ticket System and add a form to it. This form will allow the user to create a new ticket by inserting a row to the TICKETS table. You can limit which DML operations a form in APEX can perform. In this case, you will restrict it to only performing inserts.

The Form on a Table wizard will walk through all the steps required to generate a form on a table: selecting the parsing schema, selecting the table upon which to base the form, selecting the columns to include and edit, assign region and form titles, and specify column headings.

1. Run your Help Desk application.
2. Click Create in the Developers Toolbar at the bottom of the screen.
3. Select New Page and click Next.
4. Select Form and click Next.
5. Select Form on a Table or View and click Next.
6. Set the Table/View Owner to your schema and click Next.
7. Enter TICKETS for the Table/View Name, as shown in Figure 6–2 and click Next.

Specify page and region information. If the page you specify does not exist, the page will be created.

Owner: **APRESS**
 Table / View Name: **TICKETS**

* Page Number:

Use User Interface Defaults: ☒ Yes ☐ No

* Page Name:

* Region Title:

* Region Template:

Breadcrumb:

Figure 6–2. Entering the name for a table or a view

8. Enter 2 for Page Number as shown in Figure 6–3. Enter Create a Ticket for both Page Name and Region Title. Set Breadcrumb to Breadcrumb and when the page refreshes, click on Home to set it as the Parent Entry as shown in Figure 6–4, and click Next.

Specify page and region information. If the page you specify does not exist, the page will be created.

Owner: **INTRO2**
 Table / View Name: **TICKETS**

* Page Number:

Use User Interface Defaults: ☒ Yes ☐ No

* Page Name:

* Region Title:

* Region Template:

Breadcrumb:

Figure 6–3. Specifying page and region information

Create Breadcrumb Entry

Entry Name

Parent Entry

[No parent breadcrumb entry]

Select Parent Entry:

Name	Page
Home	1

row(s) 1 - 1 of 1

Figure 6–4. Specifying breadcrumb parent entry

9. For Tab Options (Figure 6–5), select Use an existing tab set and reuse an existing tab within that set. When the page refreshes, set Tab Set to TS1 (Home), and click Next.

Create Form Cancel < Previous Next >

Page: 2

Tab Options:

☐ Do not use tabs

☐ Use an existing tab set and create a new tab within the existing tab set.

☒ Use an existing tab set and reuse an existing tab within that tab set.

Tab Set

Figure 6–5. Specifying tab options

10. Set Use Tab to T_HOME and click Next.
11. Ensure that the Primary Key is set to TICKET_ID and click Next.
12. Set the Source Type to Existing Trigger and click Next.
13. Select SUBJECT, DESCR, CREATED_BY, and STATUS_ID as shown in Figure 6–6, and click Next.

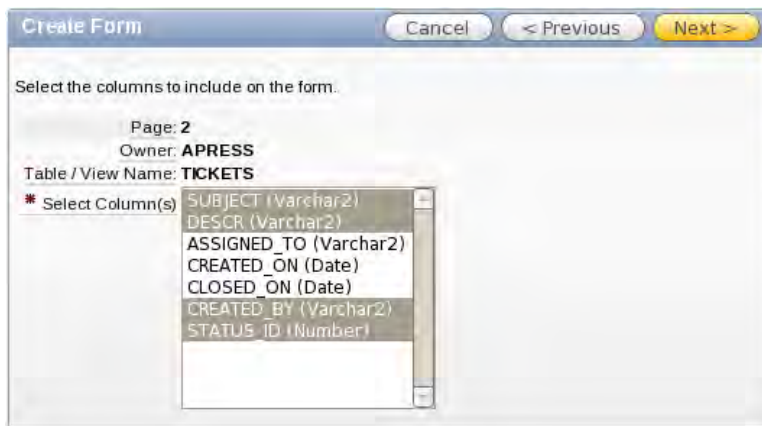


Figure 6-6. Selecting the columns to include

14. Enter Create a Ticket for the Create Button Label, as shown in Figure 6-7. Set Show Save Button and Show Delete Button to No, and click Next.

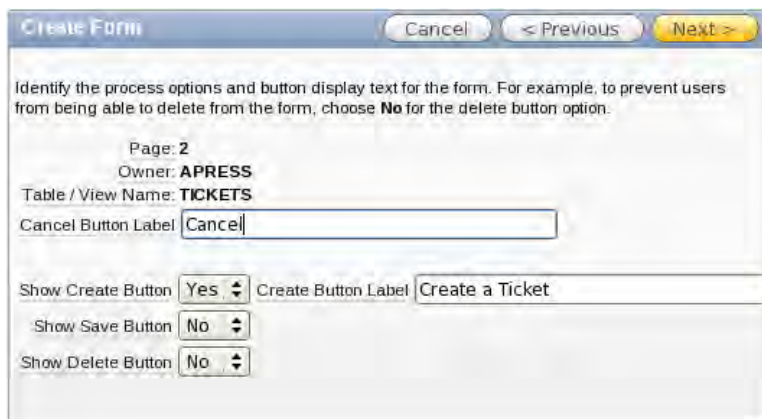


Figure 6-7. Specifying the buttons to display

15. Set both the After Page Submit and Processing Branch to Page and When Cancel Button Pressed Branch to this Page to 1 and click Next. See Figure 6-8.

Figure 6–8. Specifying processing for submit and cancel

16. Click Finish to complete the wizard.
17. Run your application.

Congratulations! You've just created a fully operational form on the TICKETS table. Your form should look similar to Figure 6–9.

Figure 6–9. Running the form on the TICKETS table

Notice that the form region is labeled as you specified in step 8, the form contains fields for the four columns you selected in step 13, and the Create button is labeled as you specified in step 14. Notice that the four fields are each created as the default element type specified in the User Interface Defaults for the TICKETS table that you created in the previous chapter. The help text you specified for each column is there and it pops up in a new window when you click on the item label. The Cancel button brings you to the home page—page 1, as you specified in step 15. APEX did a lot of work for you!

Modifying a Form on a Table

The APEX wizards do handle most of the work for you. However, it's rare that you won't have to make some minor changes to what the wizard creates. Now that you have your Tickets form on page 2, you're going to make a few changes to polish it up a bit.

You'll change the label templates for P2_SUBJECT and P2_CREATED_BY (the items that correspond to the SUBJECT and CREATED_BY table columns) to Required Label with Help. Use of the Required Label with Help label template indicates to the end user that this is a required field on your form. You'll also reduce the width of P2_CREATED so that it does not take up as much space.

1. Edit page 2 of your application.
2. Edit the item P2_SUBJECT by double-clicking on its name.
3. In the Label region shown in Figure 6–10, set the template to Required with Help and click Apply Changes.

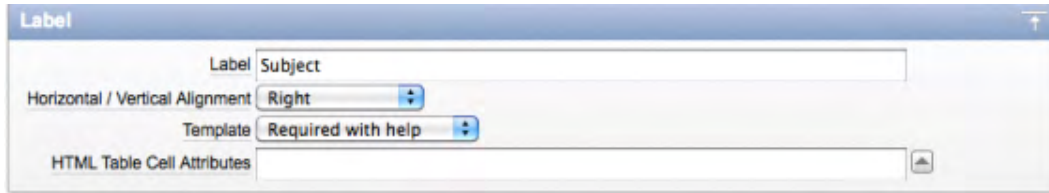


Figure 6–10. Modifying the label templates

4. Edit the item P2_CREATED_BY by double-clicking on its name.
5. In the Label region, set the template to Required with Help.
6. In the Displayed region, shown in Figure 6–11, set the width to 20 and click Apply Changes.

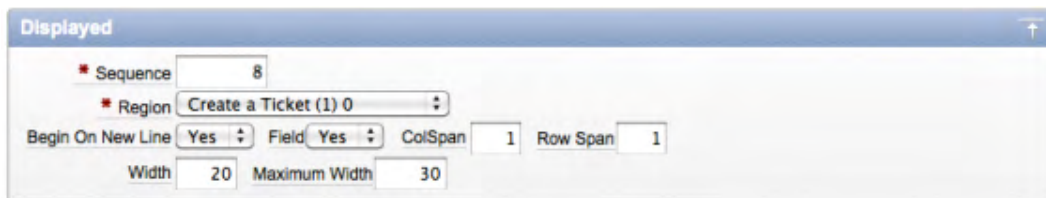


Figure 6–11. Setting the display attributes

Next, you want to hide the P2_STATUS_ID item from the user because you don't want the user to change this value. You do, however, want all new tickets to be created with a default value of OPEN. Since you cannot guarantee which STATUS_ID maps to which STATUS, you can call a simple function and pass in the STATUS. This function will, in turn, return the corresponding STATUS_ID, which will be set as the default value for P2_STATUS_ID.

7. Edit the item P2_STATUS_ID by double-clicking on its name.
8. In the Name region, set the Display As to Hidden.
9. In the Default region shown in Figure 6–12, set the Default Value to RETURN get_status('OPEN'); and set the Default Value Type to PL/SQL Function Body, then click Apply Changes.

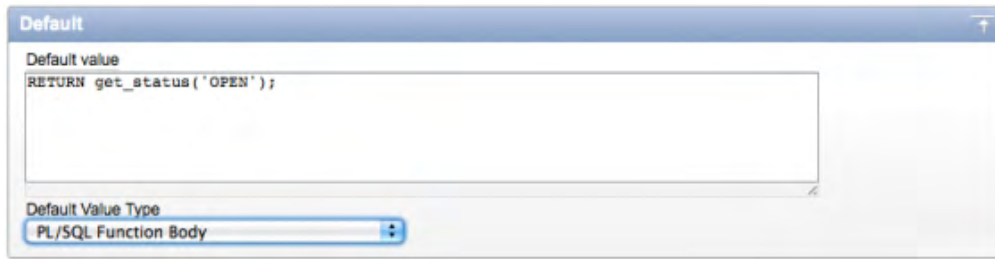


Figure 6–12. Specifying a default value

Next, you'll want to set page 2 to be a public page. You want any user—authenticated or not—to be able to access this page.

10. Edit the page attributes for page 2 of your application by double clicking on its name in the Page Rendering tree.
11. Set page 2 to be a public page. Refer back to Chapter 4 for detailed steps.

Finally, you need to make sure that your users enter values for the Subject and Created By fields. There are two ways to make a field mandatory within APEX. You'll use one method for each field.

For the Subject field, you'll create a validation. While a validation takes more steps, it gives you more control over how and when it is performed.

1. Edit page 2 of your application.
2. Create a new Validation by right-clicking the Validating node on the Page Processing tree and selecting Create Validation, as shown in Figure 6–13.

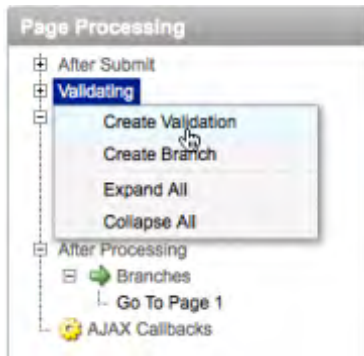


Figure 6–13. Choosing to create a new validation

3. Select Item Level Validation and click Next.
4. Select Create a Ticket: 2 P2_SUBJECT (Subject) and click Next.
5. Select Item Not Null and click Next.
6. Set validation name to P2_SUBJECT is NOT NULL, as shown in Figure 6–14, and click Next.

Create Validation [Cancel] [< Previous] [Next >]

Specify the sequence in which your validation executes. Identify a name for the validation to make it easy to locate in the future. Also specify the location where an error message display if the validation fails.

Page: 2
 Level: Item level validation
 Item: P2_SUBJECT
 Validation Method: Item specified is NOT NULL

* Sequence: 10
 * Validation Name: P2_SUBJECT is NOT NULL
 Error Display Location: Inline with Field and in Notification

Figure 6–14. *Entering the details for a new validation*

7. Under the Error Message Text Area, click the quick link that reads [#LABEL# must have some value.] and click Next.
8. Take the defaults on the final screen and click Create.

At this point, you'll see a new validation under the Page Processing region called `P2_SUBJECT is NOT NULL`. Next, you'll use the second method to make the Created By field mandatory. To do this, simply set an attribute of the input item.

9. Edit `P2_CREATED_BY` by double-clicking its name.
10. In the Settings section shown in Figure 6–15, set value required to Yes and click Apply Changes.

Settings

Value Required: Yes
 Submit when Enter pressed: No
 Disabled: No

Figure 6–15. *Making a value required*

Once you return to the page edit screen, you'll see that no new validation has been created. That is because you used the item-level attribute instead of creating a full validation.

The main difference between the attribute and a full validation is that you can't conditionally control when the attribute is applied and you don't have direct control over the error message that is emitted.

Go ahead and run your application again. At this point, you will be able to enter new Tickets into the system, but not see them anywhere outside of SQL Workshop.

Looking Behind the Scenes

Now that you have working form, let's take a look at just what the APEX Form wizard built in order to understand a bit more about how your form works. You'll use the Firefox Web Developer Toolbar Add-in to display the form details. Figure 6–16 illustrates your Create a Ticket form with the form details exposed.

The screenshot shows a web form titled "Create a Ticket". At the top right are buttons for "<button>", "Cancel", "<button>", and "Create a Ticket". The form fields and their associated HTML tags are as follows:

- Ticket ID:** An input field with the value "1402623877877105". The HTML tag is `<input id="P2_TICKET_ID" name="p_t01">`.
- Subject:** An input field with the value "15EB752FC1FAB957I". The HTML tag is `<input name="p_arg_names" value="15EB752FC1FAB957I">`. The label is "*Subject".
- Description:** A text area with the value "1402800120877116". The HTML tag is `<input name="p_arg_names" value="1402800120877116">`. The label is "*Descr".
- Created By:** An input field with the value "1403029382877116". The HTML tag is `<input name="p_arg_names" value="1403029382877116">`. The label is "*Created By".
- Status:** An input field with the value "1403204571877120". The HTML tag is `<input id="P2_CREATED_BY" name="p_t04" size="20" maxlength="30">`.
- Status ID:** An input field with the value "1403404867877120". The HTML tag is `<input name="p_arg_names" value="1403404867877120">`.
- Status Value:** An input field with the value "2". The HTML tag is `<input id="P2_STATUS_ID" name="p_t05" value="2">`.

At the bottom, there is a text area with the value "4DB3B85375C10311I" and the HTML tag `name="p_arg_values" value="4DB3B85375C10311I">`.

Figure 6–16. Form on TICKETS Table with form details exposed

■ **Note** The Firefox Web Developer Toolbar add-in is a free web development tool that lets you inspect various aspects of a web page. To learn more about Web Developer and other web development tools, see Add-Ins for Firefox at <https://addons.mozilla.org>.

The highlighted input tags display the input identifier and name for each field of the form. Both are unique for each form field. The input identifier is the column name prepended with the page number. The input name identifies the element names that APEX uses internally to process data in the form. Note the columns you did not choose to display in the form, TICKET_ID and STATUS_ID, are still present in the HTML of the page.

A look behind the scenes tells you more. Click on Edit Page 2 in the Developer's toolbar to view the elements that comprise your new form. You should see results similar to those in Figure 6–17.

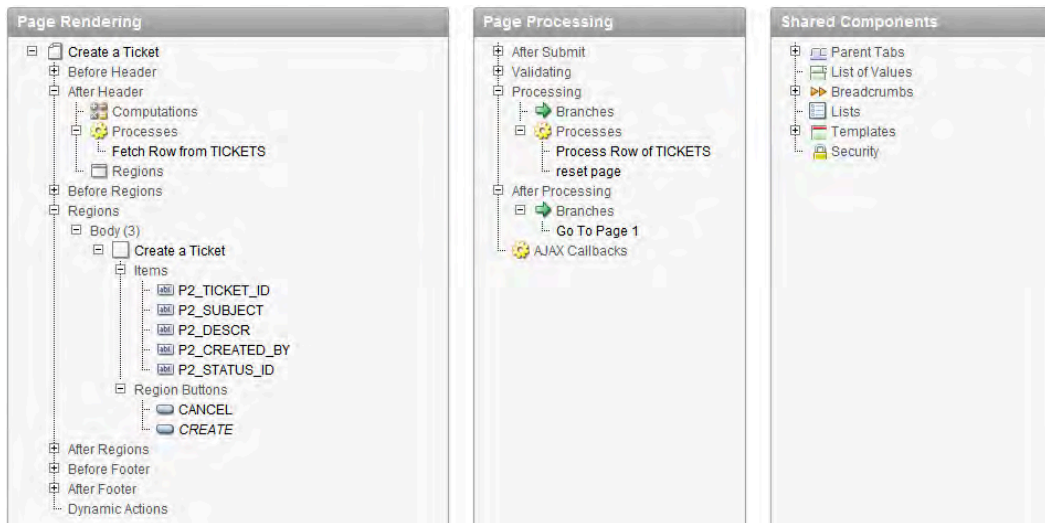


Figure 6–17. Elements of a form as viewed from Application Builder

The Page Rendering region contains APEX objects required for page rendering. The Page Processing region contains objects required for page processing, such as validations, processes, and branches. The Shared Components region contains APEX objects that are shared across pages, such as tabs, list of values, breadcrumbs, templates, and security schemes.

For your new Tickets form, in the Page Processing section, you see the wizard has created one item for each of the columns in your TICKETS table, plus two buttons called CANCEL and CREATE, and a Fetch Row from TICKETS process. This process is an Automated Row Fetch process, which does exactly what its name says: it fetches a row from the designated table into the current form. The attributes of the Automated Row Fetch process specify the table owner, the table name, the primary key column(s), success and failure messages, and a condition.

Notice that the TICKET_ID item is present on the page, but is not rendered on the form. It is visible in the Display Details view of the form as the first element on the page, with no visible element associated with it. TICKET_ID is a hidden item. APEX hidden items exist to hold a value, but are not rendered on the page—they are not visible to the user. In this case, the hidden TICKET_ID column holds the primary key value for the TICKETS row. As the primary key, TICKET_ID is used by the APEX processes to pull data from the database, and to process inserts, updates and deletes on a TICKETS row.

In the Page Rendering column, you have a Process row of Tickets process, a reset page process, and a Go To Page 1 branch. The Process Row of Tickets process does just that, process one row of the TICKETS table using the values from the items that correspond to the columns of the TICKETS table. This process fires when the user clicks the CREATE button. The reset page process clears the items on the page. It fires when the user clicks the CANCEL button.

In the Shared Components region, you need to expand the Parent Tabs tree node to see that this page uses the Home tab in the TS1 tab set. Expanding the Breadcrumbs region shows the Breadcrumb object. Under Templates, you see your form uses the default One Level Tabs – Right Sidebar(optional/table based) page template, the Form Region template, two different Label templates, and the default Button template.

You will see that the APEX Form wizards all create items, buttons, and processes, but in different combinations to suit the specific needs of the form type. The other APEX form wizards perform essentially the same way, with slight differences in process types and navigation objects to

accommodate the underlying data source: table or view, procedure, query or web service. Next, let's look at a form on a procedure.

Form on a Procedure

Another way to create a form in APEX is to create it based on the parameters of a PL/SQL procedure. Instead of the traditional DML processes, APEX will call the associated procedure and execute whatever logic is embedded within it. This method is also referred to as “using Table APIs” because this is the option to use if all access to tables in your workspace schema must be through a Table API.

Creating a Form on a Procedure

The process to create a Form on a Procedure is almost identical to that of a Form on a Table. You'll create a new page containing a form on the CONTACT_US stored procedure to enable users to contact you through the Ticket System application.

1. Run your application.
2. Click Create in the Developer's Toolbar.
3. Select New Page and click Next.
4. Select Form and click Next.
5. Select Form on a Procedure and click Next.
6. Set the Procedure Owner to your schema and click Next.
7. Enter CONTACT_US for the Stored Procedure Name, as shown in Figure 6–18, and click Next.

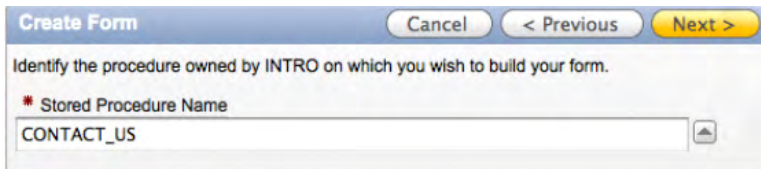


Figure 6–18. Creating a form on a stored procedure

8. Enter 3 for page number, Contact Us for both page name and region name, and set Breadcrumb to Breadcrumb, as shown in Figure 6–19. Then click on Home to set it as the parent entry, as shown in Figure 6–20, and click Next.

Create Form [Cancel] [< Previous] [Next >]

If the page you specified does not exist, Application Builder will create that page for you.

Owner: **APRESS**

Stored Procedure Name: **CONTACT_US**

* Page Number:

* Page Name:

Region Template:

* Region Name:

Submit Button Label:

Cancel Button Label:

Breadcrumb:

Figure 6–19. Specifying form page, region, and button names

Create Breadcrumb: Entry

Entry Name:

Parent Entry:

[No parent breadcrumb entry]

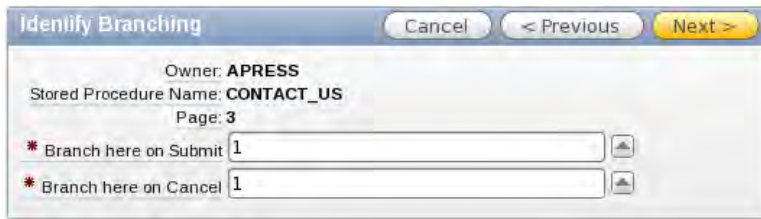
Select Parent Entry:

Name	Page
<u>Home</u>	1
Create a Ticket	2

row(s) 1 - 2 of 2

Figure 6–20. Selecting the breadcrumb parent entry

9. For Tab Options, select Use an existing tab set and reuse an existing tab within that tab set. When the page refreshes, set Tab Set to TS1 (Home) and click Next.
10. Set Use Tab to T_HOME and click Next.
11. Leave the Invoking Page and Button Label blank and click Next.
12. Enter 1 for both Branch here on Submit and Branch here on Cancel, as shown in Figure 6–21. Then click Next.



The 'Identify Branching' dialog box shows the following configuration:

- Owner: APRESS
- Stored Procedure Name: CONTACT_US
- Page: 3
- Branch here on Submit: 1
- Branch here on Cancel: 1

Figure 6–21. Specifying branching options

13. In the Select Procedure Arguments dialog in Figure 6–22, set the label for P_FROM to From. Set the label for P_BODY to Message. Set the display type for P_BODY to Textarea and then click Next.
14. Click Finish.



Argument	Label	Include	Default	Display Type
P_FROM	From	Yes		TEXT
P_BODY	Message	Yes		Textarea

Figure 6–22. Specifying procedure arguments

Modifying a Form on a Procedure

Once again, the wizard has done most of the work, but you have a few minor changes to make before your Form on a Procedure is complete. You want both the FROM and MESSAGE values to be required, so you'll change their label templates and set their Value Required attribute to Yes.

1. Edit page 3 of your application.
2. Edit P3_FROM by double-clicking on its name.
3. In the Label section, change the template to Required with help.
4. In the Settings section, change value required to Yes.
5. Click the > icon at the top of the page (next to the Apply Changes button) to save your changes and advance to the next item.
6. In the Displayed section, set the width to 80 and the height to 5.
7. In the Label section, change the template to Required with help.
8. In the Settings section, change value required to Yes.
9. Click Apply Changes.

Next, set page 3 to be a public page. You want any user—authenticated or otherwise—to be able to send you a message through the Contact Us page.

10. Edit page 3 of your application.
11. Set page 3 to be a public page. Refer back to Chapter 4 for detailed steps.

Lastly, you'll modify the process that was created to include a success message.

12. In the Page Processing tree, edit the process Run Stored Procedure by double-clicking on its name.
13. In the Messages region, enter the following for the Process Success Message:

Your message has been sent.

14. Scroll to the top and click Apply Changes.

Run your application and test out the Contact Us form. Each time you submit a record, an e-mail will be sent info@example.com. If you want to change the destination address for the e-mail, you can use the SQL Workshop's Object Browser to edit the CONTACT_US procedure.

Looking Behind the Scenes

From the user perspective, there is no indication that the form you've just created was created on a procedure. Looking in the Application Builder, the objects in the Page Rendering sections are similar to what you saw in your Form on a Table on page 2, but not exactly. Let's take a look to see what makes your Form on a Procedure different from the Form on a Table. Edit page 3 of your application. Your Application Builder page should look similar to that in Figure 6–23.

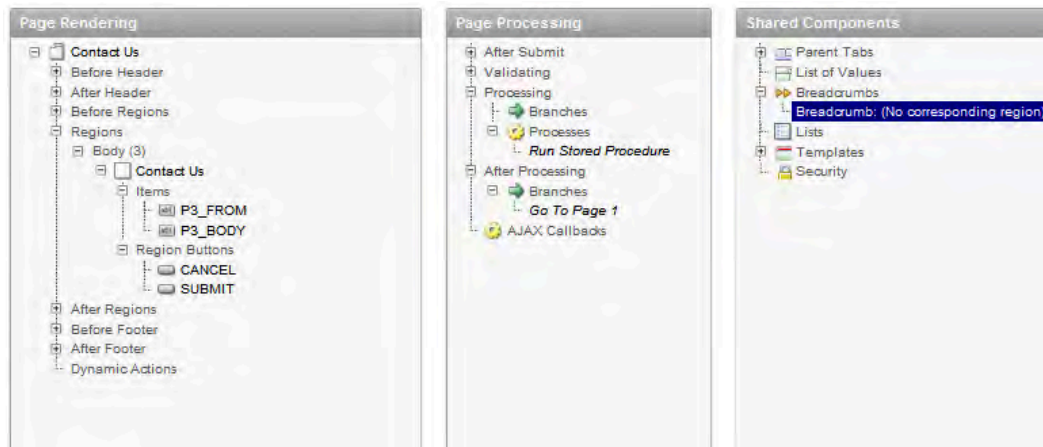


Figure 6–23. Elements of a Form on a Procedure as viewed from Application Builder

In the Page Rendering section of the Application Builder, you have two items, P3_FROM and P3_BODY, corresponding to your two form fields, From and Message. You have two buttons, Cancel and Submit.

In the Page Processing section, you have a process and a branch. However, your process is a different type—a PL/SQL anonymous block. This powerful type of process executes the PL/SQL procedure specified in the Source element. The PL/SQL procedure can be a stored PL/SQL procedure or an anonymous PL/SQL block, as long as the code is syntactically correct between a BEGIN and END

statement. In this case, the process calls the `CONTACT_US` procedure using the `P3_FROM` and `P3_BODY` item values as input parameters. The body of the `CONTACT_US` procedure is what creates and sends an e-mail to the instructor. Thus, the key difference between the Form on a Table and the Form on a Procedure is in the Page Processing process that gets executed upon click of the Create button. Your APEX wizard has automatically provided the process type required for the selected form type.

The Shared Components region contains the standard entries for the table, breadcrumb and page, tab, region, label, and button templates, the same as for your Form on a Table. Again, it was nice of the Form wizard to create all these elements for you.

Master-Detail Report and Form

One of the most popular features in APEX is the Master-Detail form and report wizard. With a single, simple wizard, you can quickly create a report and corresponding forms to manage data stored in a master-detail fashion. You will use this wizard to create a report and forms for the `TICKETS` and `TICKET_DETAILS` tables.

Creating a Master-Detail Report and Form

First, you will create the report and form on application pages 200, 210, and 220. Since you do not yet have those pages created, the wizard will do that for you.

1. Run any page in your application.
2. Click Create in the Developer's Toolbar.
3. Select New Page and click Next.
4. Select Form and click Next.
5. Select Master Detail Form and click Next.
6. See Figure 6-24. Set the Table/View Owner to your schema. Set the Table/View Name to `TICKETS`. When the page refreshes, add all of the columns by clicking on the `>>` icon. Click Next.

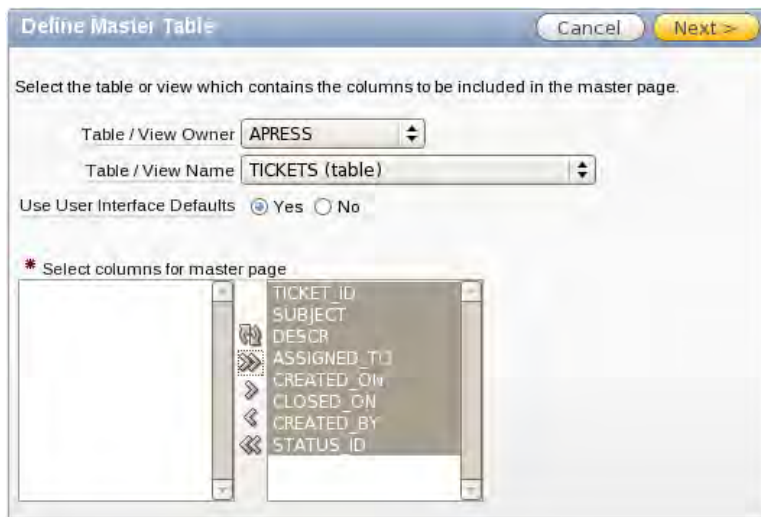


Figure 6–24. Creating the master page

7. Select TICKET_DETAILS for the Table/View Name. When the page refreshes, add the following columns by double-clicking on their name. You should end up with results like those in Figure 6–25.
 - TICKET_DETAILS_ID
 - TICKET_ID
 - DETAILS
 - CREATED_ON
 - CREATED_BY
 - ATTACHMENT

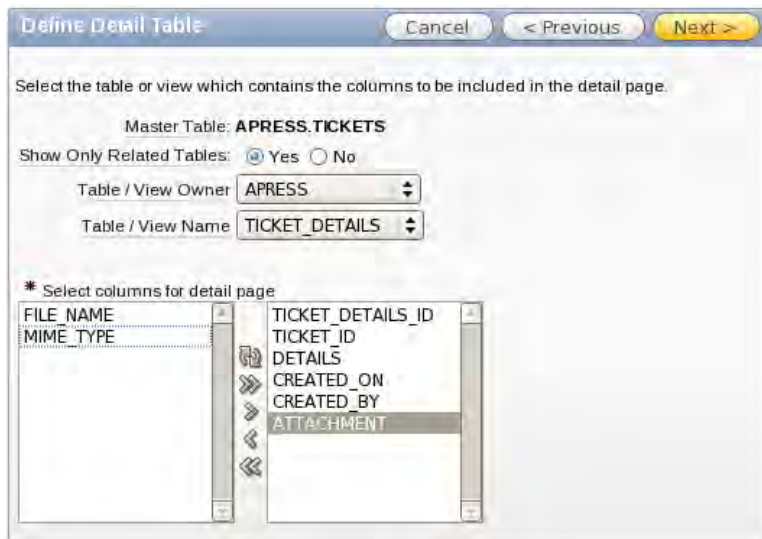


Figure 6-25. Defining the detail table

8. Click Next.
9. Set the Primary Key Source to Existing Trigger for the Master table and click Next.
10. Set the Primary Key Source to Existing Trigger for the Detail table and click Next. Do NOT click Finish at this point.
11. Set Include master row navigation? to Yes, as shown in Figure 6-26. Set the Master Row Navigation Order to CREATED_ON and click Next.

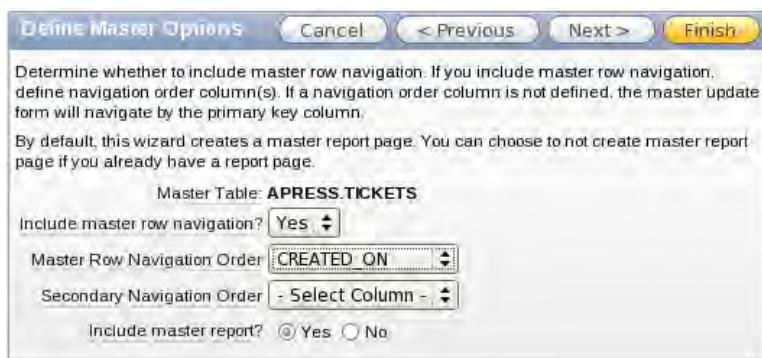


Figure 6-26. Defining master row navigation options

12. Set Build Master Detail with: to Edit detail on separate page and click Next.

13. On the next page, set the items to the values shown in Figure 6–27.

Page Attributes [Cancel] [Previous] [Next]

Master Table: **APRESS.TICKETS**
Detail Table: **APRESS.TICKET_DETAILS**

	* Page	* Page Title	* Region Title
Master	200	Tickets	Tickets
Detail	210	Manage Tickets	Manage Tickets
			Ticket Details
Detail 2	220	Ticket Details	Ticket Details

Breadcrumb: [Breadcrumb]

Figure 6–27. Specifying page attributes

14. Set the Breadcrumb to Breadcrumb.
15. In the Create Breadcrumb Entry region, set the items to the values shown in Figure 6–28.

Create Breadcrumb Entry

Entry Name (Master Report): Tickets

Entry Name (Master Detail Page): Manage Tickets

Entry Name (Detail Form): Ticket Details

Parent Entry: No parent breadcrumb entry
[No parent breadcrumb entry]

Select Parent Entry:

Name	Page
Home	1
Contact Us	3
Create a Ticket	2

row(s) 1 - 3 of 3

Figure 6–28. Creating a breadcrumb entry

16. Click Next.
17. Set the Tab Options field in Figure 6–29 to Use an existing tab set and create a new tab within the existing tab set. When the page refreshes, set the Tab Set to TS1 (Home) and enter Tickets for the New Tab Label and click Next.

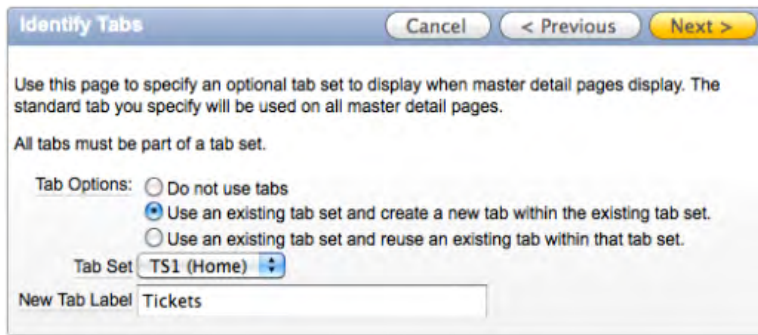


Figure 6–29. Setting your tab options

18. Confirm your selections and click Create.

When the wizard completes, you have a working master–detail form on the TICKETS and TICKET_DETAILS tables, plus a report on the TICKETS table. This is perhaps one report more than you expected, but APEX knows that in most cases, you need the report to select the master-detail record to be edited, so that report gets created at the same time for convenience. The Master-Detail Form wizard created one report and two forms, plus the links and branches for navigation and the processes for performing database transactions. The Tickets report has a link to the Tickets form, which allows editing of ticket master data and lists ticket details. The Ticket Details region on the Manage Tickets page has an edit link to the Ticket Detail form, where the user can add, update, or delete ticket detail information. All of the items, buttons, processes, and even the column links were created by the Master-Detail Form and Report wizard.

Again, while you can build a master-detail form and report manually, the wizard is much faster and certainly more efficient. Now let’s make some adjustments to the report and the forms to suit your requirements.

Modify a Master-Detail Report

Next, you’ll modify the report to add CSV export capabilities, change the sorting options, and modify the date format mask. Then you’ll clean up the two edit forms.

1. Edit page 200 of your application.
2. Edit the Report Attributes by right-clicking on the name of the Tickets region and selecting Edit Report Attributes, as shown in Figure 6–30.

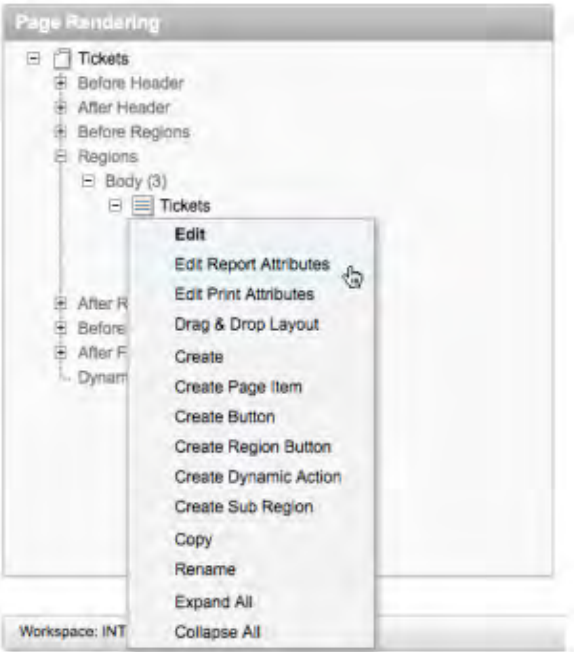


Figure 6–30. Choosing to edit report attributes

- 3. Enable column sorting for all columns by checking the Sort box for each column. Hide the DESCR column from the report by un-checking the corresponding Show checkbox. See Figure 6–31.



Figure 6–31. Editing column attributes

4. Under Layout & Pagination, set Enable Partial Page Refresh to Yes.
5. Under Report Export, set the following options, which you can also see in Figure 6–32:
 - Enable CSV Output: Yes
 - Separator: ,
 - Enclosed By: “
 - Link Label: Export to Excel
 - Filename: tickets.csv

Report Export

Enable CSV output: Yes | Separator: , | Enclosed By: “

Link Label: Export to Excel

Filename: tickets.csv

Figure 6–32. Setting report export options

6. Scroll to the top of the page and edit the CREATED_ON column by clicking on the Edit icon. This will also save the changes you made in steps 2 through 5.
7. In the Column Attributes section for Number/Date Format (Figure 6–33), select this format mask: Monday, 12 January, 2004. Selecting it will return fmDay, fmDD fmMonth, YYYY into the Number/Date Format.

Column Attributes

Display As: Display as Text (escape special characters, does not save state)

Number / Date Format: fmDay, fmDD fmMonth, YYYY

Element Width: | Number of Rows: |

Number of Columns (Radio Group): |

Element Attributes: |

Element Option Attributes: |

Figure 6–33. Selecting a date format mask

Values entered in the Column Attributes region set the format, width, number of rows (for text areas), the number of columns (for radio groups), and other element attributes for the report column.

8. Enter font-weight:bold for the CSS Style field shown in the Column Formatting section (Figure 6–34). Click Apply Changes.

Figure 6–34. Choosing your column formatting options

CSS and HTML formatting directives entered in the Column Formatting region are applied to the report column when the page is rendered.

9. Edit the TICKET_ID column by clicking on its edit icon.
10. In the Column Definition region, set Include in Export to No and click Apply Changes.
11. Run the page to view your changes.

Note that when you sort an APEX report column by date, the report will sort based on the value of the actual date, not the displayed value. This is a built-in feature of APEX. Also, when you export to Excel, the TICKET_ID column is not a part of the resulting CSV file, which is the result of your setting the Include in Export option to No.

Next, you will remove the STATUS_ID and replace it with the corresponding value, pulled into your report by a slight adjustment to your query.

1. Edit page 200 in your application.
2. Edit the Report Attributes again by right-clicking on the Tickets region and selecting Edit Report Attributes.
3. Click the Query Definition sub-tab.
4. Click Add/Remove Columns.
5. See Figure 6–35. Ensure that Table/View Owner is set to your schema name and that Show Related Tables Only is set to Yes.
6. Set the Table/View Name to STATUS_LOOKUP and wait for the page to reload.
7. Select the STATUS column, and move it to the Columns Selected list by clicking on the > icon.



Figure 6-35. Selecting report columns

8. From the Columns Selected list, remove the TICKETS.STATUS_ID column by selecting it and clicking on the < icon.
9. Click Next to verify the Join Conditions. You'll see the dialog in Figure 6-36.
10. Validate that the Join Conditions are correct and click Apply Changes.

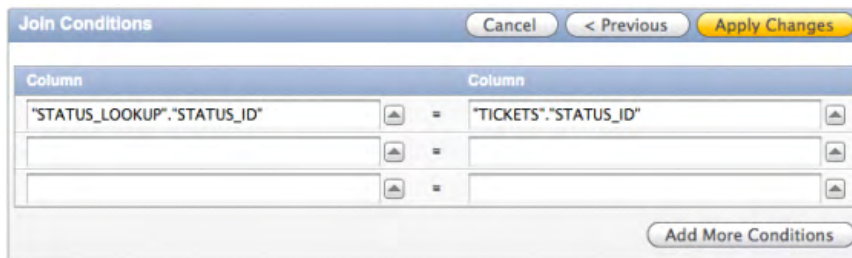


Figure 6-36. Joining the tables

11. Edit the Report Attributes again by right-clicking on the Tickets region and selected Edit Report Attributes.
12. Enable sorting for the Status column by checking the Sort checkbox and click Apply.
13. Click the + symbol next to the Report Columns for the Tickets region. Use your mouse to drag and drop columns within the Column list to reorder them. Reorder the columns so that STATUS is the second column in the list. Drag-and-drop changes will automatically be saved.

Run your application to see the changes to your Tickets report. You should see results like those in Figure 6–37 through 6–39. The Created On and Status values are now more readable, and you can sort by column by double-clicking on the column heading.
















Home Tickets						
Tickets						
Tickets Create						
Edit	Subject	Status	Assigned To	Created On	Closed On	Created By
	Wireless signal not strong enough	CLOSED	SCOTT	Saturday, 25 September, 2010	01-OCT-10	JOHN
	Virus Definitions Dates	CLOSED	SCOTT	Thursday, 23 September, 2010		JOHN
	VPN Client Install Issues	OPEN	DOUG	Friday, 17 September, 2010		JACKIE
	Test Ticket	OPEN		Wednesday, 13 October, 2010		Doug Gault
	Speakers are too soft	OPEN	SCOTT	Wednesday, 15 September, 2010		JERMAINE
	Smartphone will not sync with Outlook	OPEN	SCOTT	Sunday, 19 September, 2010		MICHAEL
	Several dead pixels on screen	PENDING	DOUG	Monday, 20 September, 2010		ALEX
	PC will not turn on	CLOSED	JOHN	Friday, 01 October, 2010		RINGO
	Network drive not being mapped	OPEN	DIMITRI	Monday, 27 September, 2010		GEDDY
	Need to install SP2	OPEN	DIMITRI	Tuesday, 28 September, 2010		ALEX
	Need more memory	OPEN	DOUG	Thursday, 30 September, 2010		GEORGE
	Mouse is not working	OPEN	DIMITRI	Thursday, 16 September, 2010		TITO
	MSIE Crashed 4 times	CLOSED	SCOTT	Wednesday, 29 September, 2010		JOHN
	Keyboard busted	PENDING	JOHN	Tuesday, 14 September, 2010		MICHAEL
	I think I have a virus	OPEN	JOHN	Friday, 24 September, 2010		ROBERT
Export to Excel						
				row(s) 1 - 15 of 21	Next	

Figure 6–37. The Tickets report

Tickets

Cancel Delete Apply Changes < >

Subject Cannot log into E-Mail

***Descr** User called and cannot log into his MS Outlook e-mail Account.

Assigned To SCOTT

Created On 02-OCT-10

Closed On 02-OCT-10

Created By PAUL

Status OPEN

20 of 21

Ticket Details Create

Edit	Ticket Id	Details	Created On	Created By	Attachment
	1	Username is: MJONES	02-OCT-10	SCOTT	[datatype]
	1	User had CAPS lock key on	01-OCT-10	SCOTT	[datatype]

1 - 2

Figure 6–38. The Manage Tickets form

Ticket Details Cancel Delete Apply Changes

Ticket Id 1

Details User had CAPS lock key on

Created On 01-OCT-10

Created By SCOTT

Attachment Browse...

Figure 6–39. The Ticket Details form

Session State

Next, let's add a Search field to your report to allow your users to filter for a specific ticket they may be interested in. Before you do, here's a brief explanation of session state that will help you understand how APEX keeps track of the values associated with a user's session.

Understanding Session State

Session state is what allows APEX to keep track of all the values that belong in a particular user's APEX session. Session state is particularly useful for keeping track of values as a user moves from page to page in the application.

Unlike a stateful database application, where a connection is maintained continuously and all values retain their value until changed, removed, or the session ends, an APEX application does not maintain a continuous connection to the database. APEX is a stateless system—the APEX engine generates HTML pages based on directives stored in the APEX repository. Each page rendering is a stateless transaction. An APEX session ties the stateless HTML pages together.

An APEX session is logically and physically distinct from the underlying database sessions. A database session is stateful and an APEX session is stateless. To illustrate the difference, think of a database session as a phone call on a land line. The parties are connected for the duration of the conversation. Both parties have to invest resources to carry on a conversation. Even if no one is talking, the connection—and the link between the two parties—remains, as shown in Figure 6–40.

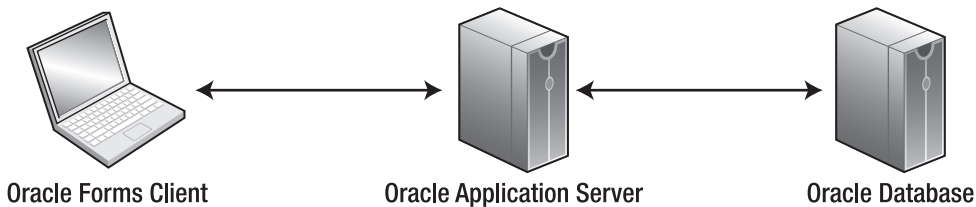


Figure 6–40. Database session communication

Think of an APEX session as a text message. The parties are not directly connected; they push information in one direction at a time, even if the communication is an entire conversation via a series of texts. Figure 6–41 illustrates APEX stateless session communication.

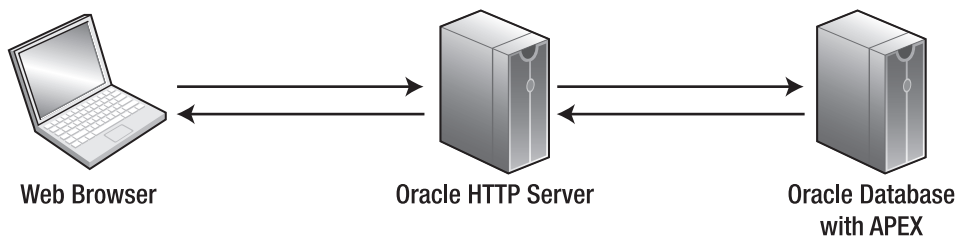


Figure 6–41. APEX session communication

Sharing Database Connections

Multiple APEX users can share the same database connection. There is a one-to-many relationship between APEX users and database sessions. This is why APEX can scale as well as it does—it doesn't need dedicated database sessions, only a database session to use to process a request from a user.

APEX, being stateless, must rely on an external mechanism to manage session state. The APEX engine has a built-in session state management component. This session-state management is an integral part of APEX—it can't be disabled or circumvented.

Each APEX user is assigned a unique session identifier. Session state management functions the same, regardless of how the user authenticates to the system—APEX authentication, database authentication, customer authentication, or public user. Yes, even unauthenticated users are assigned a session identifier. By default, APEX will purge inactive sessions older than 24 hours every eight hours. APEX session state values are stored in a table in the database. The APEX engine recognizes the user by his session identifier and retrieves the appropriate set of session state values for the user's session.

The values of all APEX items, both page items and application items, are tied to this unique session identifier. This identifier is referred to as the APP_SESSION_ID. You can see the session identifier in the URL of most pages in an APEX application. It's highlighted in Figure 6–42.

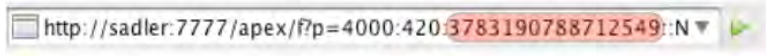


Figure 6–42. APEX session identifier in an APEX URL

Setting and Retrieving Session State

Session state is set by user input items, computations, processes, and PL/SQL code. In PL/SQL, when within an APEX process, you can set an item equal to a value, like so:

```
:P1_ITEM_NAME := 'some value';
```

In PL/SQL, when in a stored procedure, you can use the `apex_util.set_session_state` procedure to set a value in session state, like so:

```
apex_util.set_session_state( 'P1_ITEM_NAME', 'some value');
```

The syntax to retrieve session state for an item varies according to where you are referencing the item.

In templates or regions, tabs, menus or lists, use the following substitution string syntax (and do not forget the trailing dot!):

```
&P1_ITEM_NAME.
```

Use the following syntax within SQL statements:

```
:P1_ITEM_NAME.
```

From PL/SQL, use one of the following two options depending upon what type of block or program unit you are in:

```
Anonymous PL/SQL block:      :P1_ITEM_NAME.
PL/SQL Unit Called from APEX: V('P1_ITEM_NAME')
```

From within conditions, use this syntax:

```
P1_ITEM_NAME
```

Viewing Session State

To view session state, click on the Session link in the Developer's Toolbar. You should see a page like that in Figure 6–43. Then use the Page, Find, and Views parameters to view session state for the application. The drop-down view menu shown in Figure 6–44 allows you to view Page Items, Application Items, Session State, Collections, and All of the Above.

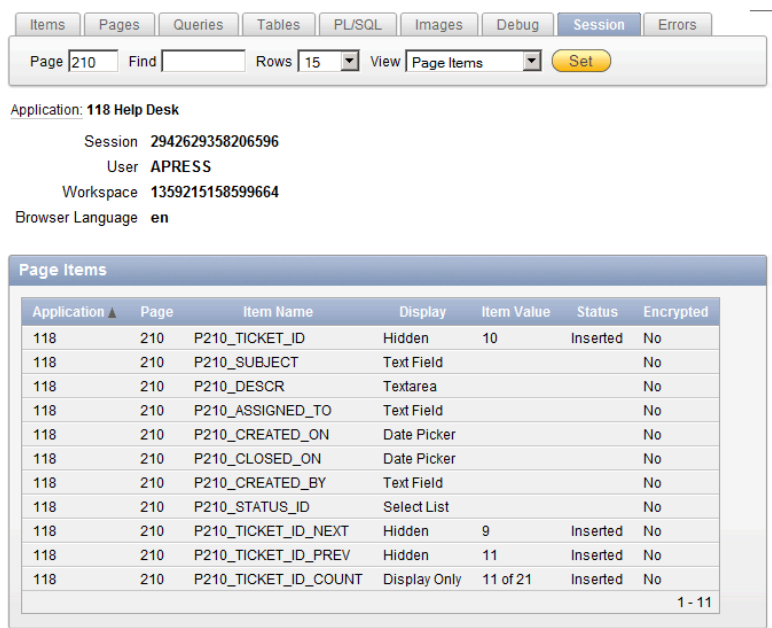


Figure 6–43. Viewing your session state

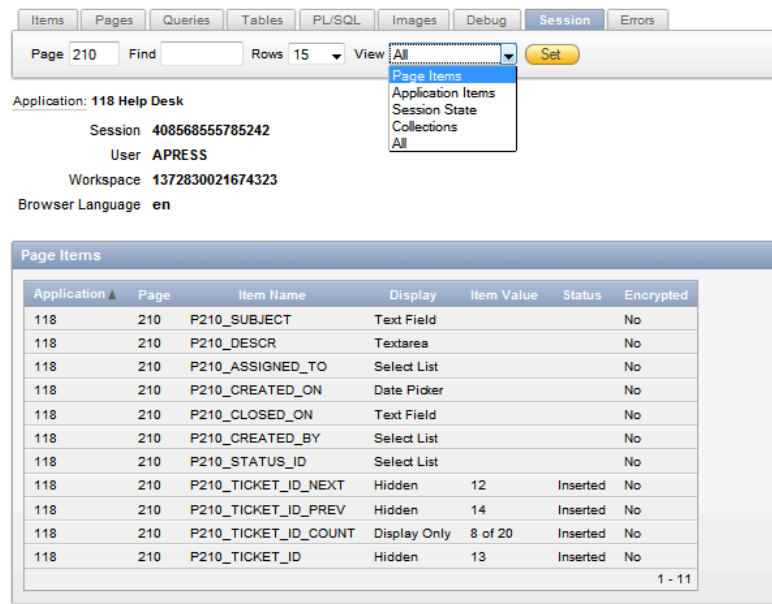


Figure 6–44. Choosing to view page items

APEX Items

There are two types of APEX items: page items, those that are displayed to the user on a page, and application items, which hold values within an application but are not displayed. When referencing item values in queries, you'll want to use bind variables. You may also want to reference some of the built-in items that are available.

Page versus Application items

APEX page items are the user interface controls used to allow users to view and enter data—Text, Text Area, Select List, Checkbox, etc. Page items are associated with a specific page and have a user interface properties associated with them; the item is displayed to the user (or not) according to the user interface properties. Figure 6–45 shows the available APEX page item types. See the APEX documentation for more information on page item types and their attributes.

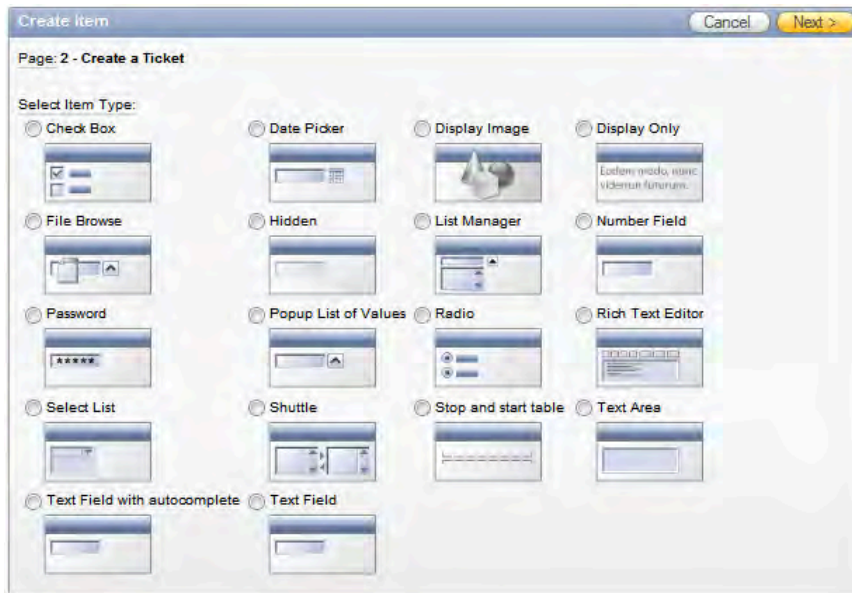


Figure 6–45. APEX page item types

Application items are not associated with a page and have no user interface properties. Application items hold values within an application that are essential but not necessarily displayed. You can use an application item much like a global variable.

The Importance of Bind Variables

When referencing APEX item values, particularly in SQL queries within your APEX application, it's important to think about SQL security basics, including SQL injection. Consider the example of an online form that allows a user to sign on with a username and password, which ultimately executes this query:

```
SELECT COUNT(*) FROM users
WHERE username = '&username'
      AND password = '&password'
```

If you enter this password

```
I_dont_know OR 'x' = 'x'
```

the resulting SQL will be

```
SELECT COUNT(*) FROM users
WHERE username = 'SCOTT'
      AND password = 'I_dont_know' OR 'x' = 'x'
```

This SQL statement will erroneously return 1, indicating True, rather than No data found. The user will be allowed in! Not good. To prevent the injection of unintended SQL, use bind variables in the SQL query, like so:

```
SELECT COUNT(*) FROM users
WHERE username = :USERNAME
      AND password = :PASSWORD
```

Now try entering the following as your password:

```
I_dont_know OR 'x' = 'x'
```

Unless this entire string is specifically your password, the database will return No data found. Your attempt to sneak past the login will fail.

We recommend the use of bind variables whenever possible. They prevent SQL injection and improve SQL performance.

Built In Items

APEX includes several built-in items for referencing key APEX application-wide session state values. These are automatically set by APEX and available for reference by the developer throughout APEX. The most common of these are:

- **APP_ID:** The application identifier of the currently running application.
- **APP_ALIAS:** The application alias of the currently running application.
- **APP_USER:** The currently signed-on user.
- **APP_SESSION:** The session identifier of the currently signed-on user.
- **APP_PAGE_ID:** The currently running page identifier.

APEX URL Syntax

Every APEX page is a call to the APEX engine. Every APEX URL is really a call to a specific page, passing various parameters. Figure 6–46 shows the URL syntax. The leading *f* in the syntax represents the page. The parameters follow the *?p* sequence.

```
f?p=
  APP_ID:
  APP_PAGE_ID:
  APP_SESSION:
  REQUEST:
  DEBUG:
  Clear Cache:
  item1,item2:
  itemValue1,itemValue2:
  printerFriendly
```

Figure 6–46. APEX URL syntax

f?p is the call to the *f* PL/SQL procedure passing the argument *p*. The argument is actually a concatenation of nine arguments combined into one, delimited by a colon. The nine elements of the *p* argument are the same for all APEX page requests. You may omit one or more of the arguments, but you must include the colon delimiters as placeholders.

The elements that comprise the *p* argument are:

- **APP_ID:** The application number or alias.
- **APP_PAGE_ID:** The page number or alias.
- **APP_SESSION:** The APEX session identifier.
- **REQUEST:** The HTML request.
- **DEBUG:** A Debug flag, set to YES or NO or omitted to use the current value of the Debug flag.
- **Clear Cache:** A list of pages for which to clear the cache.
- **Item Names:** A list of APEX item names, separated by commas.
- **Item Values:** A list of APEX item values, separated by commas that correspond in order to the items specified in the Item Names list.
- **Printer Friendly:** A flag that determines whether the page is rendered in Printer Friendly mode.

It is easiest to understand the APEX URL syntax by looking at a few examples. Table 6–1 shows several examples and explains each.

Table 6–1. APEX URL Examples

f?p=&APP_ID.:10:&APP_SESSION.:.:10	Calls page 10 of the current application using the current session and clears the session cache for page 10.
f?p=&APP_ID.:5:&APPSSESSION.:.:NO::P2_ID:1234	Calls page 5 of the current application using the current session, not in Debug mode, setting the value of P2_ID to 1234.
f?p=&APP_ID.:5:&APP_SESSION.:.:YES	Calls page 5 of the current application using the current session in Debug mode.

As you can see, the APEX URL not only supplies directions to the server, it’s also your key to what page is being requested, with what request, and with what values. So how does this URL syntax tie in to your work on the help desk application?

APEX applications store all values in an APEX session, which is securely bound to a specific user and user session. Values stored in this user session can easily be set or read by a developer. Any item—application or page—can be easily referenced from anywhere within your APEX application. Values can be referenced and passed to APEX as part of the p parameter to control which APEX page is rendered and what values are displayed on that page.

As the volume of data in your system grows, you will need a quick way to sort through it and control what data gets passed to what page. You can add a page item and then use the value of that item to filter the SQL statement for the report on page 200 of the application. In fact, an item in APEX can be referenced in a SQL or PL/SQL region, as in the predicate of a query, by using the bind variable syntax (:P1_ITEM_NAME) and as part of the APEX URL.

Getting back to your wizard-generated TICKETS report, you’ll apply what you just learned about session state, APEX items, and the APEX URL to add a new item called P200_SEARCH that the user can use to filter the TICKETS report. After you make these report modifications, you’ll take a closer look at the components and attributes of an APEX report.

Searchable APEX Reports

Reports with edit links let users scan down a list of rows and choose one to modify. Scanning reports works well for reports that are short. But when reports get long, especially more than a page or two, it’s time to add some search functionality to help a user quickly zero in on a record to edit.

Creating a Searchable APEX Report

You have already modified the TICKETS report generated by your Master-Detail Report and Form wizard to add sorting, CSV Export capability, and a readable Status value. As generated, the report has an edit link on the first column, which navigates to a Ticket – Ticket Details master-detail form. For the user to find the correct ticket to edit, you need a Search function. You’ll add a Search item and a Go button to activate the search, and you’ll modify the report query to filter on the Search value.

1. Edit page 200 of your application.
2. Create a new item in the Tickets region by right-clicking on the region name and selecting Create Page Item.

3. Select Text Field and click Next.
4. Enter P200_SEARCH for the item name, as shown in Figure 6–47. Make sure that Tickets (1) 10 is selected as the Region and click Next.

The screenshot shows the 'Create Item' dialog box. At the top, there are three buttons: 'Cancel', '< Previous', and 'Next >'. Below these, the page context is 'Page: 200 - Tickets'. The 'Display As' is set to 'Text Field'. There are three required fields marked with a red asterisk: 'Item Name' with the value 'P200_SEARCH', 'Sequence' with the value '10', and 'Region' with a dropdown menu showing 'Tickets (1) 10'.

Figure 6–47. Creating a search field

5. No changes are needed for the Item Attributes. Click Next.
6. Set the value of Submit when Enter pressed to Yes. Click Next.
7. Click Create Item.

Although you just set the item attributes so that the page will be submitted when the enter key is pressed, it's still a good practice to provide a way to submit the page using the mouse. You'll create a new button that when clicked will process the item value, store it in session state, and then reload page 200.

8. Create a new Button item by right-clicking on the Tickets region name and selecting Create Button. Be careful not to select Create Region Button, as that will place the button at the top of the region as opposed to alongside your text box.
9. Enter P200_GO as the Button name, as shown in Figure 6–48. Leave all other values alone and click Create Button.

Figure 6–48. Creating a Go button for the search function

Next, you'll adjust your report query to apply the P200_SEARCH filter. First, you need to convert your query from a structured report to a SQL report. A structured report is a result returned from the Query Builder. A SQL report is more flexible, as it allows you to enter any valid SQL, rather than rely on a limited set of declarative options. After you convert the report to a SQL query, you'll add a line to the query predicate that will use the value stored in P200_SEARCH as a filter.

10. Edit the Tickets region definition by double-clicking on its name.
11. Under Tasks region on the right side of the page, click on Convert to SQL Query. Click OK when prompted.
12. Edit the Tickets region again by double-clicking on its name.
13. Append the following line to the end of the query and click Apply Changes:
`AND UPPER(subject) LIKE '%' || UPPER(:P200_SEARCH) || '%'`
14. Run your report. Remember to test both the button and pressing Enter while editing the Search field. Both should filter the report correctly.

Any time that you add a Search item to a page, it's a very good idea to also add a Reset Pagination process. This will prevent the APEX reporting engine from losing its place in a result set.

1. Edit page 200 of your application.
2. In the upper right hand section of the page click the Create button and select Page control on this page.
3. Select Process and click Next.
4. Select Reset Pagination and click Next. You should see the dialog in Figure 6–49.
5. Ensure that the scope is set to Current Page and click Create Process.
6. Run your application.

Create Page Process [Cancel] [< Previous] [Create Process]

Page: **200 - Tickets**

Point: **On Submit - After Computations and Validations**

* Name: **Reset Pagination**

* Sequence: **10**

Scope: **Current Page** Pages:

When Button Pressed: **- No Button Condition -**

Condition Type: **- Process Not Conditional -**

[PL/SQL] [item=value] [item not null] [request=el] [page in] [page not in] [exists] [none] [never]

Figure 6–49. Specifying process options

Your Search function should work when the user clicks the Enter button and when the user clicks the Go button. But let's go one more step and alter the Subject column so that the search term is highlighted in red.

1. Edit page 200 of your application.
2. Edit the Report Attributes again by right-clicking on the Tickets region and selecting Edit Report Attributes.
3. Edit the Subject column by double-clicking on the Edit icon.
4. In the Column Formatting region, enter &P200_SEARCH in the Highlight Words element.
5. Click Apply Changes.
6. Run your application and test the search highlight capability.

Now when you enter a Search value, the matching rows are returned with the search term highlighted in red. In just a few minutes you have created a sortable, searchable report for your Ticket System. Let's look at what your report looks like behind the scenes. Figure 6–50 shows the components as seen from Application Builder.

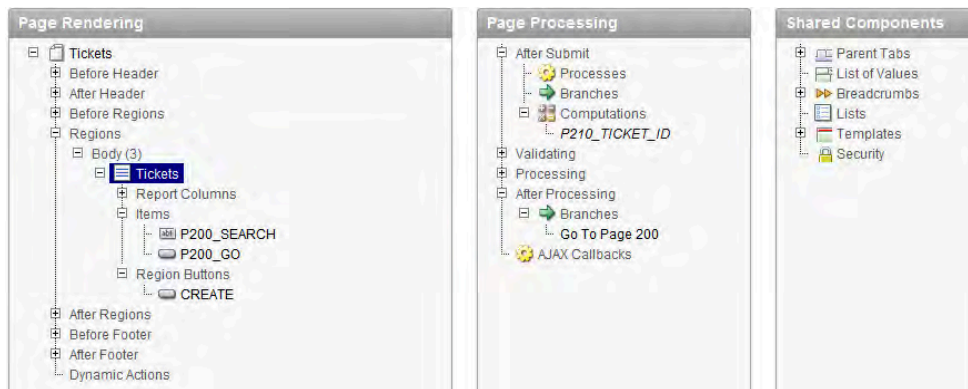


Figure 6–50. Your searchable report as seen from Application Builder

Looking Behind the Scenes – APEX Report

Let's take a closer look at the components and attribute of your TICKETS report. Edit page 200 to view the Page Rendering, Page Processing, and Shared Components regions of the Application Builder. In the Page Rendering region, you have a single Tickets region that contains report columns, the two items you just added for Search capability and a Create button. Double-click on the Tickets region name to open the Region Definition page shown in Figure 6–51.

Region Definition
Report Attributes
Print Attributes

Region: 1 of 1 Name: Tickets
Cancel
Delete
Apply Changes

Show All
Identification
User Interface
Source
Attributes
Header and Footer
Conditions
Security
Configuration
Caching
Customization
Com

Identification

Page: 200 Tickets
Title: Tickets
Type: SQL Query
exclude title from translation

User Interface

Template: Reports Region
Sequence: 10
Parent Region: - Select a Parent -
Display Point: Page Template Body (3. items above region content)
Column: 1

Source

Region Source

```

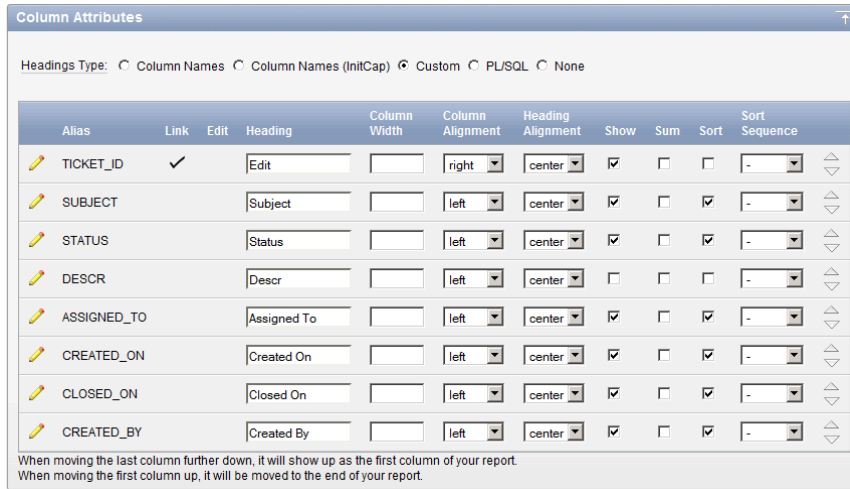
SELECT
  "TICKETS"."TICKET_ID" "TICKET_ID",
  "TICKETS"."SUBJECT" "SUBJECT",
  "TICKETS"."DESCR" "DESCR",
  "TICKETS"."ASSIGNED_TO" "ASSIGNED_TO",
  "TICKETS"."CREATED_ON" "CREATED_ON",
  "TICKETS"."CLOSED_ON" "CLOSED_ON",
  "TICKETS"."CREATED_BY" "CREATED_BY",
  "STATUS_LOOKUP"."STATUS" "STATUS"
FROM
  "TICKETS",
  "STATUS_LOOKUP"
WHERE "STATUS_LOOKUP"."STATUS_ID" = "TICKETS"."STATUS_ID"
AND UPPER( subject) LIKE '%' || UPPER( :P200_SEARCH) || '%'

```

Figure 6–51. Your Tickets report region source with the search filter

Here you see that this region type is SQL Query and it is using the Reports Region template. The source for this region is your SQL query on the TICKETS table with the modified WHERE clause to add the filter on the P200_SEARCH item, referencing P200_SEARCH as a bind variable.

Moving to the Report Attributes tab, you see a series of attribute regions. There is a list of your table columns in the Column Attributes region, shown in Figure 6–52.



Alias	Link	Edit	Heading	Column Width	Column Alignment	Heading Alignment	Show	Sum	Sort	Sort Sequence
TICKET_ID	✓		Edit		right	center	✓			-
SUBJECT			Subject		left	center	✓		✓	-
STATUS			Status		left	center	✓		✓	-
DESCR			Descr		left	center				-
ASSIGNED_TO			Assigned To		left	center	✓		✓	-
CREATED_ON			Created On		left	center	✓		✓	-
CLOSED_ON			Closed On		left	center	✓		✓	-
CREATED_BY			Created By		left	center	✓		✓	-

When moving the last column further down, it will show up as the first column of your report.
When moving the first column up, it will be moved to the end of your report.

Figure 6–52. Column attributes for the Tickets report

This region allows you to adjust the heading, column width, column alignment, and heading alignment; you can also decide if the column is shown, if a sum is required, or if you want to enable sorting on the column and if so, to define a sort sequence. Arrow keys on the right of this region allow a developer to reorder columns up and down the list.

Click on the pencil icon to go to the Column Attributes page, where you can view and edit many more column attributes, organized in a series of regions. These Report and Column attributes will be described in greater detail later in this chapter.

In the Shared Components region, you see the expected objects for the parent tab, the breadcrumb, and the page, region, report, label, and button templates. It's nothing new, but you're glad the wizard built these for you.

Next, let's focus on your Tickets and Tickets Details forms, the other components generated by your Master-Detail Form and Report wizard.

Looking Behind the Scenes – APEX Master-Detail Forms

Edit page 210 to view the Page Rendering, Page Processing, and Shared Components regions of the Application Builder. You should see results similar to those in Figure 6–53.

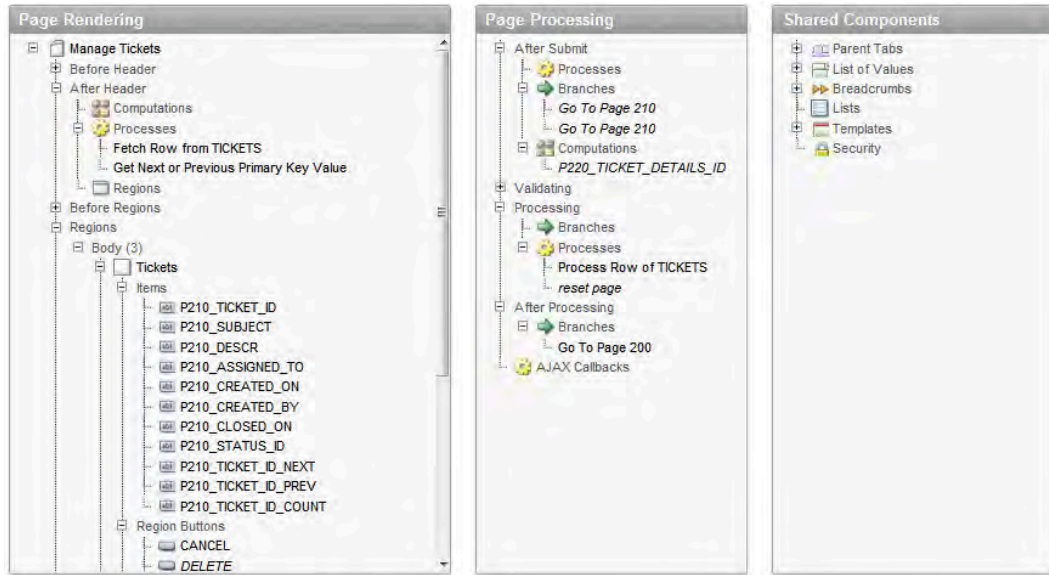


Figure 6–53. Application Builder showing components for the Master-Detail report

In the Page Rendering region, you have two After Header processes, a Ticket HTML region that contains your form items, and a Ticket Details report region.

The two After Header processes, Fetch Row from TICKETS and Get Next or Previous Primary Key Value, do exactly what their name implies. The Fetch Row from TICKETS process fetches a row from the TICKETS table for display in the form when the page passes a TICKETS_ID. The Get Next or Previous Primary Key Value process gets the next or previous TICKETS_ID value in the series and fires in conjunction with the Next and Previous buttons on your Master-Detail page.

The Tickets region indeed holds on APEX item for each of the TICKETS columns you selected to include in your Master-Detail form, as well as buttons for cancel, delete, save, create, next, and previous operations.

The Ticket Details region is a Report region that displays your ticket details and a Create button that redirects you to page 220 for creating additional ticket details.

In the Page Processing region, you see two After Submit branches that return you to this same page, an After Submit P220_TICKET_DETAILS_ID computation, two processes, Process Row of TICKETS and Reset page, and an After Processing branch to page 200. The After Submit computation gets the next TICKET_DETAILS_ID upon clicking the Create button in the Ticket Details region. The new TICKET_DETAILS_ID is passed to page 220, the Ticket Details form. The Process Row of TICKETS process performs the database DML operations for insert, update, and delete operations on the TICKETS table. The Reset Page process resets (clears) the elements of the page when the Delete button is clicked.

The After Processing branch to page 200 redirects the user to page 200, your TICKETS list, upon successful processing.

The Shared Components region includes the by-now familiar APEX elements for your page tabs, lists of values, breadcrumb, and templates.

Moving to page 220, the Ticket Details form, in the Application Build Page Processing region you see elements that look similar to those for your TICKETS form on page 210 (see Figure 6–54).

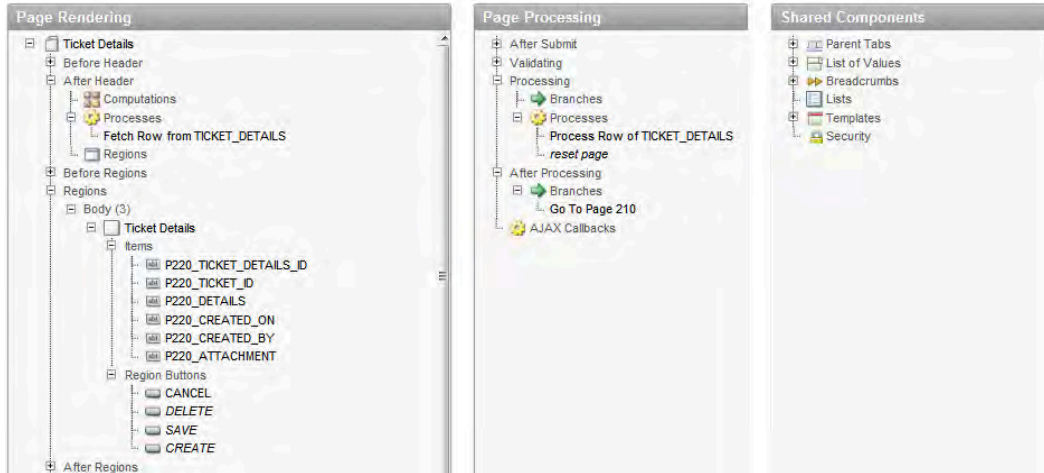


Figure 6–54. Your Application Builder showing components for your Ticket Details form

The Page Rendering region includes an After Header Fetch row from TICKET_DETAILS process, and HTML region that contains items for each of the TICKET_DETAILS columns you selected to include in your Master-Detail form, and buttons for processing.

The Page Processing region includes a Process Row of TICKET_DETAILS process for handling inserts, updates, and delete on the TICKET_DETAILS table, a Reset Page process to clear the rows upon a Delete transaction, and a Go to Page 210 branch that returns the user to the Tickets page upon completion of a Ticket Details transaction.

The Shared Components region on your Ticket Details page includes your page tabs, breadcrumbs, and templates.

Wow! Your Master-Detail wizard created a lot—a fully-functional report with Master-Detail forms, all with no code on your part. This Master-Detail example underlines the time-saving value of the APEX wizards in generating APEX components, particularly when creating more complex and multi-page components of your application.

More on APEX Forms

When creating forms, the APEX wizards will do about 80 percent of what you want them to do. The last 20 percent of fine-tuning is up to you, the developer. You'll make a number of small changes to your Tickets and Ticket Details forms, with the overall goal of increasing usability.

Item Layout

APEX provides three ways to adjust item layout: the Drag & Drop Layout tool, adjusting certain item attributes settings, and dragging items in the tree view. You'll use all of these methods to adjust your Tickets and Tickets Details forms.

APEX lays out Form items using standard HTML tables. Think of your APEX page as grid where items are placed either next to one another or on top of one another. This grid layout seems limiting, but you can rearrange items using the Drag & Drop Layout tool, dragging them up or down in the tree view in APEX 4.0, or adjusting item attributes to force items into place.

The APEX Drag & Drop Layout tool is activated by right-clicking on the region and selecting Drag & Drop Layout. The grid-based interface allows you to rearrange items by dragging and dropping on the grid. You can even add and name new items using an item panel to choose the desired item type. Not all layout options can be manipulated with the Drag & Drop tool, but this is a fast method of achieving a desired form layout or at least getting most of the way there, without the tedious task of manually editing the attributes of each individual item. You'll use the Drag & Drop tool to move your Assigned To, Created On, and Created By items to a single row.

You'll begin adjusting your Tickets form layout by altering the item P210_CREATED_ON so that it uses the date format mask that you specified when you create the application. You'll also set a default value so that it's automatically populated with today's date. Then you'll set it so that it always displays in read-only mode, preventing users from making any changes.

1. Edit page 210 of your application.
2. Edit the item P210_CREATED_ON by double-clicking its name.
3. In the Settings region, delete any entry that is in the Format Mask field. This will force the field to use the Application Data Format Mask you defined when you created your application.
4. Under the Default Value section in Figure 6–55, enter SYSDATE as the Default Value and set the Default Value Type to PL/SQL Expression.

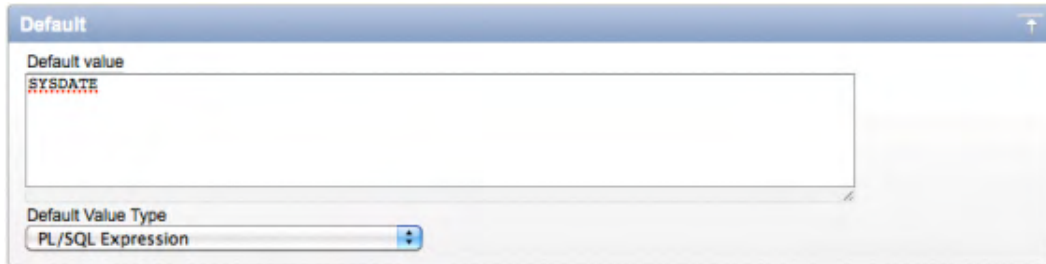


Figure 6–55. Specifying a default value for a date

5. Under the Read Only Display Settings section in Figure 6–56, set the Read Only Condition Type to Always.



Figure 6–56. Setting the read-only condition

6. Scroll to the top of the page and click Apply Changes.

You're going to set P210_CLOSED_ON to also use the application date format mask. In order to reduce errors, you'll use a little known HTML attribute to make the actual input field read-only. The user will then be forced to use the date picker popup.

7. Edit the item P210_CLOSED_ON by double-clicking on its name.
8. In the Settings region, delete any entry that is in the Format Mask. This will force the field to use the Application Data Format Mask you defined when you created your application.
9. In the Displayed region, enter 12 for the width.
10. In the Element region, which you see in Figure 6–57, add the following text immediately after the existing text in the HTML Form Element Attributes field: `readonly="readonly"`

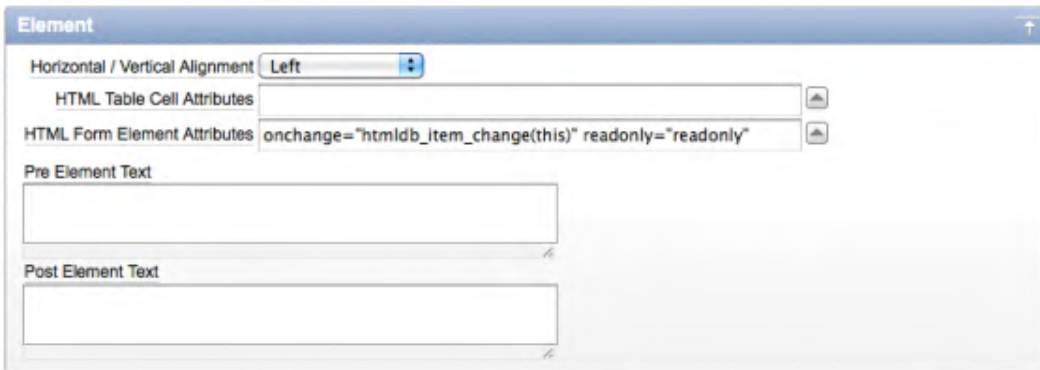


Figure 6–57. Adding an HTML form element

11. Click Apply Changes.

Now you'll rearrange the items on the page so that they are not in a single column, but rather arranged where multiple items are in the same row.

1. Edit page 210 and right-click on the Tickets region name. You'll see the menu in Figure 6–58. Select the Drag & Drop Layout option from the context menu.

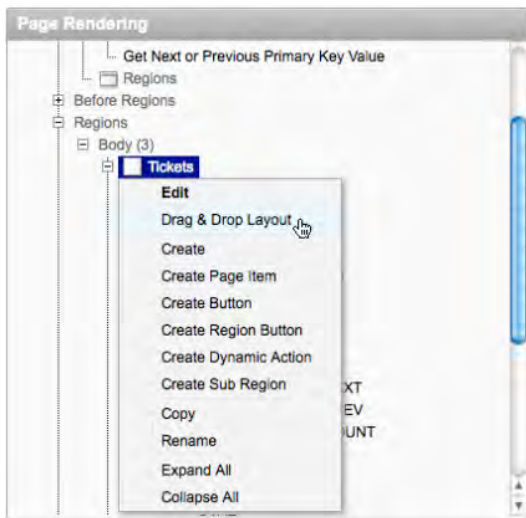


Figure 6–58. Select the Drag & Drop Layout

2. Rearrange the items so that P210_ASSIGNED_TO, P210_CREATED_ON, and P210_CREATED_BY are all on the same line, as illustrated in Figure 6–59.

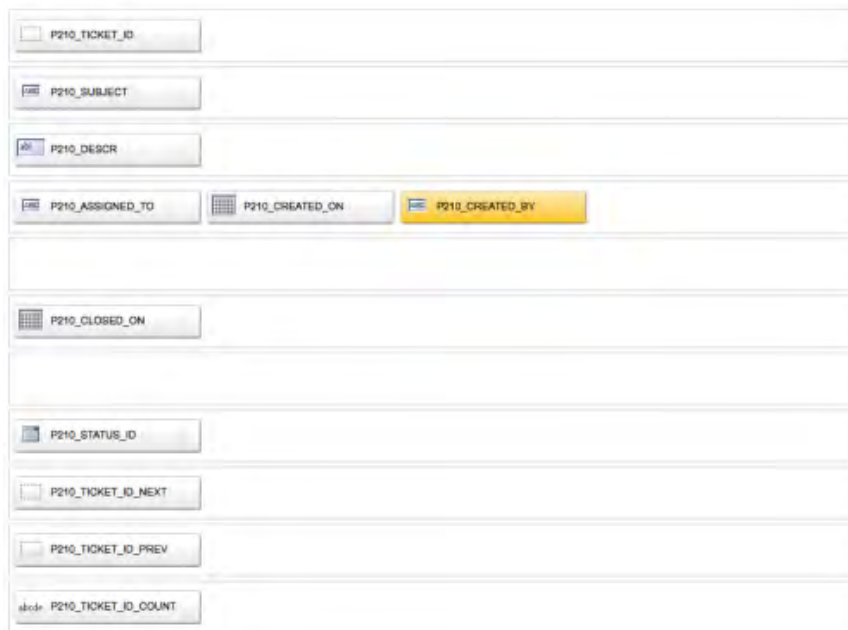


Figure 6–59. Using the Drag & Drop Layout tool

3. When complete, click Next and Apply Changes.

Next, you'll tie the LOVs that you created in Chapter 4 to the P210_ASSIGNED_TO and P210_CREATED_BY items on the form. You'll tie in the LOVs prior to fine-tuning your layout, so your layout is based on the select list items.

1. Edit page 210 of your application.
2. Edit the item P210_ASSIGNED_TO by double-clicking on its name.
3. In the Name section, set Display As to Select List.
4. Under the List of Values section (see Figure 6–60), set the Named LOV to TECHS, set Display Extra Values to No, set Display Null Value to Yes, enter - Select a Tech - for Null Display Value, and then click Apply Changes.

Figure 6–60. Setting List of Values attributes

5. Edit the item P210_CREATED_BY by double-clicking on its name.
6. In the Name section, set the Display As to Select List. Under the List of Values section, set the Named LOV to USERS, and then click Apply Changes.
7. Run your application.

You should see results like those in Figure 6–61. Notice how the Description and Subject fields are pushing out the Created On field.

Figure 6–61. The Tickets form using the new field placement

If you take a look at the page using the Web Developer Firefox add-on and outline the table cells, it will look like Figure 6–62.

Figure 6–62. The Tickets form with table cells outlined

Notice that there is a lot of white space between Assigned To and Created On. You can resolve this by changing the number of columns that the Subject & Description fields take up by altering their ColSpan value. This will allow the Created On field to begin immediately after the Assigned To field.

8. Edit the item P210_SUBJECT by double-clicking on its name. In the Displayed region, as shown in Figure 6–63, then enter 4 for ColSpan and click Apply Changes.

The 'Displayed' window shows the following configuration:

- Sequence: 20
- Region: Ticket Maintenance (1) 0
- Begin On New Line: Yes
- Field: Yes
- ColSpan: 4
- Row Span: 1
- Width: 60
- Maximum Width: 255

Figure 6–63. Altering ColSpan to reduce white space

9. Edit the item P210_DESCR by double-clicking on its name.
10. In the Displayed region, enter 4 for ColSpan and click Apply Changes.
11. Once again, run your application and notice the difference in how the items are laid out on the page now. You should see results like those in Figure 6–64.

The 'Tickets' form displays the following data:

- Subject: BSOD after rebooting
- Description: Blue Screen of Death every time system is rebooted
- Assigned To: Doug
- Created On: 14-JUL-2010
- Created By: Neal
- Closed On: [Calendar icon]
- Status Id: OPEN

Buttons at the top right: Cancel, Delete, Apply Changes, <<, >>.

Figure 6–64. Corrected layout for the Tickets form

Notice the triangular icon at the bottom right of the Description region. Click and drag this icon down and to the right, increasing the size of the Description text area. Observe that the Created On and Created By fields float to the right as size of the Description text area increases, as illustrated in Figure 6–65.

The 'Tickets' form displays the following data:

- Subject: [Empty]
- Description: [Empty]
- Assigned To: - Select a Tech -
- Created On: 11-JAN-2011
- Created By: Alex
- Closed On: [Calendar icon]
- Status: CLOSED

Buttons at the top right: Cancel, Create.

Figure 6–65. Floating Created On and Created By items in Tickets form

In this case, because of the resizable text area, the use of ColSpan to align elements is not sufficient. To prevent the floating columns caused by resizing the text area, you will add a page item of type Stop and Start Table just after your Description item. The Stop and Start Table page item adds HTML tags to stop the current HTML table and opens a new one. Doing so resets the table column widths, which improves your item alignment, as you can see in Figure 6–66.

1. Edit Page 210. Right-click on the Tickets region name and select Create Page Item.
2. Enter P210_STOP_START_TABLE for the item name, 35 for the sequence (as shown in Figure 6–66), and click Create Item.
3. Run your application and notice the difference in how the items are now aligned alongside each other, as shown in Figure 6–67.

Regardless of how the user resizes the Description text area, the Created By and Created On fields remain aligned.

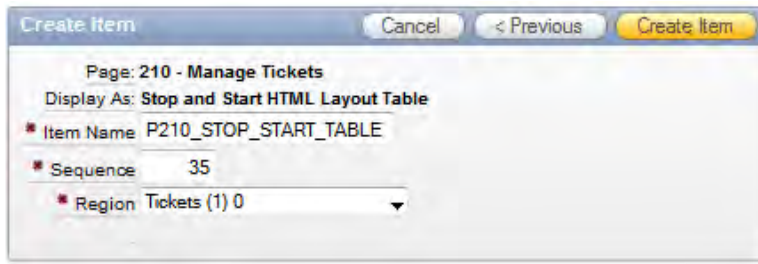


Figure 6–66. Adding Stop and Start Table item

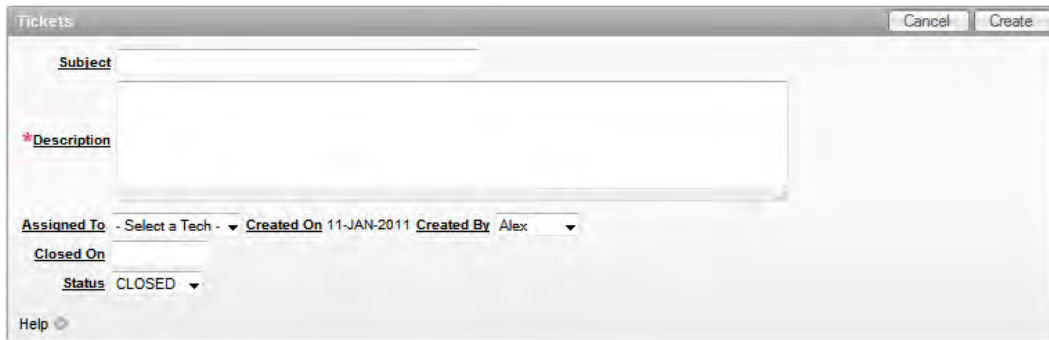


Figure 6–67. Tickets form with corrected item alignment

Master-Detail Cleanup

There are a few more minor tweaks that you need to make to your Master-Detail report and form. Let's start off by hiding the TICKET_ID column from the detail report and form. At the detail level, TICKET_ID is the foreign key and should not be an editable item.

1. Edit page 210 of your application.
2. Edit the Ticket Details report attributes by right-clicking on the region name and selecting Edit Report Attributes.
3. Hide the TICKET_ID column by unchecking the value in the Show column. You'll see the form in Figure 6–68.
4. Enable sorting for the DETAILS, CREATED_ON, and CREATED_BY columns by checking the corresponding checkbox in the Sort column.



Alias	Link	Edit	Heading	Column width	Column Alignment	Heading Alignment	Show	Sum	Sort	Sort Sequence
TICKET_DETAILS_ID	✓	Edit			right	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	-
TICKET_ID		Ticket Id			right	center	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	-
DETAILS		Details			left	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	-
CREATED_ON		Created On			left	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	-
CREATED_BY		Created By			left	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	-
ATTACHMENT		Attachment			left	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	-

Figure 6–68. Specifying Sort columns

5. Click Apply Changes.

Lastly, you'll make a few small changes to the items on Page 220.

1. Edit Page 200 of your application.
2. Edit the item P220_TICKET_ID.
3. Under the Name region, set Display As to Hidden and click Apply Changes.
4. Edit the item P220_DETAILS.
5. Under the Name region, set Display As to TextArea.
6. Under the Displayed region, set the height to 5 and click Apply Changes.
7. Edit the item P220_CREATED_ON.
8. In the Settings region, delete any entry in the Format Mask field. This will force the field to use the Application Data Format Mask you defined when you created your application.
9. Under the Default Value section in Figure 6–69, enter SYSDATE as the Default Value. Then set the Default Value Type to PL/SQL Expression.

The image shows a 'Default' dialog box. It has a title bar with a close button. Inside, there is a text field labeled 'Default value' containing the text 'SYSDATE'. Below this field is a dropdown menu labeled 'Default Value Type' which is currently set to 'PL/SQL Expression'.

Figure 6–69. Specifying the default date

10. Under the Read Only Display Settings section, set the Read Only Condition Type to Always (see Figure 6–70).

The image shows a 'Read Only' dialog box. It has a title bar with a close button. Inside, there is a dropdown menu labeled 'Read Only Condition Type' which is set to 'Always'. Below this dropdown is a list of other possible values: '[PL/SQL]', '[item=value]', '[item not null]', '[request=e1]', '[page in]', '[page not in]', '[exists]', '[none]', and '[never]'. At the bottom, there is a text field labeled 'Read Only Element Table Cell(s) Attributes' which is currently empty.

Figure 6–70. Specifying a read-only condition type

11. Scroll to the top of the page and click Apply Changes.
12. Edit the item P220_CREATED_BY. You'll see the form in Figure 6–71.
13. Set the Display As to Select List, and under the LOV section, set the Named LOV to TECHS, set Display Extra Values to No, set Display Null Values to Yes, enter - Select a Tech - for Null Display Value, and then click Apply Changes.
14. Run the application.

Figure 6–71. Controlling the LOV

Your Master-Detail report and form are now complete. Using the Master-Detail form and report wizard, you generated a report and Master-Detail form on the TICKETS and TICKET_DETAILS tables. You modified the report to contain a user-friendly status value, sortable columns, and your preferred date formats. You modified the Tickets and Ticket Details forms to order items on the page, use text areas, and select lists. Along the way, you reviewed the APEX components that comprise your report and forms, as well as the form, report, and column attributes available for customizing your forms and reports to suit your needs.

APEX Help

Providing help to end users is an often forgotten and typically tedious task. As developers, you just typically take the easy route and skip it altogether. Or the task gets minimized or cut at the end of a project. While APEX cannot magically incorporate help into your applications, it does make it a lot easier for you, as a developer, to do.

Adding a Help Text Region

The APEX Help Text region will automatically display any associated help text for a given page and its items. It can be placed on any page, including page zero. While you can choose a region template for a Help Text region, there is no way to change the style of the actual text. You'll add a Help Text region to page 210 as a sub region to the Master Edit region.

1. Edit page 210 of your application.
2. Create a new Region by clicking the Create button in the upper right of the page and selecting Region on this page from the drop-down menu.
3. Select Help Text and click Next. You'll see the form in Figure 6–72.
4. Enter Help for the title, set the region template to Hide and Show Region, set the parent region to Tickets(1) 0, and click Next.

Figure 6–72. Creating a Help Text region

5. Click Create Region.

Notice that when you run Page 210, you will see the Help rendered with a “>” next to it at the bottom of your Tickets region. The newly created Help region was created as a sub-region, therefore it appears within its parent region. Clicking the “>” will expand the region; thus, the Help text is only displayed when the user explicitly requests it.

Seeding Help Text

Notice that there is only one item, Description, showing in the Help region. This is due to the fact that some help text was added to this region when the User Interface Defaults were defined. Help text defined in the User Interface Defaults is automatically pulled into any form that is built using those UI defaults. You can manually add help text by editing each item, and you can seed any APEX item that does not have help text with a single click.

1. In the upper right section of the Application Builder, click on the Application Utilities icon, as shown in Figure 6–73, to go to the Application Utilities home page.

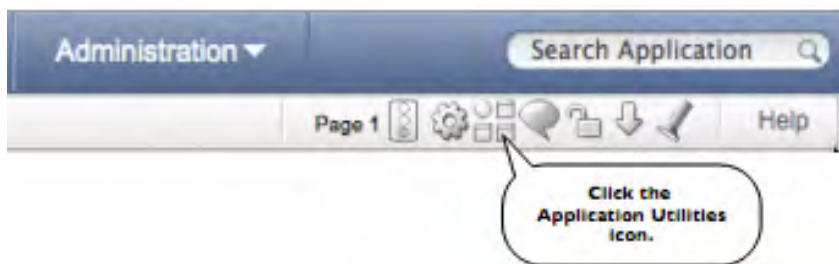


Figure 6–73. Locating the Application Utilities icon

2. In the Page Specific Utilities region on the right side of the page, click on Item Utilities.
3. Click on Grid Edit of all Item Help Text.
4. Click on Seed Item Help Text.

5. Enter NEED HELP TEXT for Default Help Text in the Seed Item Help section, as shown in Figure 6–74, and click Apply Changes.

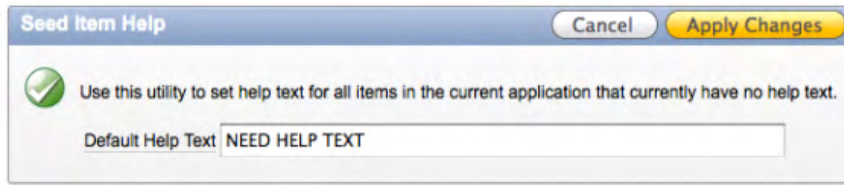


Figure 6–74. Seed item help

6. Click Cancel. The help text has been seeded; you just need to go back a page to see it.
7. Enter 210 for the minimum page number and click Go.

At this point, you will be viewing all of the help text for any item on page 210 or greater in a single interface. Feel free to change some values for any of the items that reside on page 210 to see them in the Help region.

Note that if you click on the individual item labels, a pop-up window will appear displaying the help specific to that item.

The APEX Help region will automatically display the help text for a given page and its associated items. Display of the Help text is managed by APEX behind the scenes. While not very robust—there is no way to alter the look and feel of the region with templates or otherwise—there is now no excuse for not adding Help to your application.

Declarative Blobs

In Oracle, BLOB stands for Binary Large Object and is a data type designed to store files. APEX has streamlined how you can manage BLOB columns with a new feature called Declarative BLOBs. The APEX wizards will recognize a BLOB column and automatically alter the according APEX item and report to interact seamlessly with the column. Why do you care about BLOB columns? Using BLOB columns allows you to easily upload and download files, such a documents, spreadsheets, and images, into your applications.

Plan ahead when using the Declarative Blob feature; at design time, include these columns in the tables that will use the Declarative Blob feature:

- **FILENAME:** Stores the actual file name that is used when a user downloads the file.
- **MIME_TYPE:** Stores the type of the file so that browsers know which application to launch (Word for .doc, Excel for .xls, etc.)
- **LAST_UPDATED:** Stores the date the BLOB was last updated.
- **CHARACTER_SET:** Stores the character set of the BLOB, essential for reading data out of it.

These columns are essential for reading data out of the BLOB when needed. APEX reads from the Number/Date format column attribute to map these attributes to the Blob column.

If you add a BLOB column after creating a report or form using a wizard, you will have to manually enter the Format Mask to enable the Declarative BLOB features. You can enter the bare minimum, save the item, and then use the BLOB pop-up page shown in Figure 6–75, which will be enabled.

Figure 6–75. Declarative BLOB pop-up

Since you added a BLOB column to the TICKET_DETAILS table when you ran the SQL script, there are several things you need to do to utilize declarative BLOBs. First, you have to map the FILE_NAME and MIME_TYPE columns. Let's address the form on page 220 first.

1. Edit page 220 of your application.
2. Edit the item P220_ATTACHEMENT. In the Settings section, you'll see the form in Figure 6–76.
3. In the Settings region, enter MIME_TYPE for the MIME Type Column, FILE_NAME for the Filename Column, Download for the Download Link Text.

Figure 6–76. Specifying BLOB settings

4. In the Source region, if there are any colons following the word ATTACHEMNT, remove them (see Figure 6–77).



Figure 6–77. Setting the column source type and value

5. Click Apply Changes.

Next, you'll alter the report on page 210.

1. Edit page 210 of your application.
2. Edit the Ticket Details region by double-clicking on its name.
3. Under the Tasks region, click Convert to SQL Query. When prompted, confirm your action by clicking OK.
4. Edit the Ticket Details region again by double-clicking on its name. You'll see the form in Figure 6–78.
5. Locate and open the file `ch6_report.txt`, which can be found where you extracted the class files earlier, and copy the contents into the Region Source region, replacing all text that is currently there.

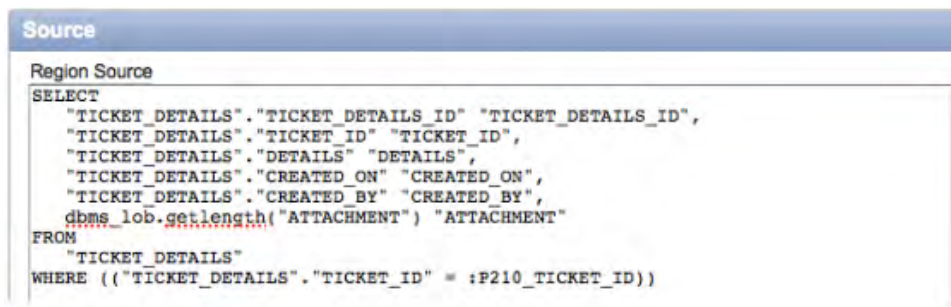


Figure 6–78. Entering the report query with a BLOB column

6. Click on the Report Attributes tab. This will save all changes made to the Region definition.

7. Edit the ATTACHMENT column. You'll see the Column Display Attributes section in Figure 6–79.
8. Locate and open the file `ch6_report_format.txt`, which can be found where you extracted the class files in Chapter 2, and copy its contents into the Number/Date Format field, which replaces all text that is currently there. Then click Apply Changes.

The screenshot shows the 'Column Attributes' dialog box. The 'Display As' dropdown is set to 'Display as Text (escape special characters, does not save state)'. The 'Number / Date Format' field contains the text 'DETAILS:ATTACHMENT:TICKET_DETAILS_ID::MIME_TYPE:FILE_NAME::inline:Download'. Below this, there are fields for 'Element Width' and 'Number of Rows'. At the bottom, there are sections for 'Element Attributes' and 'Element Option Attributes'.

Figure 6–79. *Modifying the BLOB column attributes*

Run your application. Test the file upload and download capabilities by attaching a file to one of the Ticket Detail records and then downloading it from the report.

This ability to easily upload and download files in APEX is extremely useful in building web applications where users need to upload and download data for whatever purpose. The declarative BLOB feature of APEX makes it simple for developers to add upload and download capabilities to an application.

Summary

You have reviewed most of the APEX Forms and Reports types and walked through building various forms and reports for your Tickets System using the APEX form and report wizards. Along the way, you have learned about APEX items, session state, the APEX URL syntax, adding Help to APEX page, and incorporating upload and download functionality by leveraging the declarative BLOB feature. That's a lot to digest, but the APEX wizards have done most of the work for you.

The common theme is that the APEX form and reports wizards are huge time-savers for developers, creating all of the objects—items, buttons, branches, processes, etc.—needed for a working form or report. You were then able to alter the created objects to quickly customize the form or report to suit your needs.

Still, you have not strayed far from what APEX builds for you, and you have covered only the simplest types of forms and reports. In the next chapter, you will look at more complex types of APEX forms and reports, also generated by wizards.



Forms and Reports – Advanced

This chapter focuses on more complex types of forms and reports; it also introduces charts and maps. Although these are more complex types of forms and reports, they are most often created by using the APEX Forms and Report wizards.

In the sections that follow, you'll learn how to use the APEX Form and Report wizards to add pages to your Help Desk application to manage multiple tickets on a single page, allow some interactive analysis of your ticket data, and visualize your tickets by date and status. To do so, you will create a tabular form, an interactive report, a calendar, and a pie chart, each demonstrating one of the more advanced types of APEX forms and reports.

Tabular Forms

Tabular forms allow users to edit both rows and columns of data at once, much like a spreadsheet. The developer can choose a different element type for each column—text box, text area, select list, check box, radio group, etc. Users can make changes to multiple data elements and submit them as a single transaction. APEX tabular forms will handle inserts, updates, and deletes—all with no code!

The APEX wizards create all of the required elements for a fully operational tabular form. Like all APEX forms, there is no logical relationship between items that make up a tabular form. Once the wizard creates the items, they are indistinguishable from other APEX page items and can be modified independent of one another. However, we recommend exercising caution when making modifications to items generated by an APEX wizard. Doing so can cause the tabular forms to become inoperable.

You can bypass the wizard and create your own tabular forms. As your application gets more sophisticated, you may find it more efficient to create your forms manually. However, we focus on the wizard approach in this book.

Creating a Tabular Form

You'll create a new page that contains a tabular form based on the TICKETS table. The form will allow multiple tickets to be edited on the same page. You will then alter the display properties of the tabular form columns.

1. Edit any page in your application.
2. Click on the Create button in the upper right of the page and select New Page from the menu.
3. Select Form and click Next.
4. Select Tabular Form and click Next.

5. Select your schema for the Table/View Owner, set Allowed Operations to Update, Insert and Delete, and click Next.
6. Enter TICKETS for the Table/View Name and click Next.
7. Select all of the columns as shown in Figure 7–1, and click Next.

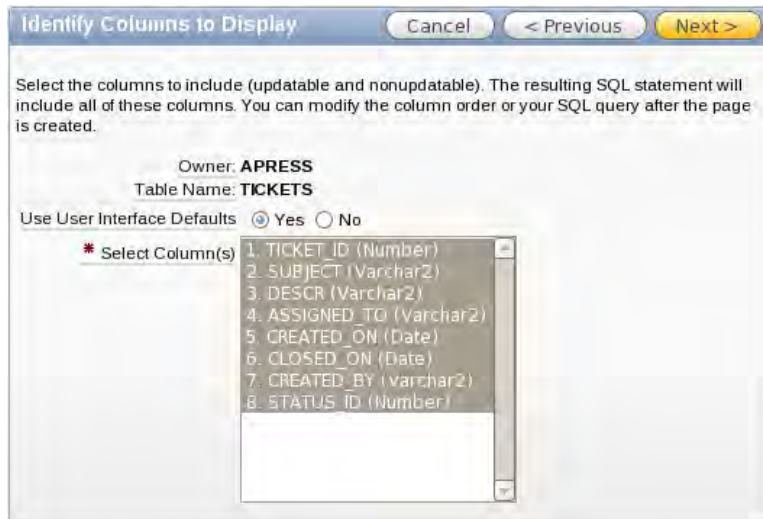


Figure 7–1. Selecting columns for a tabular form

8. Set Primary Key Column 1 to 1. TICKET_ID (Number) and click Next.
9. Set the Source Type to Existing Trigger and click Next.
10. Select all columns as Updateable Columns, as shown in Figure 7–2, and click Next.

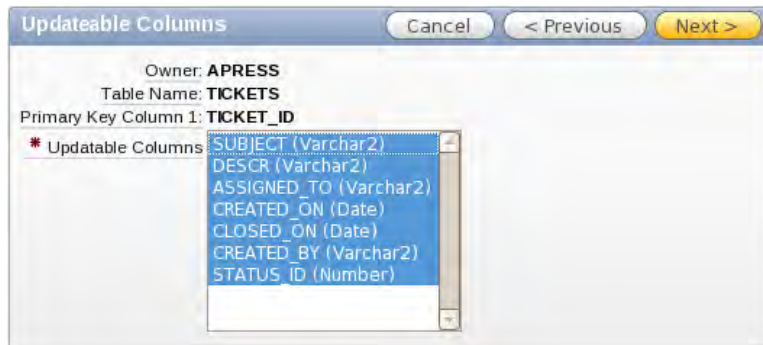


Figure 7–2. Selecting updateable columns for a tabular form

11. Enter 230 for the page number and Manage Multiple Tickets for the page name and page title, as shown in Figure 7–3.
12. Set the Breadcrumb to Breadcrumb.
13. When the page refreshes, set the entry name to Manage Multiple Tickets and the parent entry to Tickets by clicking on the Tickets link (as shown in Figure 7–4), and click Next.

Figure 7–3. Identifying page and region attributes for a tabular form

Figure 7–4. Creating a breadcrumb entry for a tabular form

14. For Tab Options, select Use an existing table set and reuse an existing tab within that tab set; when the page refreshes, set the Tab Set to TS1 (Home, Tickets, Analysis...) and click Next.
15. Set Use Tab to T_TICKETS and click Next.
16. Change Submit Button Label to Save Changes and Add Row Button Label to Add tickets, as shown in Figure 7–5, and click Next.

Figure 7-5. Specifying button labels for a tabular form

17. Set the first branch to 230 and the second one to 200, as shown in Figure 7-6, and click Next.

Figure 7-6. Identifying branching for a tabular form

18. Click Finish.

Modifying a Tabular Form

While your tabular form works, you need to make some cosmetic modifications to it so that you can better control what data is entered into it. First, you'll move the Add Tickets button to the top of the page.

1. Edit page 230 of your application.
2. Edit the Add button by double-clicking on its name.
3. Set the button position to Region Template Position #CHANGE# and click Apply Changes.

Next, you'll make some changes to the columns of the report.

4. Edit the Report Attributes of the Manage Multiple Tickets report by right-clicking on the name of the report and selecting Edit Report Attributes.
5. Uncheck the Show checkbox for TICKET_ID_DISPLAY and then make sure all the Sort checkboxes for all *displayed* columns except for [row selector] are checked, as shown in Figure 7-7.

Alias	Link	Edit	Heading	Column Width	Column Alignment	Heading Alignment	Show	Sum	Sort	Sort Sequence
[row selector]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	 		left	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
TICKET_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Ticket Id		right		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
TICKET_ID_DISPLAY			Ticket Id		left		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
SUBJECT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Subject		left		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
DESCR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Description		left		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
ASSIGNED_TO	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Assigned To		left		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
CREATED_ON	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Created On		left		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
CLOSED_ON	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Closed On		left		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
CREATED_BY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Created By		left		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
STATUS_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Status Id		right		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

Figure 7-7. Tabular form of column attributes

6. Edit the SUBJECT column.
7. Under the Column Attributes region, set Display As to Text Area, Element Width to 20, Number of Rows to 3, and click Apply Changes.
8. Edit the DESCR column.
9. Under the Column Attributes region, set Display As to Text Area, Element Width to 20, Number of Rows to 3, and click Apply Changes.
10. Edit the ASSIGNED_TO column.
11. Under the Column Attributes region, set Display As field to Select List (Named LOV).
12. Under the List of Values section, set the Named LOV to TECHS, Display Extra Values to No, Display Null to Yes, and enter – Select a Tech – for the Null display value, as shown in Figure 7-8. Then click Apply Changes.

List of Values

Named LOV: TECHS

Display Extra Values: No

Display Null: Yes

Null display value: - Select a Tech -

Null return value:

List of values definition:

Figure 7-8. Specifying LOV for the ASSIGNED_TO column

13. Edit the CREATED_BY column.
14. Under the Column Attributes region, set the Display As field to Select List (Named LOV).

15. Under the List of Values section, set the Named LOV to USERS, Display Extra Values to No, enter - Select a User - for the Null display value, as shown in Figure 7–9, and click Apply Changes.

The image shows a 'List of Values' dialog box. At the top, 'Named LOV' is set to 'USERS'. Below it, 'Display Extra Values' is set to 'No' and 'Display Null' is set to 'No'. The 'Null display value' field contains '- Select a User -' and the 'Null return value' field is empty. At the bottom, there is a large empty box labeled 'List of values definition'.

Figure 7–9. Specifying LOV attributes for the USERS column

16. Edit the STATUS_ID column.
17. Under the Column Definition region, set the column heading to Status.
18. Under the Tabular Form Attributes region, set the default type to PL/SQL Expression or Function, enter get_status ('OPEN') for the default, and click Apply Changes.

Next, you'll need to create a button on page 200 that links to your new tabular form.

1. Edit page 200 of your application.
2. Create a new button by right-clicking the Region Buttons node and selecting Create.
3. Enter MANAGE_MULTIPLE_TICKETS for the button name, Manage Multiple Tickets for the label, select Template Based Button for the button style, Button for the button template, as shown in Figure 7–10, and then click Next.

The image shows a 'Create Button' dialog box. At the top, it says 'Page: 200 - Tickets' and 'Region: Tickets'. Below that, 'Button Name' is set to 'MANAGE_MULTIPLE_TICKETS' and 'Label' is set to 'Manage Multiple Tickets'. 'Button Style' is set to 'Template Based Button' and 'Button Template' is set to 'Button'. There is a 'Button Attributes' field at the bottom which is empty. At the top right, there are buttons for 'Cancel', '< Previous', and 'Next >'. Below the 'Button Name' field, there is a row of small buttons: '[Cancel]', '[Next]', '[Previous]', '[Apply]', '[Submit]', '[Delete]', '[Finish]', '[Create]', and '[Reset]'.

Figure 7–10. Specifying button attributes

4. Set the position to Region Template Position #CREATE# and click Next.
5. Set the action to Redirect to Page in this Application, the page to 230, check reset pagination for this page, as shown in Figure 7–11, and click Create Button.

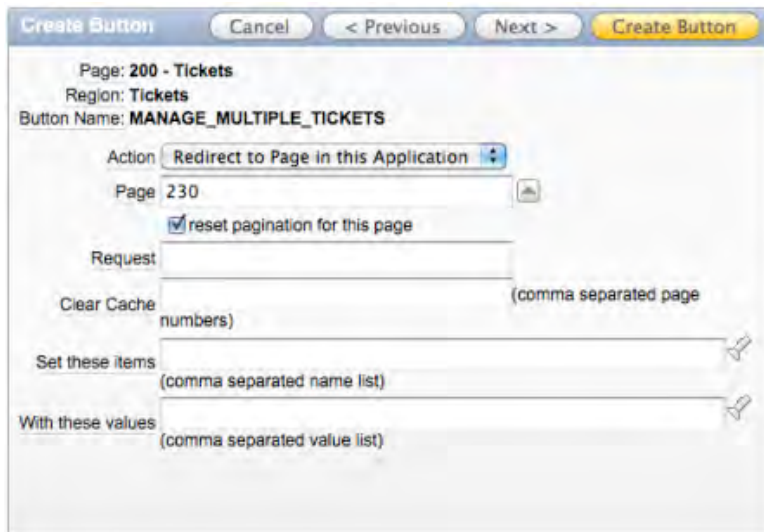


Figure 7-11. Specifying button action attributes

At this point, you should be able to navigate to your tabular form from page 200 by clicking the Manage Multiple Tickets button.

Looking Behind the Scenes

Let's take a look at what the Create Tabular Form wizard has created for you—the contents of your tabular form. Edit page 230 to view the Application Builder page for your tabular form. The Application Builder page will look similar to the one shown in Figure 7-12.

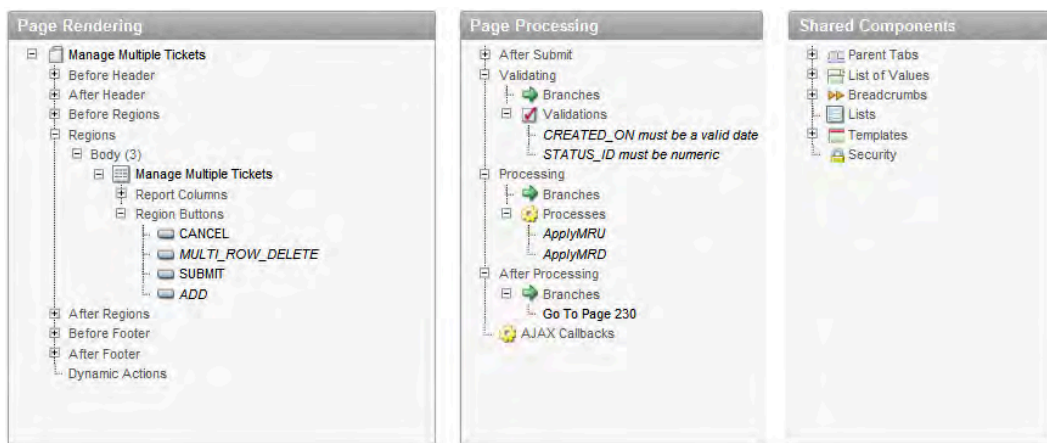


Figure 7-12. The Application Builder page for your tabular form

In the Page Rendering region, you see that APEX has created a Report region. But you created a form, didn't you? Despite its name, a tabular form is actually a SQL report with certain column-level options enabled and some processes added to handle data manipulation.

In the Page Processing region, in the Processing section, you see two processes, ApplyMRU and ApplyMRD. These special types of processes handle the multiple-row inserts and updates (ApplyMRU) and deletes (ApplyMRD) on your TICKETS table. These processes handle all DML operations on your TICKETS table for you.

APEX has also created two validations on the CREATED_ON and STATUS_ID columns, which are created automatically based on your TICKETS table column types, plus any User Interface Defaults defined on your TICKETS table.

In the Shared Components region, you see the usual page and tab templates that are the defaults for your application.

As you can see, the Multi-Row Update and Multi-Row Delete processes make the difference between your report region being a static report region and a fully-functional tabular form. And it's so much easier to let the APEX wizard create all this for you!

Interactive Reports

Your Ticket report is what's called a classic report. It's the original style of APEX report and still has practical application in a variety of situations when the requirement is for a simple list of data with no interactivity. Most applications, APEX included, now employ the APEX interactive report.

Introduced in APEX 3.1, interactive reports allow APEX to quickly and easily include user-driven ad-hoc capabilities in your applications. Interactive reports are greatly enhanced in APEX 4.0. The beauty of APEX interactive reports is that they give the end user powerful ad-hoc query capability with exactly zero lines of code by the developer. End users can customize the following:

- Searching
- Sort order
- Columns
- Breaking
- Highlighting
- Computations
- Aggregations
- Charts
- Group by
- Flashback time
- Saved reports
- Subscription (E-mail notification)

Interactive reports are technically nothing more than a report type. The Create Report wizard steps are similar, and you expend the same effort in building an interactive report as for a classic report.

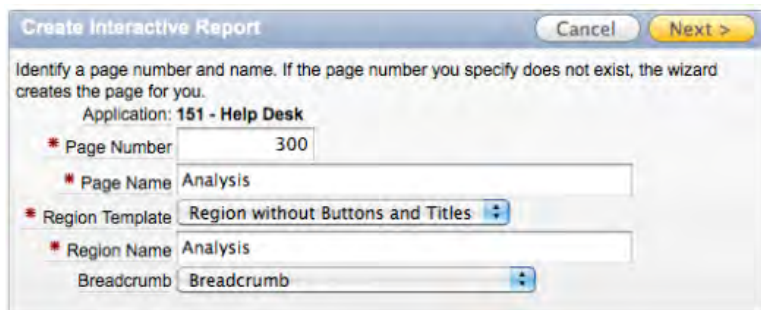
Classic reports can be easily converted to interactive reports. There is no way to revert from an interactive report to a classic report. (But why would you want to?) The end-user features and overall

value of interactive reports are best illustrated with an example, so let's add an interactive report to your application.

Creating an Interactive Report

Interactive reports require nothing more than a SQL query. APEX handles the rest. You'll start by creating a new page, tab, and interactive report all at once on a view of your Ticket System data.

1. Run any page in your application.
2. Click Create in the Developer's Toolbar.
3. Select New Page and click Next.
4. Select Report and click Next.
5. Select Interactive Report and click Next.
6. Enter 300 for the page number, Analysis for the page name, and set the region template to Reports Region, as shown in Figure 7–13.
7. Enter Analysis for the region name, set Breadcrumb to Breadcrumb, as shown in Figure 7–14, and click Next.



The screenshot shows the 'Create Interactive Report' dialog box. At the top, it says 'Identify a page number and name. If the page number you specify does not exist, the wizard creates the page for you.' Below this, the 'Application' is '151 - Help Desk'. There are four main fields: 'Page Number' (300), 'Page Name' (Analysis), 'Region Template' (Region without Buttons and Titles), and 'Region Name' (Analysis). There is also a 'Breadcrumb' field set to 'Breadcrumb'. The 'Next >' button is highlighted in yellow.

Figure 7–13. Specifying page number and name for an interactive report

Create Breadcrumb Entry

Entry Name:

Parent Entry:
[No parent breadcrumb entry]

Select Parent Entry:

Name	Page
Home	1
Contact Us	3
Create a Ticket	2
Tickets	200
Manage Tickets	210
Ticket Details	220

row(s) 1 - 6 of 6

Figure 7–14. Specifying a breadcrumb entry for an interactive report

8. Set the Tab Options to Use an existing tab set and create a new tab within the existing tab set. When the page refreshes, set the Tab Set to TS1 (Home, Tickets), enter Analysis for the New Tab Label, and click Next (see Figure 7–15).

Create Interactive Report Cancel < Previous Next >

Specify whether to use tabs for this page. You may either use an existing tab or create a new one for this page.

Page: 300

Tab Options:

- ☐ Do not use tabs
- ☒ Use an existing tab set and create a new tab within the existing tab set.
- ☐ Use an existing tab set and reuse an existing tab within that tab set.

Tab Set: TS1 (Home, Tickets) ▾

New Tab Label:

Figure 7–15. Specifying tab options for an interactive report

9. Enter `SELECT * FROM tickets_v` for the Enter a SQL SELECT Statement region, set Uniquely Identify Rows by to Unique Column, enter TICKET_ID for Unique Column, and click Next (see Figure 7–16).

Create Interactive Report

Cancel < Previous Next >

* Enter a SQL SELECT statement

SELECT * FROM tickets_v

Query Builder

Link to Single Row View Yes

Uniquely Identify Rows by Unique Column

Unique Column TICKET_ID

Figure 7–16. Entering SQL SELECT statement for an interactive report

10. Click Finish.

Running an Interactive Report

Run your application and navigate to the Analysis tab. The page will look similar to Figure 7–17. At first glance, the interactive report looks no different than any other APEX report. However, there are a number of functions that the interactive report can perform that a standard APEX report cannot.

<div>Home Tickets Calendar Analysis</div> <div>Analysis</div> <div> <input type="text"/> <input type="button" value="Go"/> <input type="button" value="Actions"/> </div>									
Ticket Id	Subject	Description	Assigned To	Created On	Created By	Closed On	Status	Number Of Details	
1	Cannot log into E-Mail	User called and cannot log into his MS Outlook e-mail Account.	SCOTT	02-OCT-10	PAUL	02-OCT-10	OPEN	2	
2	PC will not turn on	The user's PC will not turn on when the power button is pressed.	JOHN	01-OCT-10	RINGO	-	CLOSED	1	
3	Need more memory	User needs more memory installed	DOUG	30-SEP-10	GEORGE	-	OPEN	1	
4	MSIE Crashed 4 times	MSIE keeps on crashing for any site	SCOTT	29-SEP-10	JOHN	-	CLOSED	1	
5	Need to install SP2	SP2 Upgrade needed in order to be compliant	DIMITRI	28-SEP-10	ALEX	-	OPEN	1	
6	Network drive not being mapped	X: drive not being mapped to \\corp\share	DIMITRI	27-SEP-10	GEDDY	-	OPEN	1	
7	BSOD after rebooting	Blue Screen of Death every time system is rebooted	DOUG	26-SEP-10	NEAL	-	OPEN	1	
8	Wireless signal not strong enough	Wi-Fi signal not as strong as it was last week	SCOTT	25-SEP-10	JOHN	01-OCT-10	CLOSED	2	
9	I think I have a virus	Something is not right - PC is slow	JOHN	24-SEP-10	ROBERT	-	OPEN	1	
10	Virus Definitions Dates	Message stating that virus updates are needed keeps appearing	SCOTT	23-SEP-10	JOHN	-	CLOSED	1	
11	Funny smell coming from PC	There is an odd odor emanating from my PC...	DIMITRI	22-SEP-10	JIMMY	-	OPEN	1	
12	Accidentally deleted Q2.ppt	File Q2.ppt placed in Recycle Bin; bin emptied	JOHN	21-SEP-10	EDDIE	-	OPEN	1	
13	Several dead pixels on screen	There are at least 4 dead pixels on the display	DOUG	20-SEP-10	ALEX	-	PENDING	1	
14	Smartphone will not sync with Outlook	Motorola Q does not sync with Outlook contacts and calendar events	SCOTT	19-SEP-10	MICHAEL	-	OPEN	1	
15	Getting Out of Memory errors	Same Out of Memory error occurs when Office starts	JOHN	18-SEP-10	DAVID	-	PENDING	1	

1 - 15 ▶

Figure 7–17. Interactive report for Tickets Analysis

The interactive report has a built-in Search Bar, which is command central for the interactive report. All of the end user features are accessed through the Search Bar. The Search Bar is located on the top of the interactive report, in the standard location for a report search field. But this is so much more than just a search field! The Search Bar includes:

- **Finder drop down:** Represented by the magnifying glass, this feature allows the user to select which column to filter on.
- **Search field:** A search field where the user can enter and find text strings.
- **Report Select List:** A select list of all saved reports. This select list will only be visible when more than one saved report is available. We'll talk about saved reports in a moment.
- **Rows Per Page selector:** A select list of number of rows options. This function is turned off by default, as it is also available from within the actions menu.

- **Actions menu:** A menu of actions enabled for this report—the “interactive” options of the interactive report.

To use the Search field, type a string or phrase into it and click the Go button. The interactive report will list only results that match value you entered in the Search field.

To use the Finder drop down, click on the arrow at the bottom-right of the magnifying glass icon on the left of the Search bar. This action opens a menu of the report column names. Selecting a column name causes the Search to be on the selected column only.

To use the Report Select List, select one of the Report List options to navigate to the selected report. To use the Rows Per Page selector, select the desired number of rows per page to display from the Rows Per Page select list.

To use the Actions menu, click the Actions menu to expand the menu of interactive reports actions, then select the desired action.

Restricting Functionality by Report

As the developer, you have control over what Actions menu options are available to the end user. You can also control which of the preceding components are included on the Search Bar. The Include in Search Bar options, shown in Figure 7–18, allow the developer to include the Search Bar or not, and to elect which elements of the Search Bar are visible to the user. This controls end user functionality at the report level.

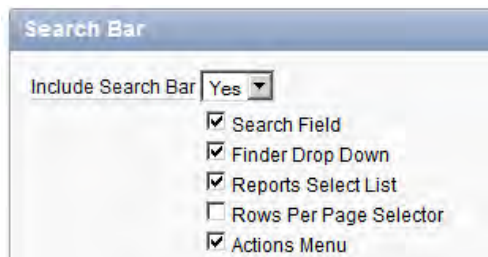


Figure 7–18. Specifying Search Bar options

The Include in Actions Menu checkboxes, illustrated in Figure 7–19, allow you to specify which Actions menu options are available to the user. Of these, the Save Public Report and Subscriptions options are only available to authenticated pages. This is because APEX needs to know information about the authenticated user to be able to save reports and send subscriptions.

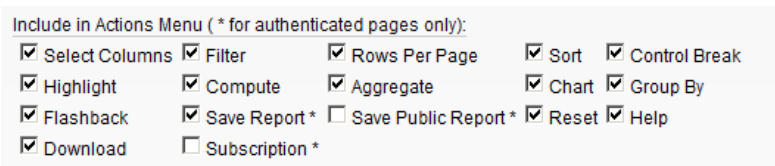


Figure 7–19. Specifying the options for the Actions menu

The remaining attributes, shown in Figure 7–20, allow you to specify authorization for saving public reports, a button template, an image for the Finder icon, an alternate image of the Actions menu, the Search button label, and the maximum rows per page.

Save Public Report Authorization: - No Authorization Required -

Button Template: Button

Finder Image: [Empty field]

Actions Menu Image: [Empty field]

Search Button Label: Go for it!

Maximum Rows Per Page: [Empty field]

Figure 7–20. Specifying Search Bar button, image, and maximum rows attributes

Restricting Functionality by Column

Specific interactive report actions can also be restricted on a column-by-column basis. For example, you can allow the report to be filtered, but not allow a specific column to be filtered. You can declaratively enable or disable the sort, filter, highlight, control break, aggregate, computer, chart and group by at the column level through the individual column report attributes page, the Search Bar region, as shown in Figure 7–21.

Column Definition

Column Name: SUBJECT

Column Type: STRING

Group: - Select Column Group -

Display Type: Display as Text (escape special characters)

Column Heading: Subject

Single Row View Label: [Empty field]

Number / Date Format: [Empty field]

Heading Alignment: center

Column Alignment: left

Allow Users To:

- ☒ Hide
- ☒ Sort
- ☒ Filter
- ☒ Highlight
- ☒ Control Break
- ☒ Aggregate
- ☒ Compute
- ☒ Chart
- ☒ Group By

Figure 7–21. Specifying individual column options

You have examined the interactive report settings you can set as a developer at the report level and at the column level. Now let's take a look at the interactive report features from the end user perspective. In the following sections, you'll examine using the key features of your interactive report as an end user.

Using the Column Heading Menu

The column headings of an interactive report contain functionality all their own and are perhaps the fastest way to format a single column of your report. Figure 7–22 illustrates the interactive report column

heading features. Double-clicking on a column heading opens a column-level menu with icon-driven options for quick sorting, removing the column from the report, adding a break on the column, searching, and filtering on the selected column. The Search Bar in this menu allows the end user to search for and filter directly on the values in that column. The Delete Column option allows the user to quickly remove the column from the report. To restore the column, use the Select Columns option. The Control Break option will add a break on the column.

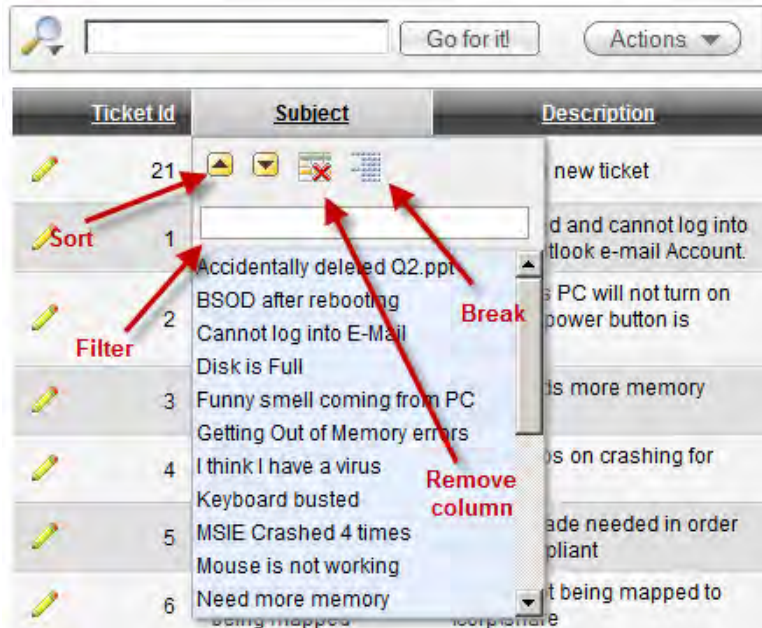


Figure 7-22. Using the Column Heading menu

Searching by Column

The magnifying glass icon on the left end of the Search Bar is actually a list of the visible columns in the report, which is helpful for a quick way of filtering on a specific column or on all columns. The selected column is the column on which the search text applies.

Entering a value in the search field applies a filter to either all columns (the default) or the selected column. Once a filter is applied, an option appears in the Control Summary region, as illustrated in Figure 7-23. The Control Summary region is the area between the Search Bar and your report. This region only appears when there is an action applied to the interactive report and serves as a key to what actions are currently applied. The Control Summary region contains one line for each action applied. Interactive report actions are additive: subsequent actions are applied in addition to the existing actions. The user can disable an action by unchecking its checkbox. The user can remove the action by clicking on the Delete icon for that action. Double-clicking on an action in the Control Summary opens that action control for editing.

The Control Summary panel can be toggled open or closed. It can be minimized by clicking on the close (minus sign) icon.

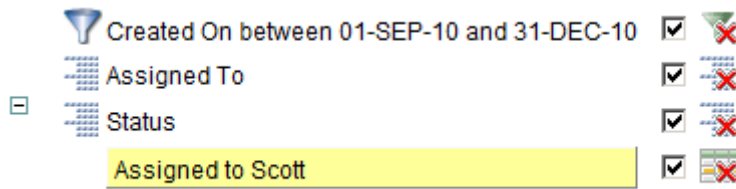


Figure 7-23. Control summary region when open

The closed Control Summary region, illustrated in Figure 7-24, can be exposed by clicking on the open (plus sign) icon.



Figure 7-24. Control summary region when closed

The Finder drop-down menu, accessible from the magnifying glass icon to the left of the Search field, displays a list of all columns in the interactive report, as shown in Figure 7-25. Selecting one of the columns limits the Search function to that column.

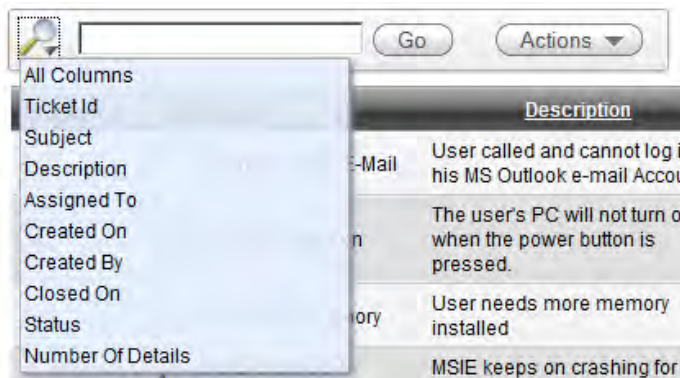


Figure 7-25. Finder drop-down menu

The Actions menu, illustrated in Figure 7-26, exposes an array of column selection, filtering, and action options. Expanding the menu further under the Format Action option uncovers additional formatting actions for sorting, break, highlighting, computing new columns, aggregating, charting, and grouping. The expanded Format Actions menu is illustrated in Figure 7-27.

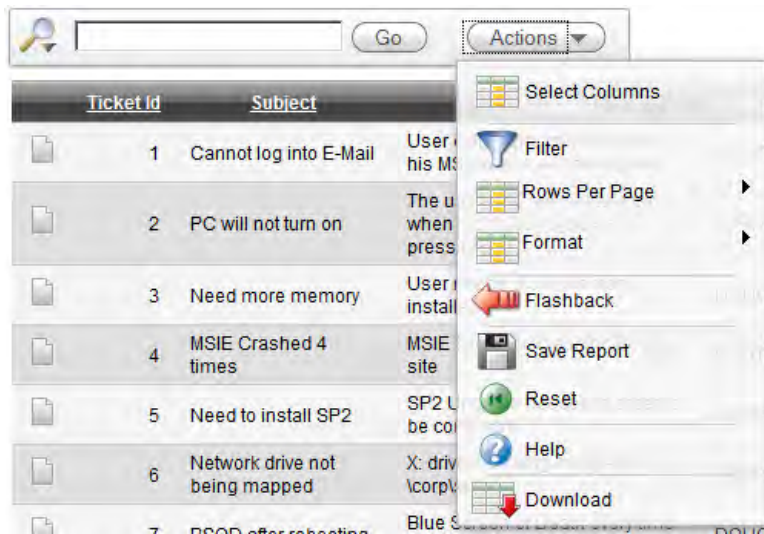


Figure 7-26. Actions menu

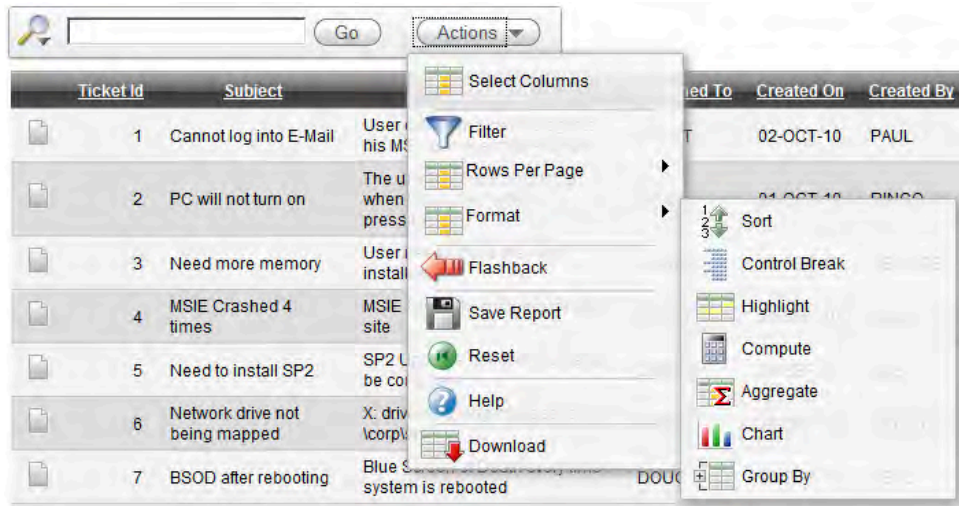


Figure 7-27. Choosing a format option from the Actions menu

Selecting Columns

The Select Column action, shown in Figure 7–28, lets the user select which columns to display and allows the user to re-order columns as desired. The shuttle control allows the user to easily add or remove columns using the center arrows and to order the columns that are displayed by using the up and down areas on the right of the region.

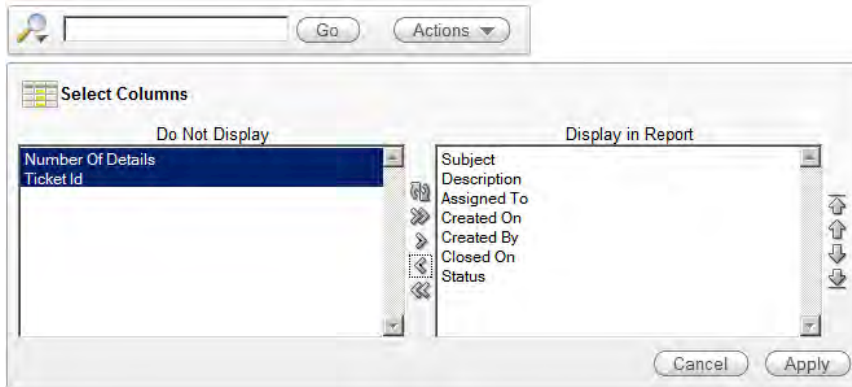


Figure 7–28. Selecting columns

■ **Note** The Select Column action of an interactive report always controls what columns are displayed. If, as a developer, you modify the SQL query to add a column to an interactive report, that new column will not be visible until the new column is moved from the Do Not Display region to the Display in Report region of the shuttle.

Filtering

The Filter action allows the user to declaratively define filters based on the result of a number of operators. A user can define multiple filters per report. Multiple filters will be combined with the logical AND operator. Filters defined through the Search bar are combined with filters defined in the Filter action.

The Filter action offers a full set of filter operations for selection, as shown in Figure 7–29.

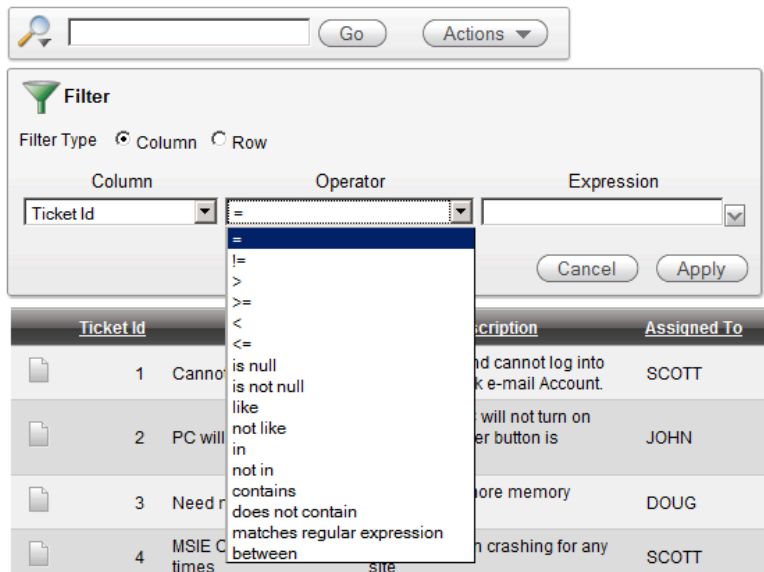


Figure 7–29. Applying a filter to an interactive report

The Filter action supports both column filters and row filters. Column filters are applied to a single column. The column filter options change interactively, depending on the type of the filter column and the selected operator. For example, if one selects a date column, like Created On, and selected the between operation, the Expression element now contains two fields, for the From and To of the between clause. In this case, the fields each have a date picker for ease in entering the Date From and To values. The end user can also construct a custom filter using the declarative Filter

Row filters allow the user to build filter conditions that are based on multiple columns in the same row. A simple row filter for your Analysis report is a filter for all tickets that were closed on the same day they were opened. The Filter expression may be built declaratively using selections in the Columns and Function/Operators regions, all illustrated in Figure 7–30, or entered manually. Within the Filter Expression, selected columns are represented by their letter alias.

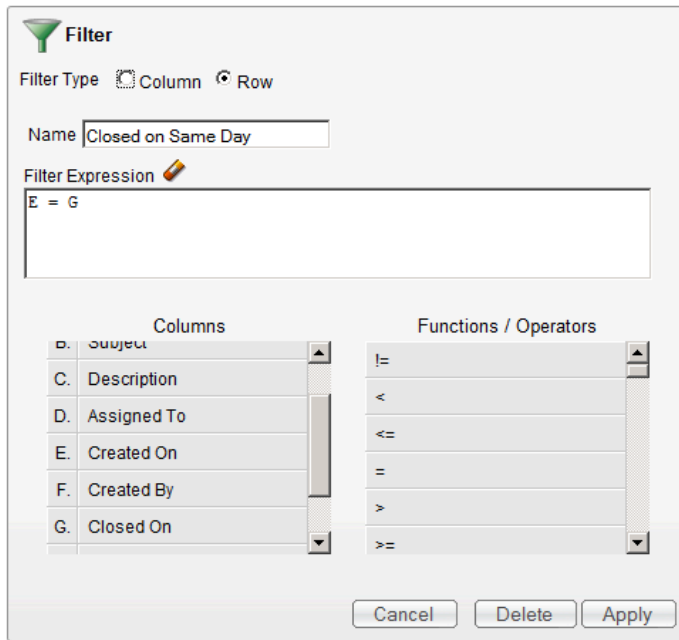


Figure 7-30. Building a row filter

Sorting

The Sort interface allows the user to specify sorts on up to six columns in ascending or descending order, and whether NULLs are sorted first or last. The sort may be on displayed and non-displayed columns (see Figure 7-31).

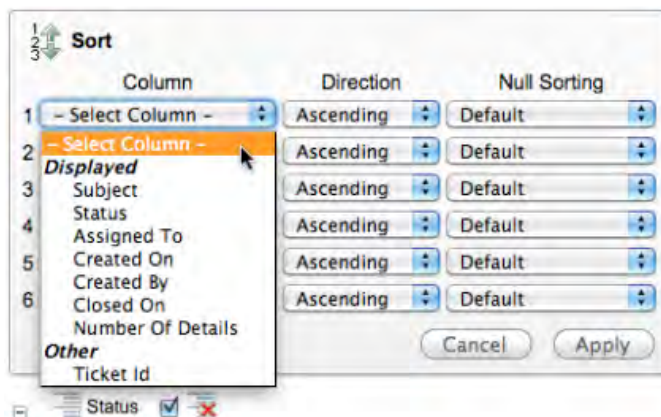


Figure 7-31. Adding sorts to an interactive report

Adding Breaks

The Control Break action allows the user to define break formatting on up to six columns. The user specifies the break column and whether the break is disabled or enabled. APEX automatically applies the declared break formats to the report. Note that break columns appear in the Control Summary as separate entries, allowing the user to enable, disable, or remove break columns individually. Figure 7–32 shows your Tickets analysis report with breaks applied on the Assigned To and Status columns.

Assigned To
Status

Assigned To : DOUG, Status : OPEN

Ticket Id	Subject	Description	Created On	Created By	Closed On	Number Of Details
3	Need more memory	User needs more memory installed.	04-FEB-2011	GEORGE	-	1
7	BSOD after rebooting	Blue Screen of Death every time system is rebooted	31-JAN-2011	NEAL	-	1
16	VPN Client Install Issues	Cannot install VPN client - installer errors out each time	22-JAN-2011	JACKIE	-	1
20	Disk is Full	No more space error keep coming up	18-JAN-2011	MARLON	-	1

Assigned To : DOUG, Status : PENDING

Ticket Id	Subject	Description	Created On	Created By	Closed On	Number Of Details
13	Several dead pixels on screen	There are at least 4 dead pixels on the display	25-JAN-2011	ALEX	-	1

Assigned To : KAREN, Status : OPEN

Ticket Id	Subject	Description	Created On	Created By	Closed On	Number Of Details
5	Need to install SP2	SP2 Upgrade needed in order to be compliant	02-FEB-2011	ALEX	-	1
6	Network drive not being mapped	X: drive not being mapped to \\corp\share	01-FEB-2011	GEDDY	-	1
11	Funny smell coming from PC	There is an odd odor emanating from my PC,...	27-JAN-2011	JIMMY	-	1
17	Mouse is not working	Mouse does not move the pointer anymore	21-JAN-2011	TITO	-	1

Assigned To : MARTIN, Status : CLOSED

Ticket Id	Subject	Description	Created On	Created By	Closed On	Number Of Details
2	PC will not turn on	The user's PC will not turn on when the power button is pressed.	05-FEB-2011	RINGO	-	1

Assigned To : MARTIN, Status : OPEN

Ticket Id	Subject	Description	Created On	Created By	Closed On	Number Of Details
9	I think I have a virus	Something is not right - PC is slow	29-JAN-2011	ROBERT	-	1
12	Accidentally deleted Q2.ppt	File Q2.ppt placed in Recycle Bin; bin emptied	26-JAN-2011	EDDIE	-	1

Assigned To : MARTIN, Status : PENDING

Ticket Id	Subject	Description	Created On	Created By	Closed On	Number Of Details
15	Getting Out of Memory errors	Same Out of Memory error occurs when Office starts	23-JAN-2011	DAVID	-	1
19	Keyboard busted	None of the keys work (I had to use someone else's PC to enter this)	19-JAN-2011	MICHAEL	-	1

Figure 7–32. Interactive report with control breaks applied

Highlighting

The Highlight action allows the user to find matching data and highlight it by row or column, specifying the background and text colors for the highlight. The Highlight Action interface is shown in Figure 7–33.

Highlight

Name: Assigned to Scott

Sequence: 10

Enabled: Yes

Highlight Type: Row

Background Color: #FFFF99 [yellow] [green] [blue] [orange] [red]

Text Color: #000000 [yellow] [green] [blue] [orange] [red]

Highlight Condition

Column	Operator	Expression
Assigned To	=	SCOTT

Buttons: Cancel, Delete, Apply

Figure 7–33. Adding highlighting with the Highlight action

The same operators that you saw in the Filters action apply here. The background and text colors may be specified by HEX notation or by using the color palettes. The Highlight action appears in the Control Summary region as a highlighted row.

Computing Columns

The Compute action allows the user to define a new column as a computation based on existing columns and functions using the Compute action interface, shown in Figure 7–34.

Compute

Computation: - New Computation -

Column Heading: Days to Close Format Mask: 999G999G999G999G999G

Computation: G - E

Columns	Keypad	Function
C. Description	() *	ABS
D. Assigned To	7 8 9 -	ADD_MONTHS
E. Created On	4 5 6 +	CASE
F. Created By	1 2 3 *	CEIL
G. Closed On	0 . /	CHR
H. Status	space ,	COALESCE
I. Number Of Details		COS

Create a computation using column aliases.
Examples:

1. (B+C)*100
2. INITCAP(B)||', '||INITCAP(C)
3. CASE WHEN A = 10 THEN B + C ELSE B END

Cancel Apply

Figure 7–34. Computing a new interactive report column using the Compute action

The user may declaratively or manually define the computed value. The declarative interface is much the same as the Row Filter interface. Columns are specified in the computation as their letter aliases. This option is quite powerful, as it allows the end user to build essentially any column they desire.

Adding Aggregates

The Aggregate action performs one of the following aggregation functions on a column:

- Sum
- Average
- Count
- Count Distinct
- Minimum
- Maximum
- Median

The selected column must be of data type NUMBER. The results are displayed at the end of the report. Note that aggregate results will only be displayed if the corresponding column is also displayed.

Adding Charts

The Chart action allows the user to display a dynamic Flash chart representation of the data in the report, as shown in Figure 7–35. The chart representation of the data is displayed instead of the tabular data representation. The display can be toggled by clicking on the chart display icon, as indicated in Figure 7–35. Use the Edit Chart link to re-enter the Chart action interface.

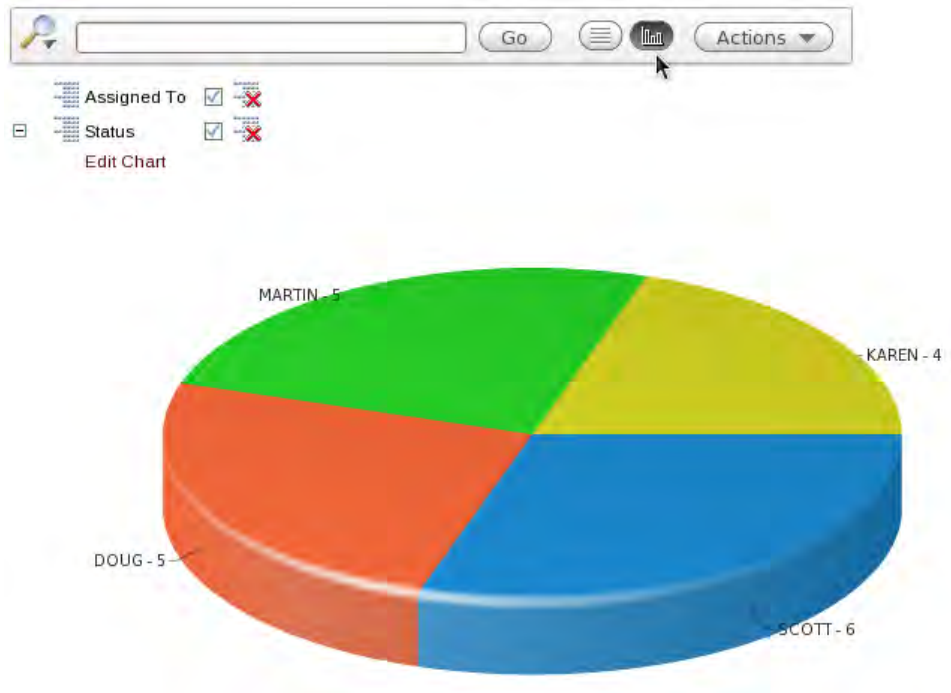


Figure 7–35. Interactive report pie chart

The following chart types are supported in an interactive report:

- Horizontal bar
- Vertical bar
- Pie
- Line

The simple chart action interface, shown in Figure 7–36, allows the user to select the chart type, assign a label column, a value column, a function, and a column to sort by.

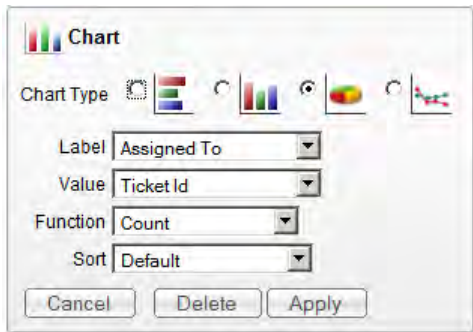


Figure 7-36. Adding a chart using the Chart action

The user does not have the full functionality of APEX charts within the Chart action, but the ease of displaying these most common chart types is quite valuable.

Grouping

The Group By action allows the user to define groups and aggregate functions on those groups, allowing the user to declaratively define their own summary views of the report data. The options of the Group By action are shown in Figure 7-37.

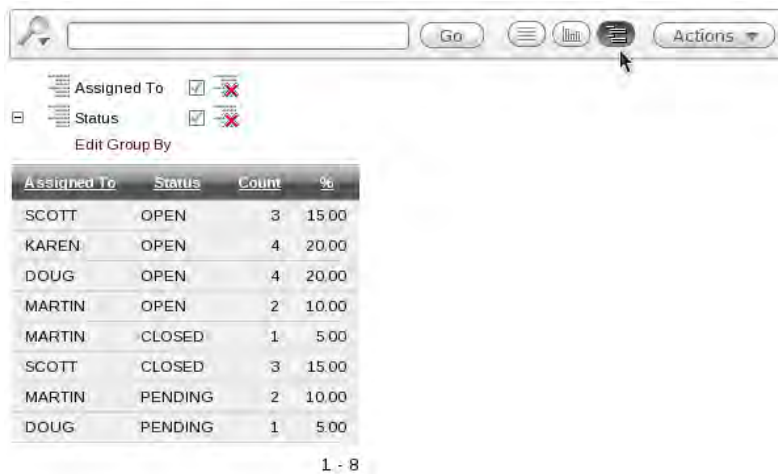


Figure 7-37. Grouping using the Group By action

Like the Chart view, the Group By view of the data has a display icon in the center of the Search Bar, as indicated in Figure 7-37. The user may display the data view, the Group By view, or, if defined, the Chart view of the data by clicking on the appropriate display icon.

Using Flashback

The Flashback action enables the user to flash back the database by the specified number of minutes to see what the data looked like at that point in time. The option is built on the Oracle database FLASHBACK feature. Database FLASHBACK must be enabled. The Flashback action asks for the number of minutes to flash back, as shown in Figure 7–38.

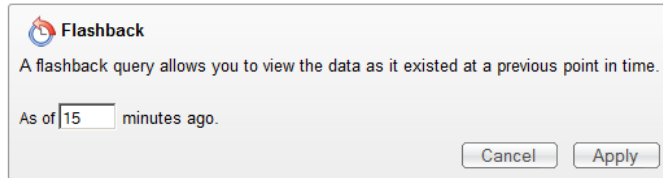


Figure 7–38. Using the Flashback action

The length of flashback time is configurable. The maximum flashback period is based on the UNDO_RETENTION parameter in the database, which is set to three hours by default.

Saving an Interactive Report

The Save Report action allows the user to save the current configuration of the interactive report as a named report. If the end user is also an APEX developer, the user will have the Save As Default Report Settings option as shown in Figure 7–39.

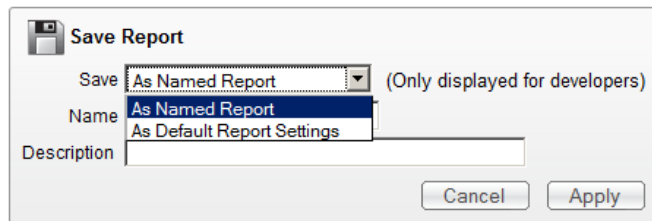


Figure 7–39. Saving an interactive report using the Save Report action

Developers may save the current report configuration as primary or alternative default report settings, as shown in Figure 7–40. User may save reports as private reports. When a report is saved, it gets added to the Reports menu in the Search Bar, as shown in Figure 7–41.

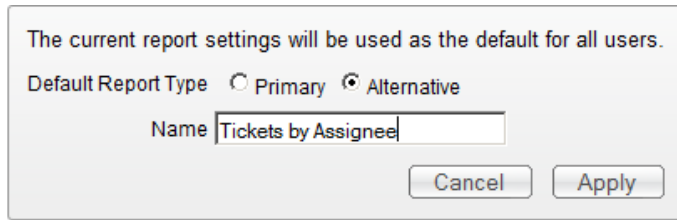


Figure 7-40. Setting an alternate saved report

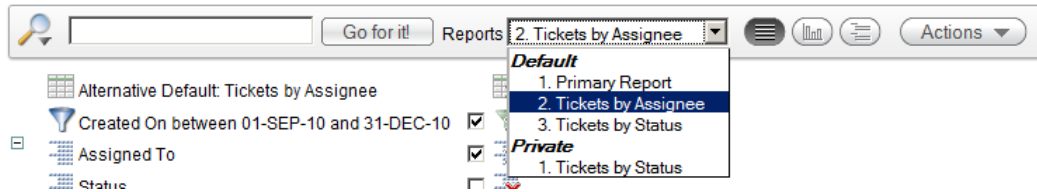


Figure 7-41. Using the default Report menu

Resetting an Interactive Report

The Reset action, shown in Figure 7-42, restores the current report to the default settings. Any changes in formation or result set (by filtering) will be lost, unless of course the report is a saved report. It may then be reinstated simply by selecting the report name from the select list.

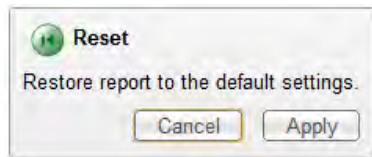


Figure 7-42. Resetting an interactive report to its default settings

Getting Help

The Help action opens a window that contains interactive report-specific help, as shown in Figure 7-43. All of the interactive report options are displayed in this Help window, regardless of whether they are enabled for the current report or not.

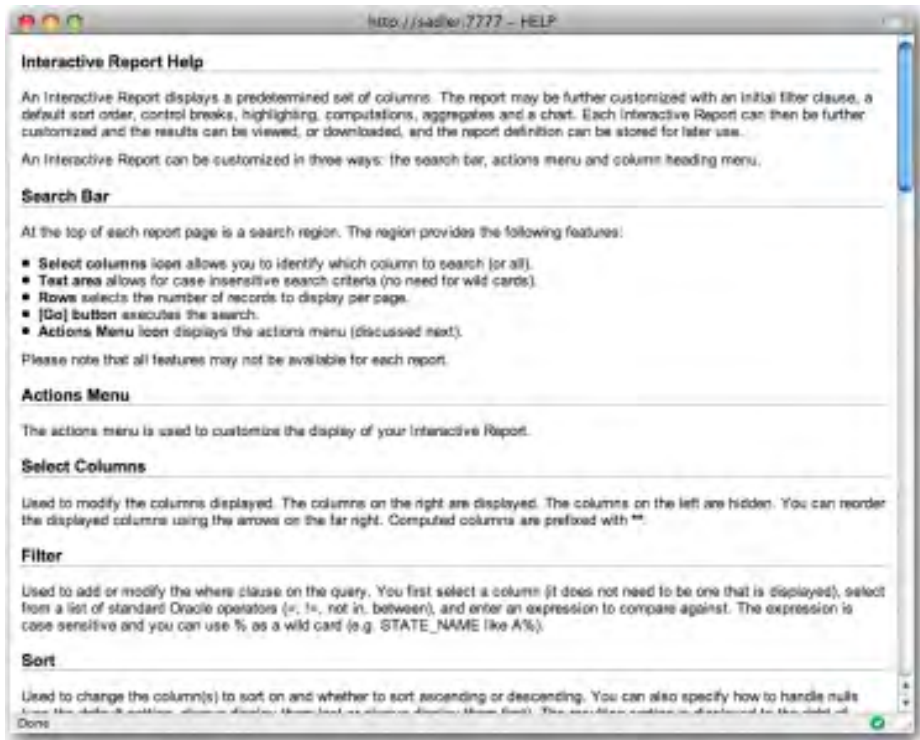


Figure 7-43. The interactive report help page

Adding a Subscription

The Subscription action allows the user to e-mail a report to designated e-mail addresses on a scheduled basis. The user enters the e-mail address, subject, frequency, and start and end date, as shown in Figure 7-44. This action is available for authenticated users only. The e-mail received is a searchable HTML version of your report. Break formatting and highlighting are not preserved.

Figure 7-44. Subscribing to an interactive report

Report subscriptions are managed by a Workspace Administrator through the Manage Service ► Manage Interactive Report Settings ► Subscriptions interface, shown in Figure 7–45.

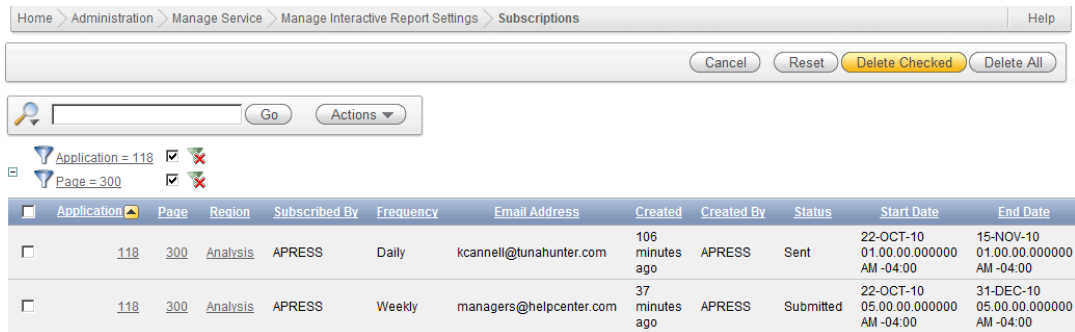


Figure 7–45. Managing subscriptions through the manage subscriptions interface

Downloading

The Download action allows the user to download the current result set of their report in one of the following formats:

- CSV
- HTML
- E-mail
- XLS (MS Excel)
- PDF
- RTF (MS Word)

The latter three formats require Oracle BI Publisher, which may require a separate license from Oracle. Figure 7–46 shows the download options without and with BI Publisher. You can specify which formats are available in the Download attributes region, as shown in Figure 7–47.

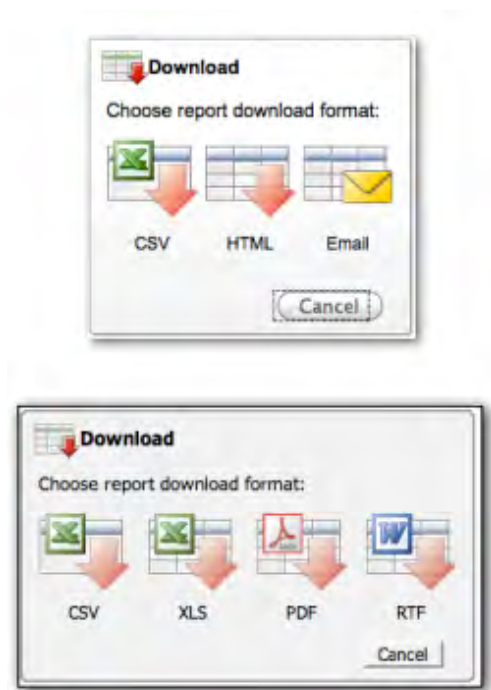


Figure 7-46. Choosing download options, without and with BI Publisher

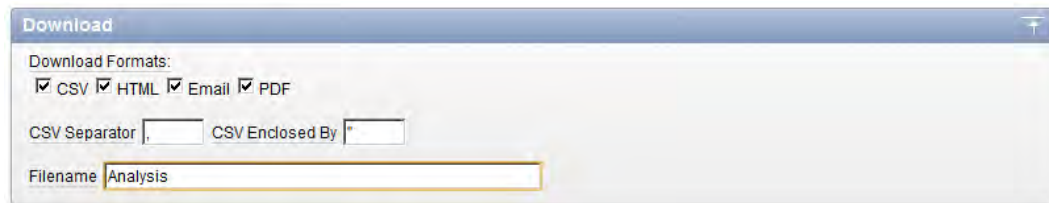


Figure 7-47. Specifying download attributes

Reports downloaded in CSV format are plain comma-delimited data. The content and order of data in the result set are retained in the CSV file but break formatting and highlighting are not.

Reports downloaded in HTML format are a searchable HTML version of the result set, as illustrated in Figure 7-48. Again, the result set content is preserved, but the break formatting and highlighting are not.

Search:

Assigned To	Status	Ticket Id	Subject	Description	Created On	Created By	Closed On	Number Of Details	Days to Close
SCOTT	OPEN	1	Cannot log into E-Mail	User called and cannot log into his MS Outlook e-mail Account.	06-FEB-2011	PAUL	06-FEB-2011	2	0
DOUG	OPEN	3	Need more memory	User needs more memory installed	04-FEB-2011	GEORGE		1	
KAREN	OPEN	5	Need to install SP2	SP2 Upgrade needed in order to be compliant	02-FEB-2011	ALEX		1	
KAREN	OPEN	6	Network drive not being mapped	X: drive not being mapped to \\corphshare	01-FEB-2011	GEDDY		1	
DOUG	OPEN	7	BSOD after rebooting	Blue Screen of Death every time system is rebooted	31-JAN-2011	NEAL		1	
MARTIN	OPEN	9	I think I have a virus	Something is not right - PC is slow	29-JAN-2011	ROBERT		1	
KAREN	OPEN	11	Funny smell coming from PC	There is an odd odor emanating from my PC...	27-JAN-2011	JIMMY		1	
MARTIN	OPEN	12	Accidentally deleted Q2.ppt	File Q2.ppt placed in Recycle Bin, bin emptied	26-JAN-2011	EDDIE		1	
SCOTT	OPEN	14	Smartphone will not sync with Outlook	Motorola Q does not sync with Outlook contacts and calendar events	24-JAN-2011	MICHAEL		1	
DOUG	OPEN	16	VPN Client Install Issues	Cannot install VPN client - installer errors out each time	22-JAN-2011	JACKIE		1	
KAREN	OPEN	17	Mouse is not working	Mouse does not move the pointer anymore	21-JAN-2011	TITO		1	
SCOTT	OPEN	18	Speakers are too soft	Cannot get good quality of sound from built-in speakers	20-JAN-2011	JERMAINE		1	
DOUG	OPEN	20	Disk is Full	No more space error keep coming up	18-JAN-2011	MARLON		1	

Figure 7–48. The searchable HMTL download of an interactive report

The e-mail download is the same output as the HTML download, delivered in an e-mail.

The XLS, PDF, and RTF download formats require integration with Oracle BI Publisher, which may require a separate license from Oracle. A complete description of the use of Oracle BI Publisher to produce reports in these formats is beyond the scope of this book. See the Oracle APEX documentation section “Advanced Printing Options and Configuration” for more details. If these options are not configured for your installation, they will not appear in the download options list.

Take some time to experiment with the features of the interactive report. If you get lost and need to start over, simply click the Actions button and select Reset. The interactive report will be reset to its original state and all modifications that you made to it will be discarded.

Modifying an Interactive Report

While the interactive report offers a tremendous amount of functionality, you may wish to limit which features are available to your end users. Each feature of the interactive report can be disabled on a report-by-report basis. In addition, you can set up default options for a specific report, making those available to all end users.

So let's take another look at your interactive report. You'll use a combination of interactive report end user actions and developer settings to achieve your modifications. First, you'll remove a column from the report and add a sort attribute using the Actions menu.

1. Run your application and navigate to the Analysis table.
2. Click on the Actions button to display the Actions menu.
3. Select the Select Columns menu, as shown in Figure 7–49.

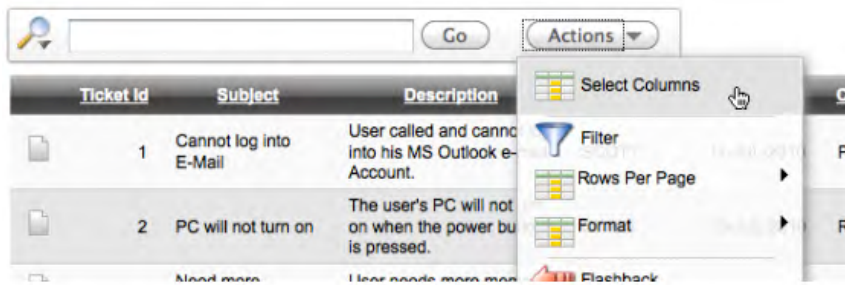


Figure 7–49. Selecting the Select Columns menu

4. Move Ticket Id to the Do Not Display section of the shuttle, as shown in Figure 7–50, by double-clicking on its name.
5. Using the up and down arrows, reorder the remaining columns so that Status appears after Subject and before Description, as shown in the display in Report section in Figure 7–50, and click Apply.

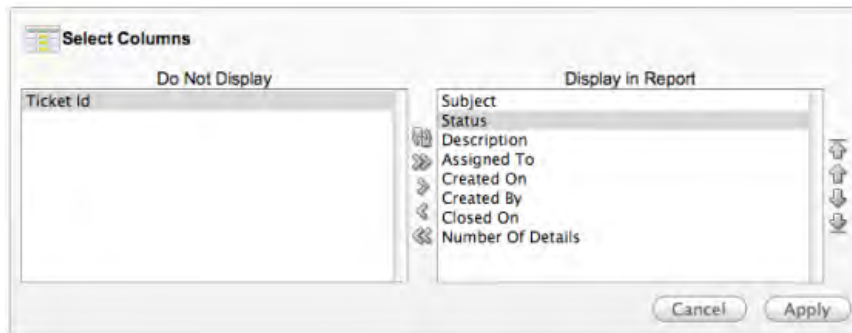


Figure 7–50. Selecting columns

Notice that the Ticket Id column is no longer displayed in your report and that the Status column appears immediately after the Subject column.

Next, you'll set your changes as default options for the interactive report. These options will be applied for all end users who use the interactive report. The Save As Default Report Settings option is only available to end users who are APEX developers.

6. Click on the Actions button and select the Save Report item.
7. Set Save to As Default Report Settings, as shown in Figure 7–51.

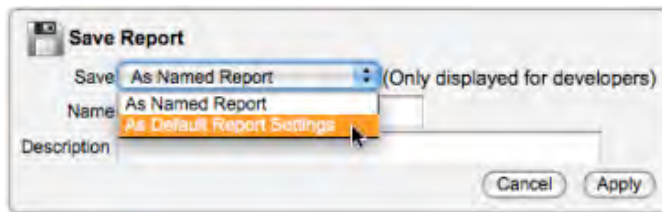


Figure 7-51. The Save As Default Report setting

8. The region will immediately change, allowing you to save the report as the primary default or as a named alternative. You'll make this one the primary default, as shown in Figure 7-52. Click Apply.

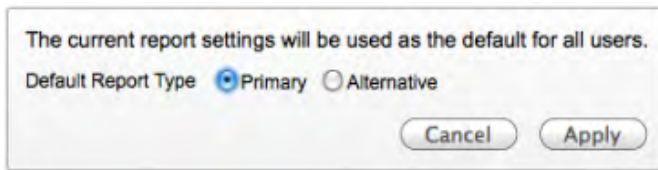


Figure 7-52. Saving a primary interactive report

Now you'll create a Named Alternative default report that does a control break on the status column.

9. Click on the Actions button and navigate to Format ► Control Break, as shown in Figure 7-53.

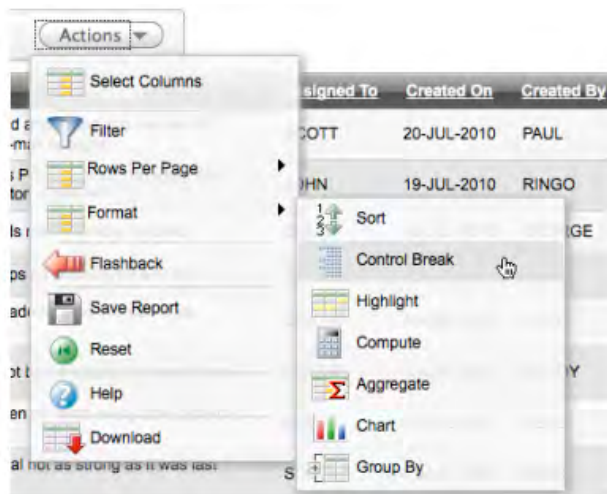


Figure 7-53. Selecting the Control Break action

10. Select Status in the first select list and make sure that the status is set to Enabled, as shown in Figure 7–54, and click Apply.

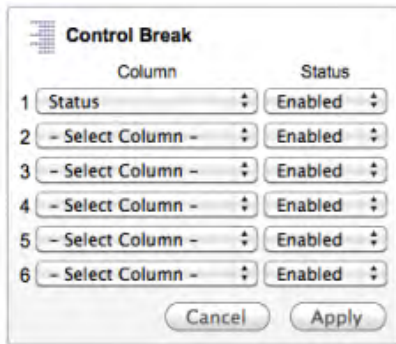


Figure 7–54. Applying a control break to an interactive report

11. Click on the Actions button and select the Save Report option.
12. Set Save to As Default Report Settings, as shown in Figure 7–55.

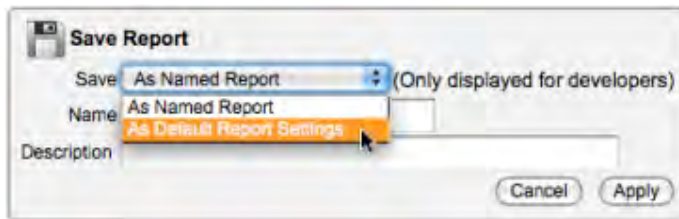


Figure 7–55. Saving an interactive report as a default setting

13. The region will immediately change. This time you'll save the report as a Named Alternative. Select Alternative, enter Tickets by Status for the name as shown in Figure 7–56, and click Apply.

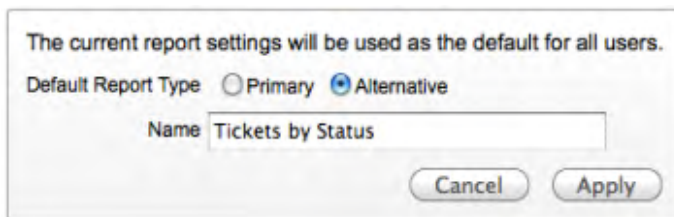


Figure 7–56. Saving on interactive report as an alternate report

You'll see that the tool bar at the top of the report has a new Reports select list that contains both your default and alternative report, as shown in Figure 7–57.

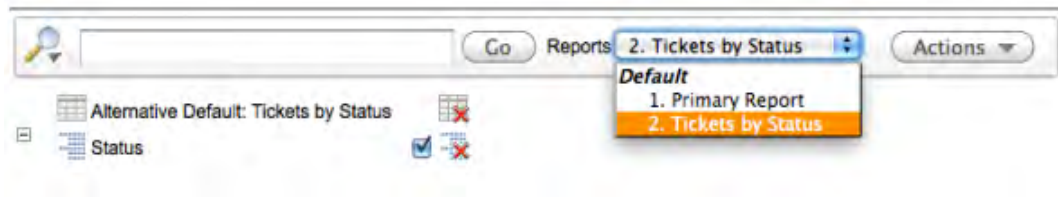


Figure 7–57. *Tickets by Status report*

You can selectively enable or disable items from the Actions menu. This will restrict what options are available to the end user for a specific interactive report.

1. Edit page 300 of your application.
2. Edit the Analysis report's interactive report properties by right-clicking on its name and selecting Edit Report Attributes.
3. Scroll down to the Search Bar region. Uncheck Flashback and Save Report from the Include in Actions Menu list, check Subscription and Save Public Report and then scroll to the top of the page and click Apply Changes (see Figure 7–58).

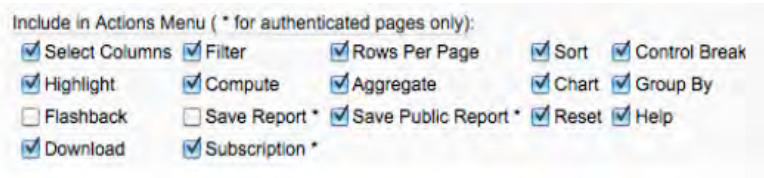


Figure 7–58. *Selecting Actions menu options*

Run your report again and click on the Actions button to expand the actions menu. Notice that the Flashback item is no longer present and that Save Report has an asterisk next to it. The asterisk denotes that only a user who is logged in as an APEX developer will be able to save this report's defaults. End users will not see this option at all. You should also see a new option for the subscriptions.

In addition to controlling which Actions appear for an interactive report, you can get even more granular and determine which columns a specific action can be performed on. Figure 7–59 shows the column-level Actions settings for your interactive report.

1. Edit page 300 of your application.
2. Edit the Analysis report's Interactive Report properties by right-clicking on its name and selecting Edit Report Attributes.
3. Edit the DESCRIPTION column.
4. In the Column Definition region, uncheck the Sort and Filter checkboxes in the Allow User To item and click Apply Changes.

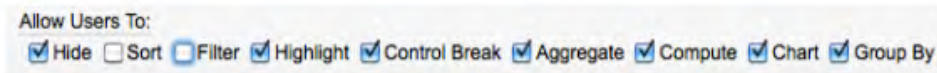


Figure 7-59. Selecting column-level actions for your interactive report

5. Run page 300 of your application.
6. Click on the Actions button and select Format → Sort. Your Sort action interface should look similar to Figure 7-60.
7. Notice that Description no longer appears as a column name in the list of columns.

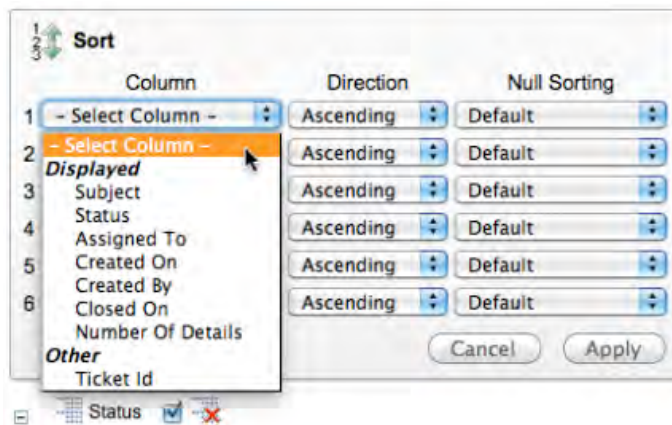


Figure 7-60. Modified select column list

By default, the interactive report will link to something called Single Row View. This view shows a read-only region that contains all of the details about a specific row. In your case, you may want to link back to the form that you created on Page 210. Thus, you can alter the interactive report to use a more traditional page link instead of the Single Row View. You do this by editing the Link Column attributes, as shown in Figure 7-61.

8. Edit page 300 of your application.
9. Edit the Analysis report's interactive report properties by right-clicking on its name and selecting Edit Report Attributes.
10. In the Link Column region, set the Link Column to Link to Custom Target.
11. Next, set page to 210, enter P210_TICKET_ID for the name of Item 1 and #TICKET_ID# for the value of Item 1 and then click Apply Changes (see Figure 7-61).

Figure 7–61. Setting the link column attributes

Run page 300 of your application. You should now be able to drill into the details of any row by clicking in the column with the Edit link..

Looking Behind the Scenes

Let's look behind the scenes for your interactive report. You may be surprised to see that there is only a single interactive report region in the Page Rendering region, as shown in Figure 7–62.

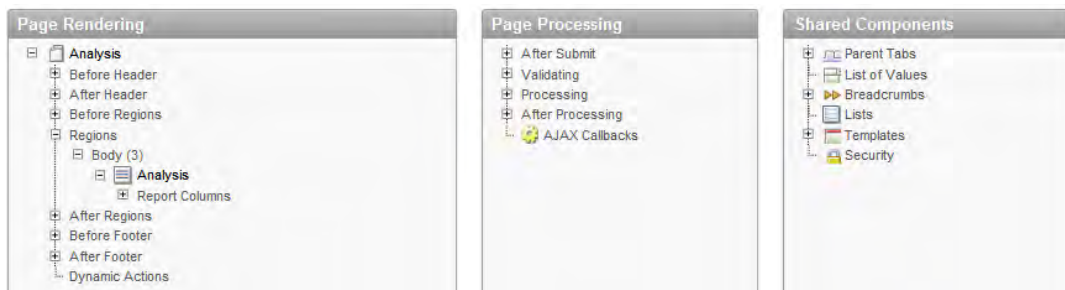


Figure 7–62. The Application Builder view of your interactive report

The Page Processing column contains no elements, and the Shared Components region contains only the expected elements for your parent tabs: the breadcrumb and templates.

This is the first case where you can't easily recreate the interactive report using standard declarative APEX elements. The additional functionality is from a collection of JavaScript functions, CSS, and HTML that are all contained within the interactive report region type. While you could build all this from scratch, the APEX interactive report is a huge timesaver.

Calendars

Sometimes there are trends in data that are not obvious when viewed in the traditional row-column format. By simply displaying data in a different method, such as a calendar report, trends can become quite obvious. The APEX calendar report can display data in a daily, weekly, or monthly view and does not require that you enter any SQL.

Understanding Calendar Types

An APEX calendar is a type of APEX report. Data is rendered on a calendar instead of in a traditional row/column format. The calendar format simply allows you to view your data in a different way. The single requirement for an APEX calendar is that the underlying table or view must have at least one DATE column.

There are two types of APEX calendars:

- **Easy/Declarative:** Created entirely by using the APEX Easy Calendar wizard.
- **SQL Calendar:** Created by entering a custom SQL query. The SQL query must contain a date column.

Data in a calendar can act as a column link, the same as any other report column. This makes it simple to build a calendar that lets the user click on a date and drill to another page or URL.

Creating a Calendar

To implement an APEX calendar, you'll create a new page and a calendar region using the Easy Calendar wizard.

1. Run your application.
2. Click on Create in the Developer's Toolbar.
3. Select New Page and click Next.
4. Select Calendar and click Next.
5. Select Easy Calendar and click Next.
6. As shown in Figure 7-63, enter 400 for the page number, Ticket Activity Calendar for both the page name and region name, and set Breadcrumb to Breadcrumb.
7. When the page reloads, enter Calendar for the entry name, and click Next (see Figure 7-64).

Figure 7–63. Creating a ticket activity calendar

Figure 7–64. Specifying the breadcrumb entry for calendar

8. Set the Tab Options to Use an existing tab set and create a new tab within the existing tab set. When the page refreshes, set Tab Set to TS1 (Home, Tickets Analysis), enter Calendar for the Tab Label, and click Next (Figure 7–65).

Figure 7–65. Specifying tabs for a calendar

9. Select your schema as the Table/View Owner, as illustrated in Figure 7–66. When the page reloads, select the TICKETS (table) and click Next.

Figure 7-66. Specifying table owner and table name for a calendar

10. For the Date Column, select `CREATED_ON`. For the Display Column, select `SUBJECT`, as shown in Figure 7-67 and click Next.

Figure 7-67. Setting column selection for a calendar

11. Click Finish.
12. Run page 400.

Upon running the calendar report, you may notice that it's hard to tell entries from one another and there's no delimiter in between them. One of the options of the calendar report is to supplement the display value with either text or HTML. You'll add a bullet at the beginning of each entry so that it will be easy to distinguish multiple entries that fall on the same day.

1. Edit page 400 of your application.
2. Edit the Calendar Attributes by right-clicking on the Calendar Region Name and selecting Edit Calendar.
3. Under the Calendar Display section, shown in Figure 7-68, set the Calendar Template to Calendar, Alternative 1; set the Display Type to Custom; and then enter the following for the Column Format: `#SUBJECT#`

Figure 7–68. Setting calendar display template and item attributes

4. Under the Display Attributes section (see Figure 7–69), change the time format to 12 Hour, set the start time to 7 am, and the end time to 7 pm.

Figure 7–69. Setting calendar display date attributes

5. Under the Column Link section, specify these values as in Figure 7–70 and then click Apply Changes:
 - Page: 210
 - Set these items: P210_TICKET_ID
 - With these values: #TICKET_ID#

Column Link

Target is a **Page in this Application**

Page **210** ☐ reset pagination for this page

Request

Clear Cache (comma separated page numbers)

Set these items **P210_TICKET_ID** (comma separated name list)

With these values **#TICKET_ID#** (comma separated value list)

URL Target

Figure 7–70. Creating a link in your calendar

While in the Calendar Report Attributes page, notice the Day Link region just below the Column Link region. These Day Link attributes are for setting a link on the calendar day to an APEX page or a URL, in the same manner as you just set a link on the SUBJECT column.

Run the page and notice that each entry in a day is much more distinguishable. Your calendar should look similar to Figure 7–71. You can now click on these entries and see the details.

Ticket Activity Calendar

Weekly Daily < Previous Today Next >

September 2010

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
29	30	31	01	02	03	04
05	06	07	08	09	10	11
12	• Disk is Full 13	• Keyboard busted 14	• Speakers are too soft 15	• Mouse is not working 16	• VPN Client Install Issues 17	• Getting Out of Memory errors 18
• Smartphone will not sync with Outlook 19	• Several dead pixels on screen 20	• Accidentally deleted Q2.ppt 21	• Funny smell coming from PC 22	• Virus Definitions Dates 23	• I think I have a virus 24	• Wireless signal not strong enough 25
• BSOD after rebooting 26	• Network drive not being mapped 27	• Need to install SP2 28	• MSIE Crashed 4 times 29	• Need more memory 30	01	02

Figure 7–71. Your ticket activity calendar

Looking Behind the Scenes

Now that your calendar works, let's look at what the Easy Calendar wizard built for you. Edit page 400. In the Page Rendering region, shown in Figure 7–72, you have three items and six buttons:

The three items are:

- P400_CALENDAR_TYPE
- P400_CALENDAR_DATE
- P400_CALENDAR_END_DATE

The six buttons are:

- MONTHLY
- WEEKLY
- DAILY
- PREVIOUS
- TODAY
- NEXT

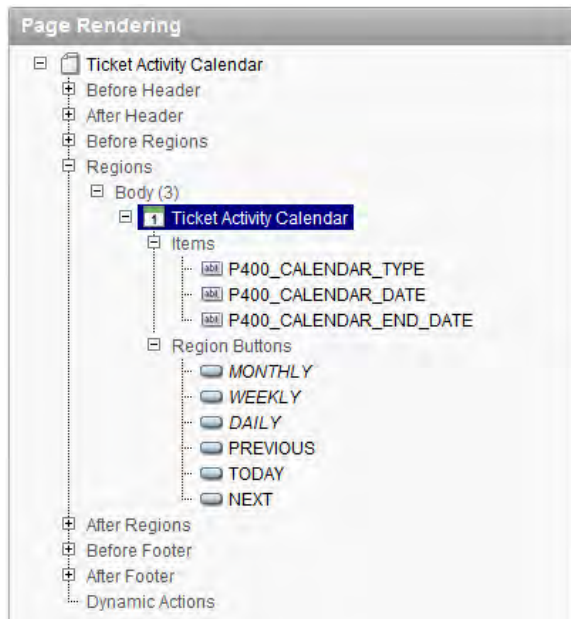


Figure 7–72. Page rendering region for your calendar

In the Page Processing section, you see a series of six processes, one corresponding to each button, for adjusting the calendar date display (see Figure 7–73). Viewing each of these processes shows that

they are each a call to the APEX utility procedure that sets the calendar display to the selected button name. Figure 7-74 shows the `month_calendar` process, which is triggered by the MONTHLY button and sets the display accordingly. There is a branch for each button and process that, in this case, keeps you on the same page.

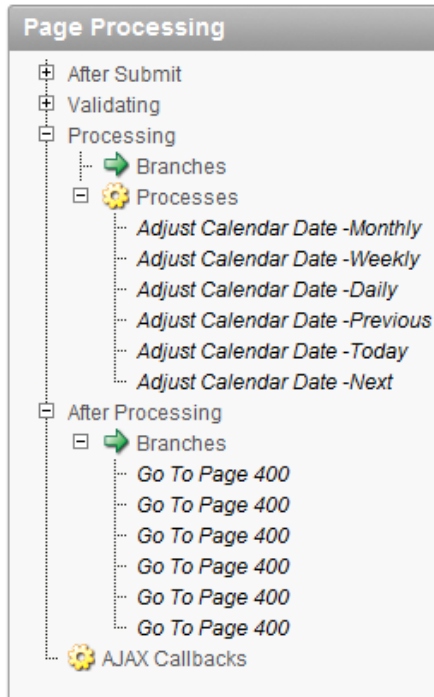


Figure 7-73. Page processing region for your calendar

```
Source
* Process [Download Source]
apex_util.month_calendar('P400_CALENDAR_TYPE');
```

Figure 7-74. The `month_calendar` process that sets the calendar view to month

Charts

In APEX 4.0, charts got a major facelift with the incorporation of AnyChart 5.1. This release of AnyChart produces charts that look much more professional than in previous releases. APEX 4.0 includes both the Gantt and Map chart types, greatly expanding declarative charting options.

Writing Queries for Charts

APEX 4.0 charts are Adobe Flash-based, include click and drill functionality, and generally operate off of a query of the type:

```
SELECT
    link,
    label,
    value
FROM
    table
WHERE
    where conditions
GROUP BY
    group by column list
ORDER BY
    Order by column list
where
```

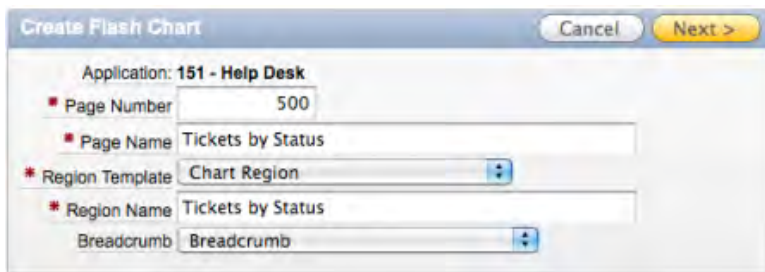
- **link** is a link to an APEX page or other URL,
- **label** is the label for the chart element, and
- **value** is the value to be charted.

The exact syntax changes slightly to suit the needs of the various chart types, but the general link-label-value format remains the same. For the correct syntax for each chart type, see the APEX online documentation.

Creating a Chart

As a simple chart example, let's build a chart of Tickets by Status in your Help Desk application.

1. Edit any page of your application.
2. Click on the Create button in the upper right of the page and select New Page.
3. Select Chart and click Next.
4. Select Flash Chart and click Next.
5. Select Pie & Doughnut and click Next.
6. Select 3D Pie and click Next.
7. Enter 500 for the page number, Tickets by Status for both the page name and region name, and set Breadcrumb to Breadcrumb (see Figure 7-75). When the page reloads, enter Chart for the entry name, as shown in Figure 7-76, and click Next.



Create Flash Chart [Cancel] [Next >]

Application: **151 - Help Desk**

Page Number: 500

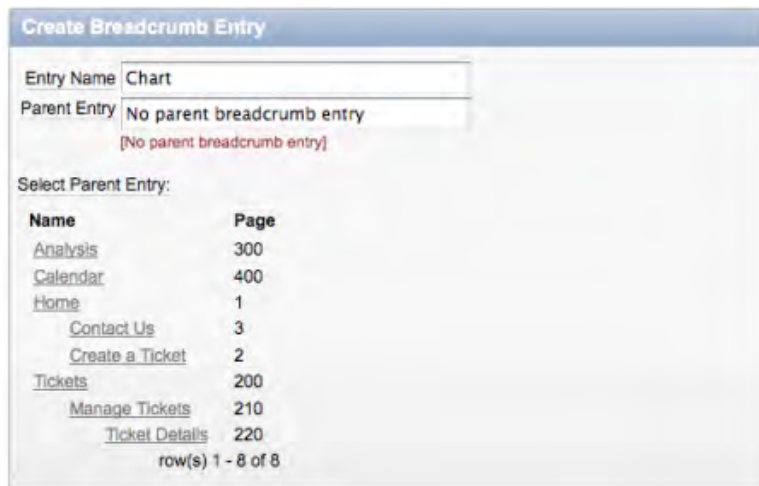
Page Name: Tickets by Status

* Region Template: Chart Region

* Region Name: Tickets by Status

Breadcrumb: Breadcrumb

Figure 7-75. Setting page, page name, and region name attributes for a chart



Create Breadcrumb Entry

Entry Name: Chart

Parent Entry: No parent breadcrumb entry
[No parent breadcrumb entry]

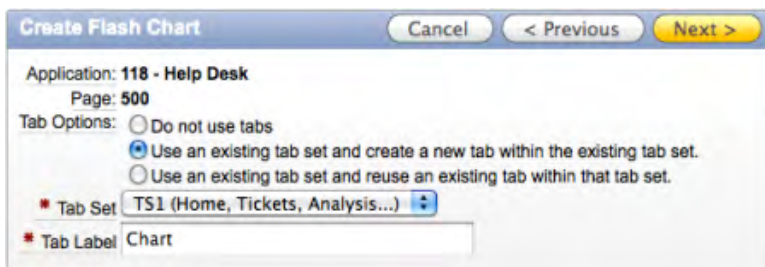
Select Parent Entry:

Name	Page
Analysis	300
Calendar	400
Home	1
Contact Us	3
Create a Ticket	2
Tickets	200
Manage Tickets	210
Ticket Details	220

row(s) 1 - 8 of 8

Figure 7-76. Setting breadcrumb attributes for a chart

8. Set Tab Options to Use an existing tab set and create a new tab within the existing tab set. When the page refreshes, set Tab Set to TS1 (Home, Tickets, Analysis), enter Chart for Tab Label, and click Next (see Figure 7-77).



Create Flash Chart [Cancel] [< Previous] [Next >]

Application: **118 - Help Desk**

Page: **500**

Tab Options:

☐ Do not use tabs

☒ Use an existing tab set and create a new tab within the existing tab set.

☐ Use an existing tab set and reuse an existing tab within that tab set.

* Tab Set: TS1 (Home, Tickets, Analysis...)

* Tab Label: Chart

Figure 7-77. Setting tab attributes for a chart

9. Set the chart title to Tickets Statuses, set the chart animation to Scale XY Center, and click Next.
10. Locate and open the file ch7_query.txt, which can be found where you extracted the book files. The contents of the file should be similar to this query:

```
SELECT
  'f?p=&APP_ID.:200:' || :APP_SESSION || '::::P200_STATUS_ID:' || sl.status_id link,
  sl.status_label,
  count(*) value
FROM
  tickets t,
  status_lookup sl
WHERE
  t.status_id = sl.status_id
GROUP BY
  sl.status_id, sl.status
ORDER BY
  3 DESC
```

11. Paste the contents of the file ch7_query.txt into the Enter SQL Query or PL/SQL Function... region, or type the above query into the region, and click Next Click Finish.

Run the page. Your chart should look similar to the one in Figure 7–78.

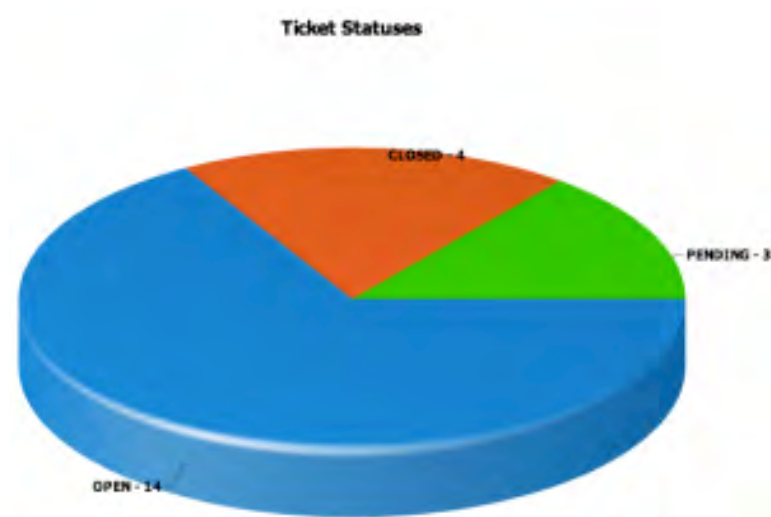


Figure 7–78. Your Tickets by Status chart

The link that we included in our SQL statement passes a Status value to the P200_STATUS_ID field on page 200. However, however you haven't created that item yet. The next steps will create the item P200_STATUS_ID on page 200 so that when a slice of the chart is clicked, the report can filter based on the status.

1. Edit page 200 of your application.
2. Create a new item by right-clicking on the Items node under the Tickets region and selecting Create Page Item.
3. Select Select List and click Next.
4. Enter P200_STATUS_ID for the item name, set the sequence to 15, and click Next (see Figure 7–79).

The 'Create Item' dialog box for 'Page: 200 - Tickets' shows the following settings: 'Display As' is 'Select List', 'Item Name' is 'P200_STATUS_ID', 'Sequence' is '15', and 'Region' is 'Tickets (1) 10'. Navigation buttons include 'Cancel', '< Previous', and 'Next >'.

Figure 7–79. Adding a P200_STATUS_ID item

5. Enter Status for the label, set Begin on New Line to No, and click Next (see Figure 7–80).

The 'Create Item' dialog box for 'Page: 200 - Tickets' shows the following settings: 'Item Name' is 'P200_STATUS_ID', 'Display As' is 'Select List', 'Label' is 'Status', 'Label Alignment' is 'Right', 'Height' is '1', 'Field Alignment' is 'Left center', 'Label Template' is 'Optional with help', and 'Begin On New Line' is 'No'. A '[Clear]' button is next to the label field. Navigation buttons include 'Cancel', '< Previous', and 'Next >'.

Figure 7–80. Setting label attributes for new item P200_STATUS_ID

6. Take the defaults for the next page and click Next.
7. Set the Named LOV to P210_TICKET_STATUS_ID, ensure that Display Null Values is set to Yes, enter – All Statuses – for the Null Display Value, enter % for the Null Return Value to set the List of Values attributes, and click Next (see Figure 7–81).

Identify List of Values

Cancel < Previous Next >

Use this page to define the list of values. Either construct a SQL statement with the number of columns required by the item type, or use the STATIC syntax. See the List Of Values Examples section for examples.

Application/Page: 116/200
 Item Name: P200_STATUS_ID
 Display As: Select List
 Named LOV: P200_STATUS_ID
 Display Null Value: Yes
 Null Display Value: -All Statuses-
 Null Return Value: %
 Cascading LOV Parent Item(s): %

* List of Values Query

Create or edit static List of Values
 Create Dynamic List of Values

Figure 7–81. Setting LOV attributes for new item P200_STATUS_ID

8. In the Default field, enter %, as shown in Figure 7–82, and click Next.

Figure 7–82. Setting the default value for new item `P200_STATUS_ID`

Finally, you will have to change the query for the Tickets report on page 200 to account for the value of the item `P200_STATUS_ID` is set to.

9. Edit the Tickets report by double-clicking on its name.
10. Append the following line to the end of the query and click Apply Changes:

```
AND tickets.status_id LIKE :P200_STATUS_ID
```

Now, run your application and navigate to the Chart page. Click on any value in the chart, and that value should be passed to the Tickets page and passed in to the Status filter. The resulting report should only display those records that correspond to the status that was click in the chart.

Looking Behind the Scenes

Viewing your Chart page in the Application Builder, you see that the only element generated is your chart region in the Page Rendering region, as shown in Figure 7–83. The chart region is interesting in that it has a Series element, which contains your SQL query. The chart region embodies the logic that passes your query to the AnyChart engine to produce your chart.

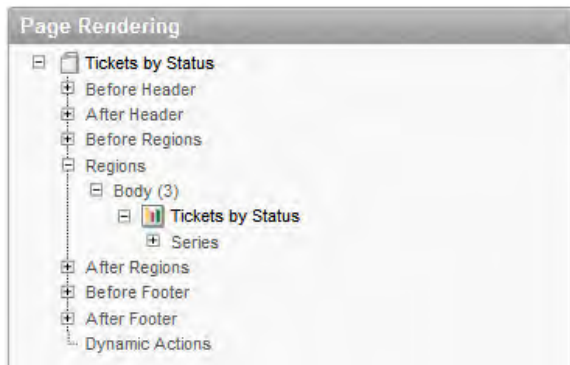


Figure 7-83. Page rendering section of the Tickets by Status chart

Double-click on the Tickets by Status chart region to open its region definition. The source region will look similar to that in Figure 7-84.

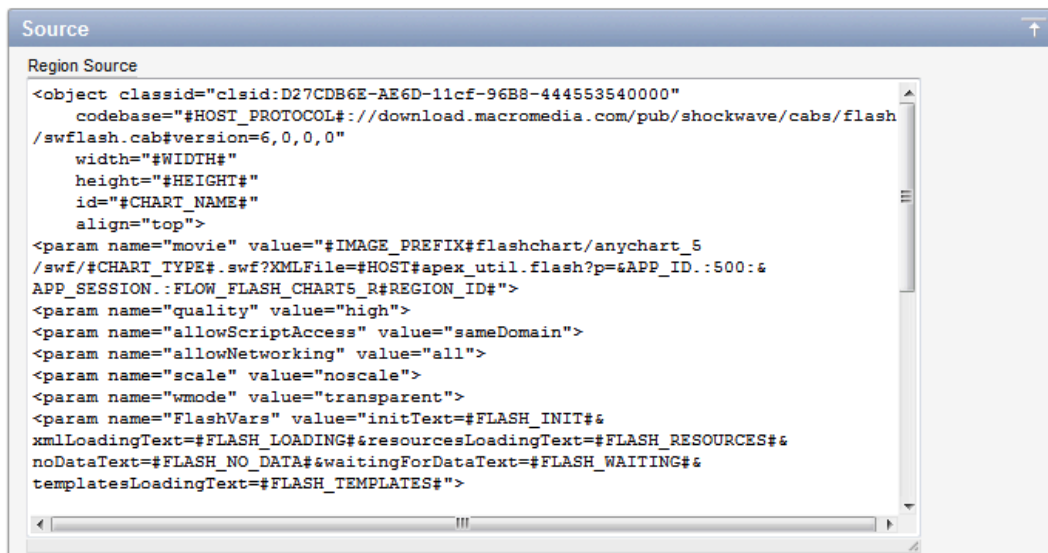


Figure 7-84. Chart region source code

In Chart regions, the Source section of the Region Definition contains not the SQL, but the XML that produces the chart. For your purposes, you will leave this XML alone. While it's possible to customize a chart by editing this XML, leave that exercise for advanced users. For more information on customizing a chart through its XML attributes, see the AnyChart documentation at www.anychart.com.

Summary

You've reviewed most of the APEX forms and reports types, and you've walked you through building various forms and reports for your Tickets System using the APEX form and report wizards. You created an interactive report and made adjustments as a developer and an end user. You have introduced charts and maps, and added a chart to your application to visualize your ticket status.

The common theme is that the APEX Form and Reports wizards are huge time-savers for developers, creating all of the objects—items, buttons, branches, processes, etc.—needed for a working form, report, or chart. You were then able to alter the created objects to quickly customize the generated form or report to suit your needs.

Still, you have not strayed far from what APEX builds for you.

As your application becomes more complex, there will be places where you wish to add additional code to enforce business rules or perform more complex processing logic than a simple insert, update, or delete. To do so, you utilize the various programmatic elements of APEX. You'll address the topics of validations, computations, and processes in the next chapter.



Programmatic Elements

This chapter covers the programmatic elements that can provide both simple and complex features to the APEX framework. APEX provides simple declarative features with wizards to guide you. Being integrated with the database, APEX can also leverage the full power of the PL/SQL engine inside the Oracle database. Now with the implementation of APEX 4.0, even JavaScript interactivity has been made declarative and extendable in the framework.

Conditions

Throughout the building of the help desk application, there are times when you will want to take advantage of the conditional logic available with APEX components. Rather than trying to understand every type of condition (there are sixty in the list of condition types), you should focus mainly on grasping the concept of a condition.

The condition feature provides a place where logic can turn on or off the particular piece of APEX technology. Before action is taken to display or execute a particular APEX component, the condition applied to that component is evaluated for a TRUE or positive result.

The logic options available to develop a condition are very broad. The condition type defines the particular mechanics used to evaluate the condition using parameters as appropriate. Simple page item comparisons are the easiest to explain. For example, a process may only need to be run if a particular page item has a value. In the case of sending an e-mail, an attempt to send a message should only be made if an e-mail address is given. From that simple start, conditions can become as complex as you need them to be. In advanced cases, conditions can also include browser and web server options.

Take time to review the condition types that are available and become familiar with their usage. It is not as important to understand the technical implementation or syntax of each item as much as what options make up a single condition. This familiarity will be helpful when you start defining APEX components and understanding considerations for a flexible and modular application design.

Validations

The purpose of validations are to assist in providing data quality and to assure integrity of data entered by the user. Mechanically, validations are tests that evaluate to TRUE or FALSE. Validations are evaluated when a page is processed, or submitted. All of the validations are evaluated; a FALSE return from any one of them prevents additional page processes from executing and, ideally, will result in feedback to the user. Validations can also be executed on the client side using JavaScript. Although the interactive nature of JavaScript can be very attractive in the user interface, it can also be circumvented easily. Any validations that are executed in JavaScript should also be supported with appropriate validations during page processing or at the database level.

■ **Note** It is a good practice to assume that every transaction is malicious. It is possible to implement validations strictly for security purposes, but sometimes it is difficult to step away from a process enough to identify where weak points may exist. For example, in a shopping cart application, what would happen to the total if someone ordered -1 of a product? Would they automatically get a credit? Take extra time in the development process to look at your application to identify where security weaknesses may exist and implement features to make it generally more robust and secure.

There are three types of validations: item level, tabular form validations, and page level. Item level validations operate against a single APEX item. Page level validations are used when multiple items are involved in validating the condition. Tabular form validations behave similar to item level validations, but are only available on tabular form items. You will use an example of each in your help desk application. First though, let's look at a special form of item level validation used to mark a value as required.

Required Values

One of the most basic validations is the value required validation. Requiring a value is a common need, and APEX 4.0 now supports that very basic validation through what is essentially a not null flag at the page item level. You don't need to create a full-blown validation to make an item required. You simply make a choice on a drop down list.

Continuing your help desk application, you will implement a value required validation on the subject field.

1. Edit page 210 of your application.
2. Edit the P210_DESCR page item.
3. Under the Settings section, change the Value Required to YES as shown in Figure 8–1. (Depending on how you set up your UI defaults, the Value Required may already be set to Yes.)



Figure 8–1. Requiring a value to be present

4. Click the Apply Changes button.

To test the new validation, start by creating a ticket. Before entering any values, click on the Create button. **Figure 8–2** shows the expected results with both a consolidated page validation message box and item validation messages. In the application, the Subject element had already been set up with a value required validation. The error messages for validations are cumulative. You will see all validation messages when a page is processed.

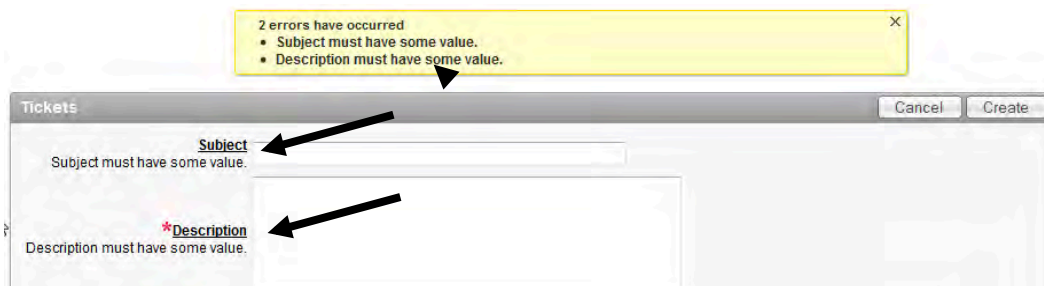


Figure 8–2. Validation shows required values for two elements both in-line and consolidated at the page level

■ **Note** The message text shown is a default and can be replaced by application specific text as a feature of globalization in the shared components area. There is only a single default for the entire application per language. When you need custom messages in a single-language application, I recommend using standard validation types that allow a different message for each validation that you create.

Item Level Validation

Validations on a single element can have attributes specific to that element and behavior can be customized as required by that element. The example you’re going to implement here is a validation that checks its condition only when a specific criteria is true. The requirement is to have an end date entered whenever the status is closed.

1. Edit page 210 of your application
2. Using the Create button, select Page control on this page as shown in **Figure 8–3**.

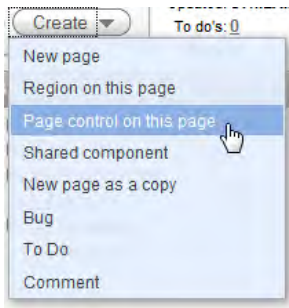


Figure 8–3. Preparing to create a validation for the page

3. Select Validation as in Figure 8–4 for the control type, then click Next.

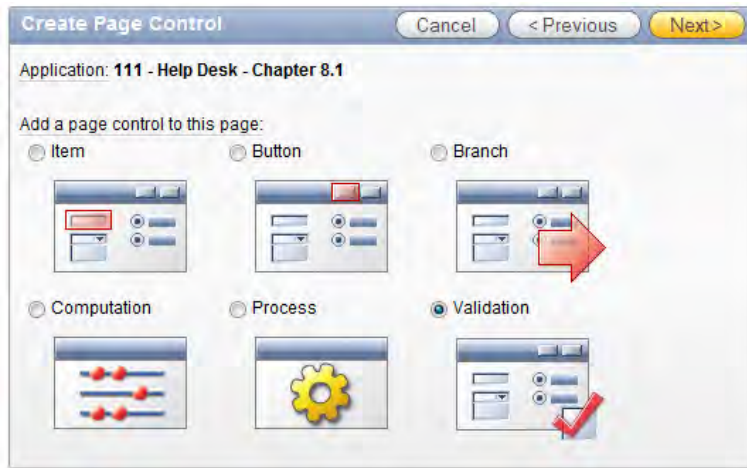


Figure 8–4. Choosing to create a validation

4. Select the Item Level Validation, which can be seen as the left most option in Figure 8–5. Click the Next button to continue.

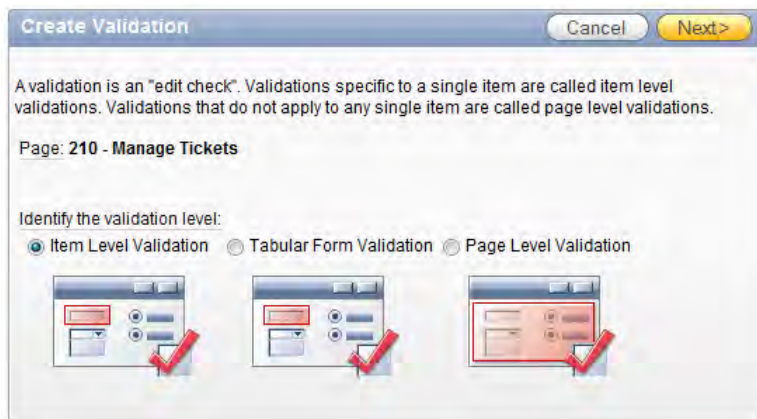


Figure 8-5. The three types of validations

5. Select the item to validate, P210_CLOSED_ON from the list of items in Figure 8-6.

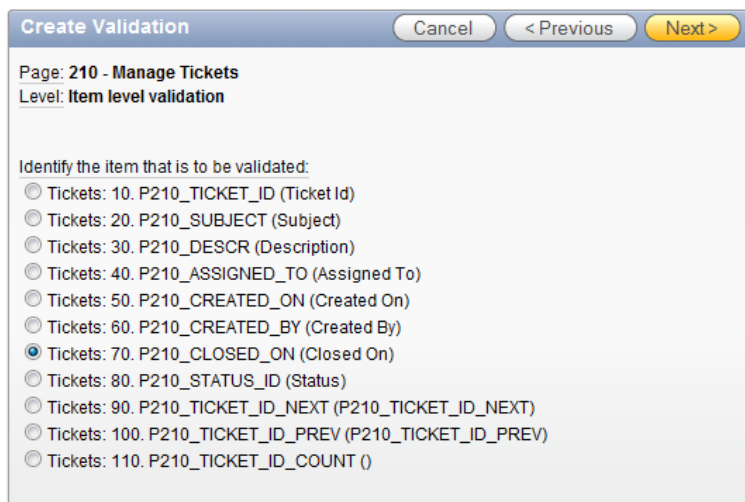


Figure 8-6. Selecting an item to validate

■ **Note** There is a shortcut for getting to the item validation. Using the right mouse button context menu on the page element, there is an option to create a validation for the element that you have selected.

6. Select Item Not NULL from the options in **Figure 8–7**.

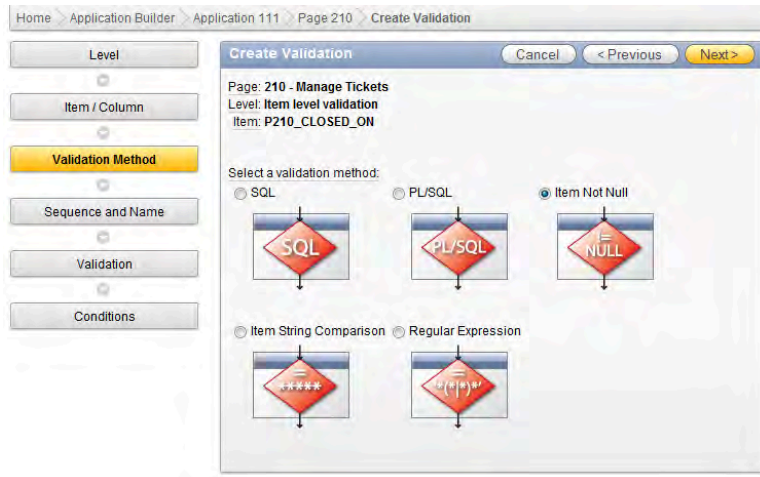


Figure 8–7. Selecting a validation method

7. Now enter a sequence number and a name. The validation sequence and name are strictly for your reference. In this case, a name is sufficient and the sequence is not important. Leave the defaults as shown in **Figure 8–8** and click the Next button.

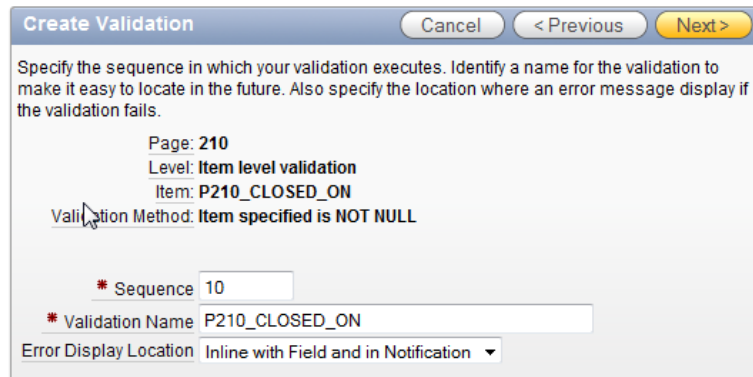


Figure 8–8. Specifying sequence, name, and display location

8. In the validation step of the wizard, leave the validate item as the default selected earlier and enter a custom error message. The error message in **Figure 8–9** uses a substitution variable #LABEL# to include the label of the item in the message. This way when the label on the form item changes in the future, the validation error message will automatically reference the new label.

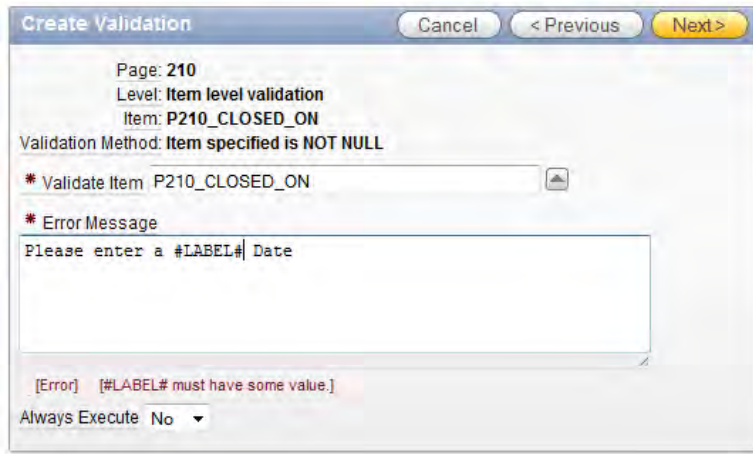


Figure 8–9. *Error Message definition*

9. The step in the wizard provides the additional functionality you want to utilize in this validation. Here you can make the validation only apply when the current status is one of CLOSED. You accomplished this by applying a condition type of PL/SQL Function Body Returning a Boolean as shown in **Figure 8–10**.

The following is the code to use:

```
IF :P210_STATUS_ID = get_status('CLOSED') THEN
  RETURN TRUE;
ELSE
  RETURN FALSE;
END IF;
```

10. Click the Create button to complete the wizard.

Create Validation [Cancel] [< Previous] [Create]

Page: 210
 Level: Item level validation
 Item: P210_CLOSED_ON
 Validation Method: Item specified is NOT NULL
 Name: P210_CLOSED_ON
 When Button Pressed: - Select Button -

Condition Type
 PL/SQL Function Body Returning a Boolean
 [PL/SQL] [item=value] [item not null] [request=e1] [page in] [page not in] [exists] [none] [never]

Expression 1

```
IF :P210_STATUS_ID = get_status('CLOSED') THEN
  RETURN TRUE;
ELSE
  RETURN FALSE;
END IF;
```

☐ Do not validate code (parse code at runtime only).

Figure 8–10. Setting the condition that must evaluate to TRUE for the validation to be used

Once the validation has been created, it will show up in rendering and page processing sections on the APEX page editor screen as illustrated in **Figure 8–11**. Both references point to the same implementation and are shown for easy navigation.



Figure 8–11. Validations created will show in two places on the builder page

This validation will now require that a value is entered for the Closed On item when the ticket status changes to CLOSED. The condition applied to the validation will be evaluated every time the page is submitted.

Page Level Validation

Page level validations apply to one or more items simultaneously and often can be an entire PL/SQL block of code that needs to evaluate true in order for the validation to be successful. Your requirement for the help desk application is to compare the Created On date with the Closed On Date to assure they occur in chronological order. A ticket that is closed before it is created does not make any sense. This is a good example of using a validation to assure data quality. Here's how to create the validation that you need:

1. Edit page 210 of your application.
2. Using the Page Create button select Page control on this page.
3. Select Validation and click Next.
4. Select Page Level Validation and click Next.
5. Select PL/SQL method as shown in Figure 8–12 and click Next.

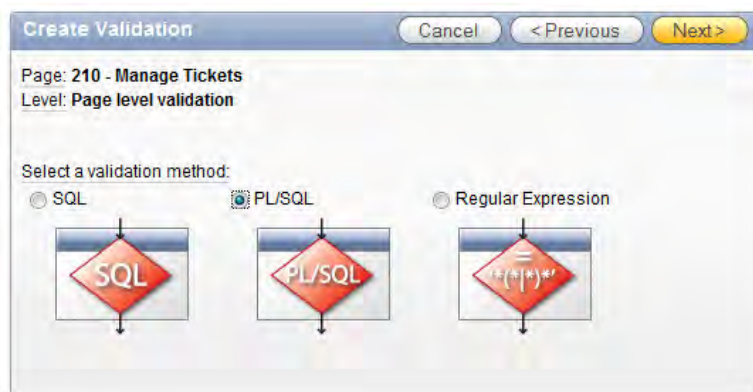


Figure 8–12. Choosing to create a validation involving PL/SQL code

6. Select Function Returning Boolean as shown in Figure 8–13 then click Next.

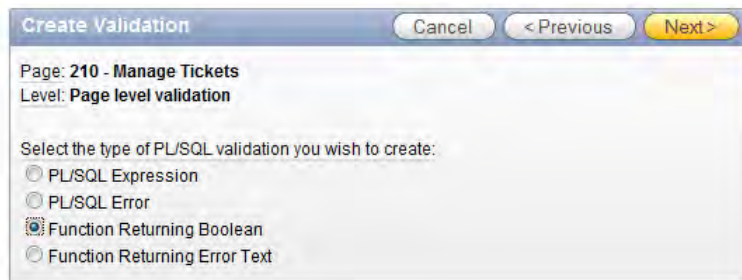


Figure 8–13. Choosing the form in which you'll implement the PL/SQL validation code

7. The validation name automatically generated for page level validations is not very descriptive. It is a good practice to give the validation a descriptive name to easily identify it. For your help desk application, enter Closed Date must be After Creation Date for the validation name. Also, change the error display location to Inline in Notification. Figure 8–14 shows the values expected. Click Next to continue.

Create Validation [Cancel] [< Previous] [Next >]

Specify the sequence in which your validation executes. Identify a name for the validation to make it easy to locate in the future. Also specify the location where an error message display if the validation fails.

Page: **210**
 Level: **Page level validation**
 Validation Method: **Function Returning Boolean**

* Sequence:
 * Validation Name:
 Error Display Location:

Figure 8–14. Validation name and display location

8. The validation step in the wizard does the bulk of the work. Here you need to enter the following code to execute the validation. You also define the error message that is displayed as a result of validation failure. For your requirement, enter a message of Closed On Date must be Later than the Created Date. Figure 8–15 shows the completed values. Click Next to continue.

```
IF TO_DATE(:P210_CREATED_ON, 'DD-MON-YYYY') >
  TO_DATE(:P210_CLOSED_ON, 'DD-MON-YYYY')
THEN
  RETURN FALSE;
ELSE
  RETURN TRUE;
END IF;
```

9. You will not need a condition on this validation. Click Create to finish the wizard.

Create Validation [Cancel] [< Previous] [Next >]

Page: 210
Level: Page level validation
Validation Method: Function Returning Boolean

* Validation

```
IF TO_DATE(:P210_CREATED_ON, 'DD-MON-YYYY') >
TO_DATE(:P210_CLOSED_ON, 'DD-MON-YYYY')
THEN
  RETURN FALSE;
ELSE
  RETURN TRUE;
END IF;
```

* Error Message

Closed On Date must be Later than the Created Date

[Error] [Validation error occurred.]

Always Execute ☐ No

Figure 8–15. The code for your PL/SQL validation

In your application you now have a feature that helps assure the quality of the data being entered. This type of data check assures that any metric that calculates time from start to end will not produce a negative answer due to dates. This improves the quality of the data and the reliability of the metrics that are produced in reports.

Tabular Form Validation

Tabular forms in APEX 4 are able to perform validations better than in previous versions. The wizard that creates a tabular form also adds validations for you. The wizard will create validations automatically based upon the data model. However, a wizard can only know so much about your business process, and the data model may have more flexibility than you want in your application. In the case of your help desk application, you don't want the subject to be blank when it is edited from the multiple tickets screen. To implement that restriction, you will apply a validation to the tabular form.

1. Edit page 230 of the application.
2. Using the Page Create button select Page control on this page.
3. Select Validation and click Next.
4. Select Tabular Form Validation and click Next.
5. In the dialog, select Manage Tickets: 4. SUBJECT option as shown in Figure 8–16, then click Next.

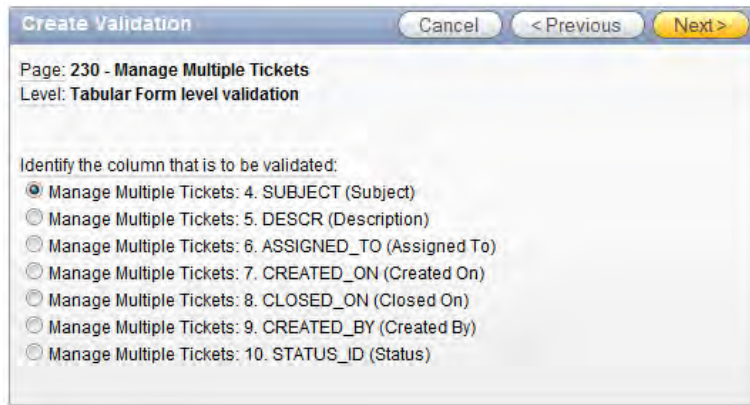


Figure 8-16. List of items available from the tabular form

6. Select Column Not Null as shown in **Figure 8-17** and click Next.



Figure 8-17. Wizard-driven options for null testing and string comparison

7. Set the validation name to Subject is Not Null and error display location to Inline in Notification as seen in Figure 8-18. Click Next.

Create Validation [Cancel] [< Previous] [Next >]

Specify the sequence in which your validation executes. Identify a name for the validation to make it easy to locate in the future. Also specify the location where an error message display if the validation fails.

Page: 230
 Level: **Tabular Form level validation**
 Column: **Manage Multiple Tickets: 4. SUBJECT (Subject)**
 Validation Method: **Column specified is NOT NULL**

* Sequence:
 * Validation Name:
 Error Display Location:

Figure 8–18. *Specifying a validation name*

8. For the error message, click on the quick link below the error message box that shows #COLUMN_HEADER# must have a value. This will automatically populate the error message using built-in substitution strings. You should see results like those in Figure 8–19. Click Next.

Create Validation [Cancel] [< Previous] [Next >]

Page: 230
 Level: **Tabular Form level validation**
 Validation Method: **Column specified is NOT NULL**
 Validation Column: **Manage Multiple Tickets: 4. SUBJECT (Subject)**

* Error Message

[Error] [#COLUMN_HEADER# must have a value.]

Always Execute:

Figure 8–19. *An error message using substitution variables*

9. No conditions are required for this validation. Simply click the Create button on the last step of the wizard.

Running the Multiple Tickets page, the new validation can be tested by either adding new tickets or removing the subjects of existing ones and attempting to save the changes. In **Figure 8–20**, each row that does not meet the validation requirement will show as highlighted and in a list of errors at the top of the page. In this example, the two rows that don't have subjects fail the validation and are flagged as needing attention.

Subject	Description	Assigned To	Created On	Closed On	Created By	Status
	User called and cannot log into his MS Outlook e-mail Account.	Scott	08-AUG-2009	08-AUG-2009	Alex	OPEN
PC will not turn on	The user's PC will not turn on when the power button is pressed.		07-AUG-2009			CLOSED
	User needs more memory installed		06-AUG-2009			OPEN
MSIE Crashed 4 times	MSIE keeps on crashing for any site	Scott	05-AUG-2009			CLOSED

Figure 8–20. Results that fail validation are highlighted and presented in the message area

The wizard shows the basic string comparison or Not Null options for validations. After the validation has been created, when edited, there are additional validation options. Note that validation names that start with the word “Column” operate on the individual rows of the tabular form. The validation can be modified to work on other discrete items on the page versus the rows of the tabular form. Validations have options to work against tabular form columns and pages items. Do not consider tabular form fields as page items because tabular form items are dynamically crated and challenging to reference as form items.

Computations

The APEX computation is analogous to a PL/SQL function. The intent is to act upon an item in the application by setting the value using a variety of methods. This allows information to be derived rather than just stored in the data tables. Computations can be implemented when a page is rendered or after a page is submitted back to the server, depending on the needs of the application. Computations can act on any item available within an application. Items that can be set include items on the current page, items on another page, and even application level items.

There is also a type of computation that can be used at the application level. This is available under an application’s shared components. This type of computation has additional options for execution points including a computation point called On New Instance that will execute when a new session (or instance) is given to a user when they login.

Execution

It is important to understand when a computation is executed relative to when a value is shown on a page and to when other values are available to the computation. When using values of items in a computation, the current session state for that item is the value that will be used. A computation sets an item value in session state, and any processing (computations, validations, or processes) that use that item after it has been set will see the results of that computation. When a page is rendered, it will show what is in the session state for that item at the time it is shown on the page. The computation point is the setting that determines when the computation is executed. On the page definition screen, there are

several computation points shown in the page tree. The computation point can be adjusted by clicking and dragging the computation in the tree to a different computation point, or by editing the computation and changing the values for sequence and computation point directly. The sequence only orders the computations within a given computation point. In general, the page renders and processes as shown on the page definition screen, starting at the top and going down the list to the bottom as shown. There are only minor exceptions such as dynamic actions and AJAX callbacks, which have variable points of execution.

Types

Computations have much of the same flexibility as other APEX components. They can be complex or simple with the full capabilities of the Oracle Database to support them. The types of computations are

- **Static Assignment:** Simple static text value.
- **PL/SQL Function Body:** Uses PL/SQL function syntax with a RETURN statement.
- **SQL Query (return single value):** Any SQL statement as long as it returns a single row and single column.
- **SQL Query (Return colon separated value):** SQL used for multi-select items.
- **SQL Expression:** Expression used in the SELECT portion of an SQL statement.
- **PL/SQL Expression:** Same as SQL Expression.
- **Item Value:** Name of an another item in the application.

Computations can be conditional in the same manner as many of the other APEX components. The conditions can be as complex as the business rules require with the ability to utilize the database features and APEX session items to evaluate the condition. Conditions evaluating to TRUE will result in the computation being executed.

Creating a Computation

Your help desk application has a requirement to display the number of days a ticket has been open. The result should be a derived value and change depending on the day and status of the record being reviewed. You will accomplish this by putting a new item on the page that displays the result of the computation.

1. Edit page 210.
2. You'll use the shortcut this time to create your page item. Use the right mouse button while over the Ticket Items section of the page rendering tree seen in Figure 8–21 and click Create Page Item.

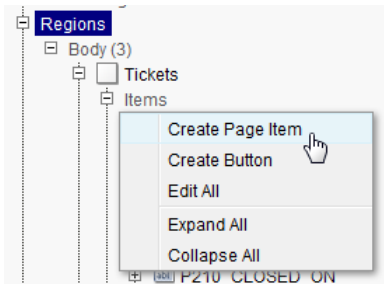


Figure 8–21. Using the shortcut menu to create a new page item

3. Select Display Only and click Next.
4. Enter P210_DAYS_OPEN for the item name. The other values can be left as defaults, as seen in Figure 8–22

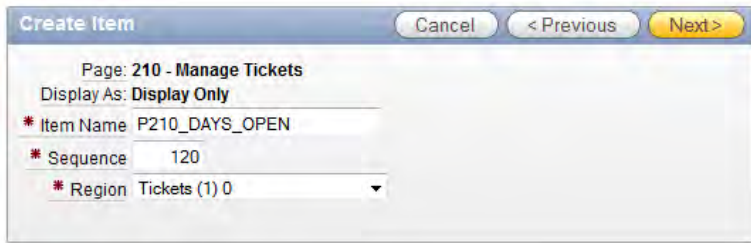


Figure 8–22. Entering an item name

5. For item attributes, leave the default values. Click Next.
6. For settings, leave the default values. Click Next.
7. For source, leave the default values. Click Create Item. Now there's a new item in the region that you will use as a container for the calculation.
8. Edit the newly created P210_DAYS_OPEN by double clicking on the item.
9. Update the condition of the item so that it shows only that P210_TICKET_ID is not null. Figure 8–23 shows the condition type as Value of Item in Expression 1 is NOT NULL and the name of the item as P210_TICKET_ID.



Figure 8-23. Showing an item only when another item contains a value

10. Click Apply Changes to complete the update of the condition.
11. Click Create Computation using the mouse context menu shown in Figure 8-24.

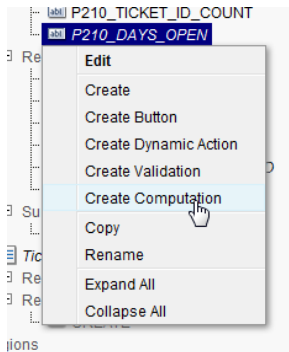


Figure 8-24. Using the right mouse button shortcut to create a computation

12. Select computation type SQL Query (return single value). All other values should default similar to those shown in Figure 8-25. Click Next when finished.

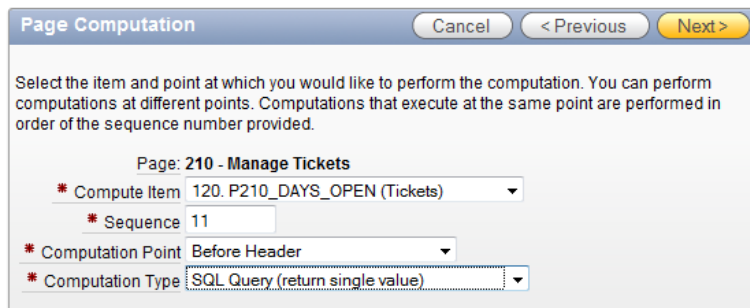


Figure 8-25. Setting the computation type

13. Enter the following SQL statement (also shown in Figure 8–26). Then click the Create button.

```

SELECT
  DECODE(status, 'CLOSED', closed, open_or_pending) days_open
FROM
  (
    SELECT
      ROUND(sysdate - t.created_on) open_or_pending,
      NVL(ROUND(t.closed_on - t.created_on),0) closed,
      sl.status status
    FROM
      tickets t,
      status_lookup sl
    WHERE
      t.status_id = sl.status_id
      and t.ticket_id = :P210_TICKET_ID)

```

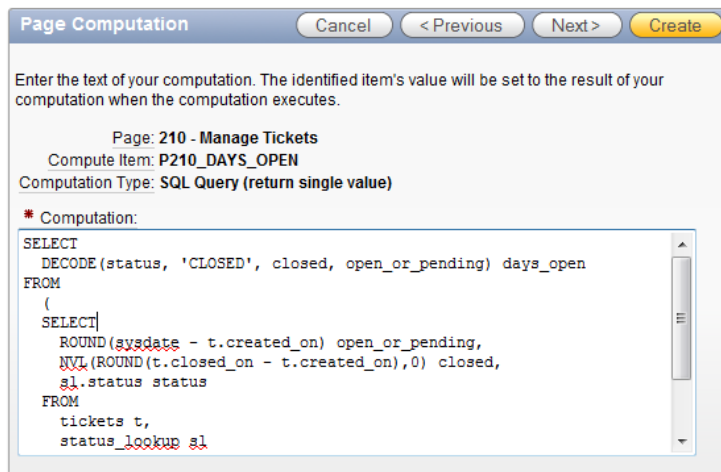


Figure 8–26. *Entering the SQL statement for a computation*

14. Optionally, move the Days Open counter next to the subject using the drag and drop editor, also located on the context menu. This editor is handy for simple layout changes, requiring only that you click and drag an item to a new location. Figure 8–25 shows the element in its new position.



Figure 8–27. *The drag and drop layout editor*

To see the results of adding the new item, run the application and navigate to the tickets report (page 200). Click on one of the Edit icons to bring up the single record view (page 210). You should now see the results of the computation as a number of days. When starting the process of creating a new ticket, the field will not be seen because the condition will prevent the field from showing.

Processes

If computations are analogous to database functions, then processes are analogous to database procedures. A process is a container for a unit of logic.

Processes are arguably the most complex part of APEX. Processes are the tool used to deal with data processing back to the database as well as references to APIs such as those used to send e-mail and perform any other business logic required in the application. When dealing with data forms, the APEX wizard will create built-in processes that manage the reading and writing of data from the form. Those types of built-in processes are called data manipulation processes.

Processes, similar to computations, can occur both during page rendering and page processing. On the page definitions editor, the processing locations are identified by the yellow gear icons. Processes support the APEX condition feature, which allows processes to be written as individual logic units with conditions determining whether the logic is needed.

Execution Points

Process execution points are the same as for computations. The most commonly used execution points for processes are “On Submit – After Computations and Validations” and “On Demand – Run this process when requested by AJAX” as these points support button press activities and dynamic actions. The full list is as follows:

- On New Instance - (new session)
- On Load - Before header
- On Load - After header
- On Load - Before regions
- On Load - After regions
- On Load - Before footer
- On Load - After footer
- On Submit - Before computations and validations
- On Submit - After computations and validations
- On Demand - Run this process when requested by AJAX

Processes can be defined at the individual page level or at the application level as part of the shared components. Functionally, page processes and application processes behave the same. The difference is where business logic is contained. For processes that need to run on all pages, an application process can be defined. Also, just as with regions, Page Zero can be used to define processes that run on every page, but only for page rendering.

Process Types

The different process types each have a different use depending on the requirement. The types and their uses are as follows:

- **PL/SQL:** General use for utilizing database PL/SQL logic.
- **Reset Pagination:** Used to reset pagination features in reports.
- **Session State:** Used to clear session state values; also referred to as cache.
- **Data Manipulation:** Built processes for reading and writing to the database.
- **Web Services:** Submits a request to a web service provider.
- **Form Pagination:** Most often used in master detail forms.
- **Send Email:** Declarative interface to easily send e-mail.
- **Close Popup Window:** Used to support pop-up window handling.
- **Run On Demand Process:** Calls an application level on-demand process.

Process in the Help Desk Application

The details behind processes can be very complex. In order to provide an adequate example, you will include a simple process in your help desk application: a requirement for the help desk applications to keep track of the last time a record was modified. You'll do this by updating a last updated date on the record every time it is saved. There's more than one way to accomplish this task. You'll do it with a process.

First, you'll need to add the LAST_UPDATED field to the TICKETS table. To do this you'll use the SQL Workshop again.

1. From the SQL Workshop Drop down menu, choose Object Browser as indicated in Figure 8-28.

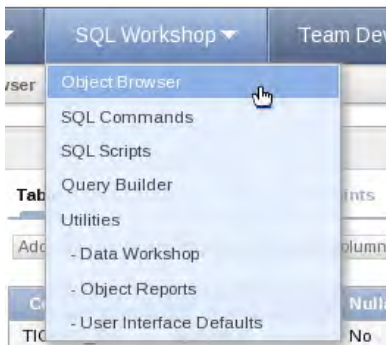


Figure 8-28. Navigating to the SQL Workshop Object Browser

2. Select the TICKETS table from the list of objects on the left hand side.

3. Click on the Add Column button above the table definition as shown in Figure 8–29.

Column Name	Data Type	Nullable	Default	Primary Key
TICKET_ID	NUMBER	No	-	1
SUBJECT	VARCHAR2(255)	No	-	-
DESCR	VARCHAR2(4000)	Yes	-	-
ASSIGNED_TO	VARCHAR2(50)	Yes	-	-
CREATED_ON	DATE	No	-	-
CLOSED_ON	DATE	Yes	-	-
CREATED_BY	VARCHAR2(50)	Yes	-	-
STATUS_ID	NUMBER	Yes	-	-

1 - 8

Figure 8–29. Navigating to the SQL Workshop Object Browser

4. Enter LAST_UPDATED for the for the column name and DATE for the type, and click Next.
5. Click Finish.
6. Now you can add the process to the page. Edit page 210 of the application.
7. Using the right mouse button context menu, click Create while over the Processes icon in the processing area, as shown in Figure 8–30.

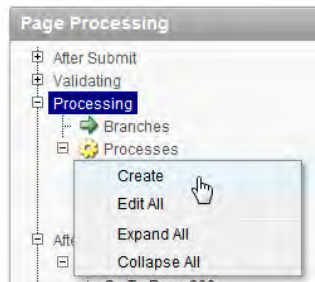


Figure 8–30. Shortcut for creating a process at a specific process execution point

8. Select PL/SQL from the list of process type options shown in Figure 8–31. Click the icon, or click the Next button.

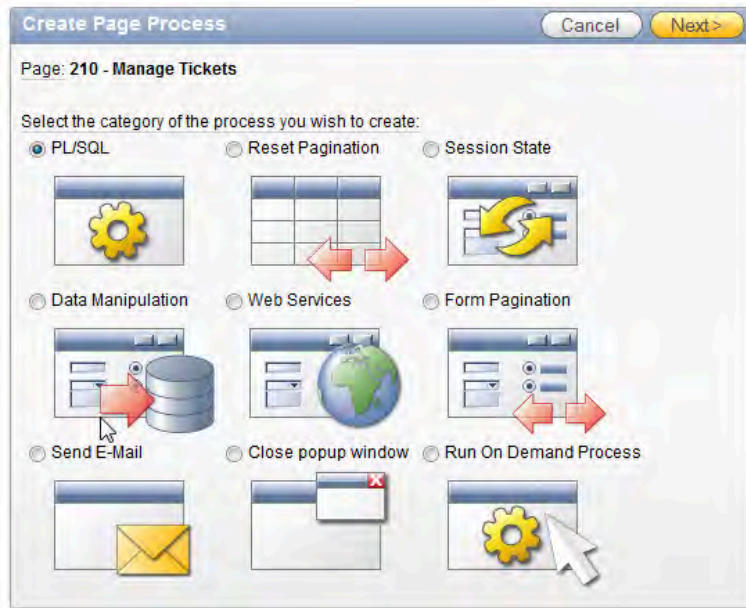


Figure 8-31. List of process types available

9. Give the process a name that identifies the purpose. You'll name the process Last Updated as seen in Figure 8-32. The default for sequence number will be sufficient, and the processing point defaults to the popular On Submit – After Computations and Validations, which fits your need. Click Next when finished.

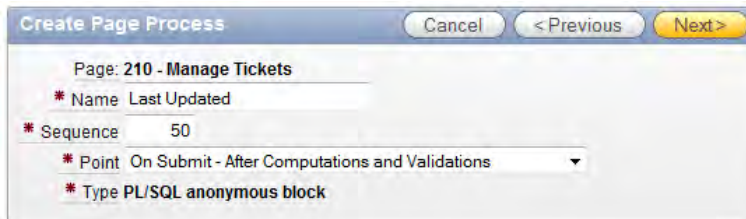


Figure 8-32. Name, sequence, and process point definitions

This is the step where you set the contents of your anonymous PL/SQL block. If you are unfamiliar with a PL/SQL anonymous block, it is PL/SQL code that has a BEGIN and an END that wrap the contents. It is necessary to follow PL/SQL syntax conventions including ending statements with semi-colons. It is possible to nest anonymous blocks of code, but is not necessary for this example.

10. Enter the SQL provided into the area provided (see Figure 8-33).


```
UPDATE tickets SET last_updated = sysdate
WHERE ticket_id = :P210_TICKET_ID;
```

11. Click the Next button when finished, not the Create Process button, as you still want to apply a condition to this process.

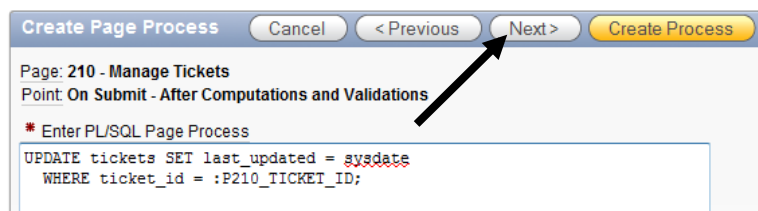


Figure 8–33. Click the Next button

12. Leave the messages empty. These messages will show up at the top of a page as a feedback to the user after the process worked as expected or had a failure. For your requirement you don't need to notify the user the update date was changed. Click Next, not the Create Process button.
13. For the condition section, change the “when button pressed” value to SAVE (Apply Changes) as shown in Figure 8–34. Finally, you may click the Create Process button.

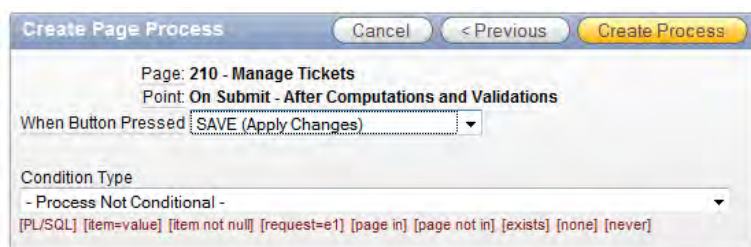


Figure 8–34. Setting the condition of the process to only work when the Save button is pressed

At this point, the process has been created. Currently, you don't show the last updated date in the summary report. In order to see the value on the report, it is necessary to add the Last Updated column to the query the report draws data from.

1. Edit page 200 (Tickets).
2. Edit the Tickets region by double clicking on the region in the tree.
3. Add the Last Updated date to the source query of the report using the change highlighted below. Click Apply Changes when finished.

```
SELECT
  "TICKETS"."TICKET_ID" "TICKET_ID",
  "TICKETS"."SUBJECT" "SUBJECT",
  "TICKETS"."DESCR" "DESCR",
```

```

"TICKETS"."ASSIGNED_TO" "ASSIGNED_TO",
"TICKETS"."CREATED_ON" "CREATED_ON",
"TICKETS"."CLOSED_ON" "CLOSED_ON",
"TICKETS"."CREATED_BY" "CREATED_BY",
"STATUS_LOOKUP"."STATUS" "STATUS"
, "TICKETS"."LAST_UPDATED" "LAST_UPDATED"
FROM
"STATUS_LOOKUP",
"TICKETS"
WHERE "STATUS_LOOKUP"."STATUS_ID" = "TICKETS"."STATUS_ID"
and upper(subject) like
'%' || upper(:P200_SEARCH) || '%'
and tickets.status_id like :P200_STATUS_ID

```

To test and review your change, run the application and navigate to the Tickets report. Edit any ticket and click the Apply Changes button. You should now see a value for Last Updated report indicating the current day.

This is a quick example of how you used a process to apply some form-based logic. When the form is used to make changes, a brief piece of PL/SQL makes a record change automatically. Packages, procedures, and APIs all would be reachable using processes similar to this one.

PL/SQL Regions

The PL/SQL region type is effectively an open container for PL/SQL with the additional option to generate output. Oracle Web Application Toolkit (OWA) procedures such as `http.p` can be used to generate the output. References to APEX items can be made using bind variable syntax (`:P1_ITEM_NAME`), the `v` function (`v('P1_ITEM_NAME')`), or the substitution string syntax (`&P1_ITEM_NAME.`) to support the logic contained in the region.

PL/SQL regions differ from process regions in that PL/SQL regions are executed only during page rendering whereas processes can run both on page processing and page rendering. PL/SQL regions have the advantage of being able to generate content directly on the page. A use case for this type of output is the need for a complex report format that is beyond the ability of a standard report template. In that case, a PL/SQL package that generates the needed HTML output can be written and called by a PL/SQL region.

In your help desk application, you want to make the home page a bit more useful by adding in a quick summary of the number of tickets an individual has open. This will only really be applicable if someone is logged in. So if they are not logged in, a simple greeting message will suffice. You will accomplish the task of adding the summary by adding a PL/SQL region with some logic to output the appropriate message.

1. Start by editing page 1.
2. Edit the APEX Issue Tracker Region.
3. In the Identification section, change the type to PL/SQL (anonymous block) as shown in Figure 8–35.

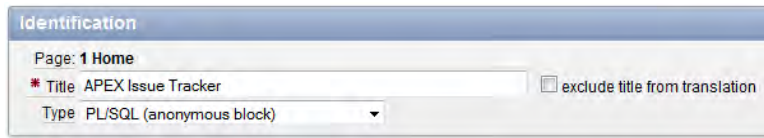


Figure 8–35. Creating a PL/SQL region

4. Enter the following code into the Region Source:

```
DECLARE
  l_count NUMBER;
  l_status_id NUMBER := get_status('OPEN');
BEGIN
  IF :APP_USER != 'nobody' THEN
    SELECT count(*)
      INTO l_count
    FROM tickets
    WHERE assigned_to = :APP_USER
    AND status_id = l_status_id;

    http.p('<h1>Welcome to the APEX Issue Tracking System, '
          || :APP_USER || '</h1>'
          || 'You have ' || l_count || ' Open tickets.<br />'
          || 'Select an option from the list');
  ELSE
    http.p('<h1>Welcome to the APEX Issue Tracking System</h1>'
          || 'Select an option from the list');
  END IF;
END;
```

The code above implements logic that makes a decision based on the user substitution variable `:APP_USER` and tailors the `http.p` output according to that distinguishing factor. APEX provides “nobody” as a username when a user is not yet logged in, so the logic keys off of that value.

When the PL/SQL region is generated for a user not yet logged in, a simple welcome message is produced (see Figure 8–36). When a user who has credentials is logged into the application, a message similar to that in Figure 8–37 is produced that shows a user-specific greeting and a quick count of the number of open tickets assigned to that user.



Figure 8–36. Issue Tracker PL/SQL region when user not yet logged in

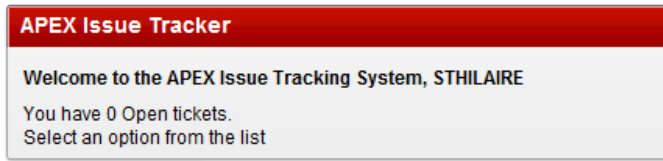


Figure 8–37. With authenticated user, PL/SQL region generates greeting and ticket count

In this section, you have created a dynamic PL/SQL region that alters the output based on the application user. This section’s example, although simple, shows how the content of a region can be as dynamic as necessary with the use of the PL/SQL in the database.

Dynamic SQL

Dynamic SQL is a term for SQL that is not finalized at design time, but rather is assembled at runtime by any number of dynamic criteria. Dynamic SQL is used when the exact requirements of a SQL statement are not known until runtime, or when the SQL needs to change while the application is running. Dynamic SQL lets you modify column lists, where clauses, joins, and any other portion of an SQL statement while an application is running.

APEX supports dynamic SQL in reports and can support PL/SQL functions returning SQL statements as a result. There are some constraints, however. Functions must return a valid SQL statement. Depending on the implementation, a statement may need to return a set of generic columns if the number of columns is not known or will vary.

Your application has the requirement to differentiate public tickets from private ones. To accomplish that goal, you will implement a Public Flag feature. Implementing the flag will require a quick update to your data model and then an implementation of Dynamic SQL on the Home Page report. You’ll start by making the data modification.

1. Navigate to the SQL Workshop.
2. Click SQL Commands icon.
3. Enter the following SQL Statement and click the Run button.

This will add the new column called `public_flag` to your Tickets table.

```
ALTER TABLE tickets ADD (public_flag VARCHAR2(1))
```

4. Enter the following SQL Statement and click the Run button.

This updates all of the current tickets to a default value of N.

```
UPDATE tickets SET public_flag = 'N'
```

Now that the data model modifications are complete, move on to the application. You will add the option to see and edit the new value in the ticket edit screen.

5. Click on the Application Builder tab.
6. Select the application by clicking the corresponding icon or link.
7. Edit page 210 the application.
8. Add an item to the tickets region by right-clicking on the region’s name and selecting Create Page Item as shown in Figure 8–38.

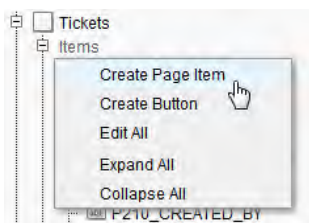


Figure 8-38. Adding a page item to be your public flag

9. Select Radio Group item type and click Next.
10. Enter P210_PUBLIC_FLAG for the item name as in Figure 8-39, select Tickets for the region, and click Next.

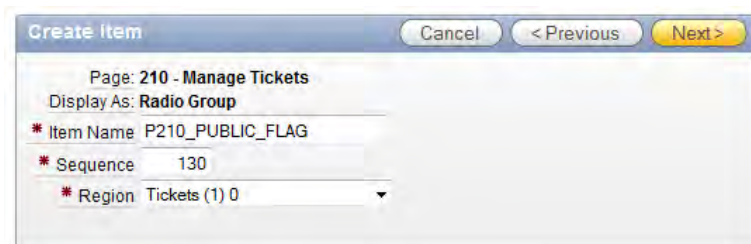


Figure 8-39. Specifying the item name

11. Set the label name to Public Flag and the label template to Required with Help as in Figure 8-40. Click the Next button when finished.

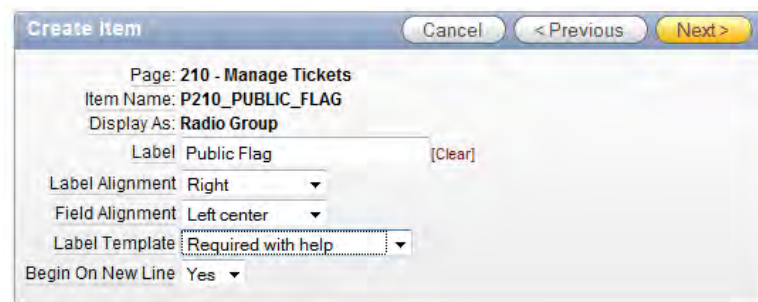


Figure 8-40. Specifying display attributes

12. Set the value required field to Yes and set number of radio columns to 2, as in Figure 8-41. Click Next when finished.

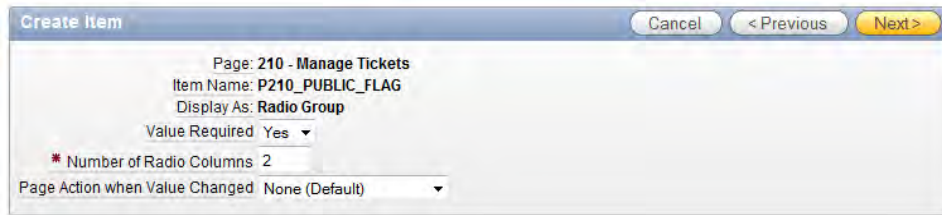


Figure 8–41. Marking the value as required

13. Set display null value to No, and enter STATIC:Y,N for the List of Values query shown in Figure 8–42 . Click Next when finished.

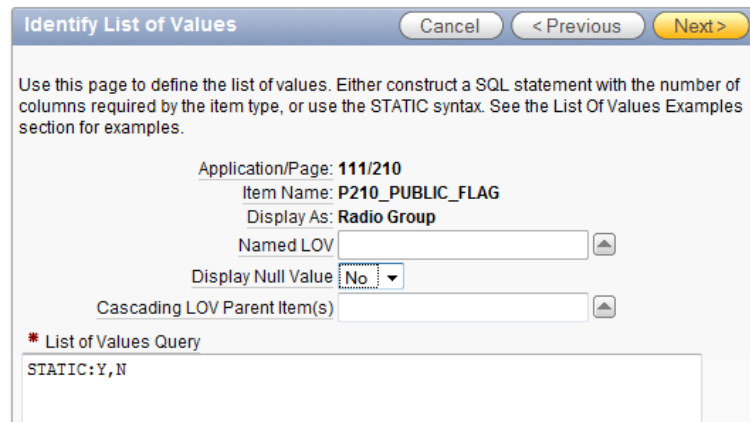


Figure 8–42. The LOV for the flag

14. Set the source type to Database Column, ensure that the database column name is PUBLIC_FLAG, enter N for the default (as seen in Figure 8–43), and click Create Item button.

Figure 8–43. Specifying a database column and default value

15. Use the drag and drop tool to move P210_PUBLIC_FLAG so that it is next to P210_STATUS_ID, similar to Figure 8–44. This is strictly for ease of reading and usability. Click Next and then Apply Changes to finish the layout modification.

Figure 8–44. Layout modification for easier usability

Now that you have a PUBLIC_FLAG column in your data model and the ability to control it through the Tickets form, you can create the Dynamic SQL report on Page 1 to display tickets with a public option for unauthenticated users.

1. Navigate to and edit Page 1 in your application.
2. Create a new region by clicking the Create button and selecting Region on this page, as indicated in Figure 8–45.

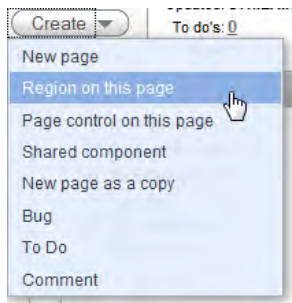


Figure 8–45. Creating a region for the SQL to generate your report

3. Select Report and click Next.
4. Select SQL Report and click Next.
5. Enter Current Open Issues for the title, as shown in Figure 8–46. Click the Next button when finished.

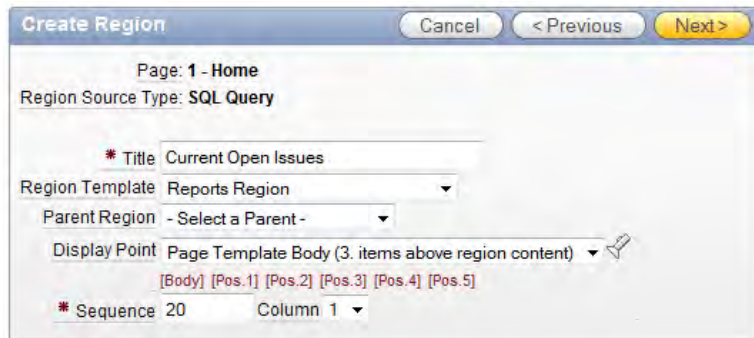


Figure 8–46. Region title and display point

6. Enter the following SQL into the region source. Click the Create Region button when finished to default all of the remaining settings.

```
DECLARE
  l_sql VARCHAR2(255);
BEGIN
  l_sql := l_sql || '
SELECT
  subject,
  created_on,
  assigned_to
FROM
  tickets t,
  status_lookup sl
WHERE
```



```

        t.status_id = sl.status_id
        AND sl.status = 'OPEN'
    !';

IF :APP_USER = 'nobody' THEN
    l_sql := l_sql || q'! AND public_flag = 'Y' !';
END IF;

RETURN l_sql;
END;
```

To see the results of this report fully, you need to have set a few tickets with the new public setting. Navigate to the tickets summary screen as a logged in user and change a few OPEN tickets to have the PUBLIC option set to yes. When navigating to the home screen as a logged in user, a full list of open tickets should be seen as in Figure 8–47. After logging out, you will see only the tickets that have been identified as PUBLIC.

APEX Issue Tracker

Welcome to the APEX Issue Tracking System, STHILAIRE

You have 0 Open tickets.
Select an option from the list

Current Open Issues

SUBJECT	CREATED_ON	ASSIGNED_TO
Need more memory	06-AUG-2009	-
MSIE Crashed 4 times	05-AUG-2009	SCOTT
Need to install SP2	04-AUG-2009	ISABELLA
Network drive not being mapped	03-AUG-2009	ISABELLA
BSOD after rebooting	02-AUG-2009	SHANNON
I think I have a virus	31-JUL-2009	OWEN
Funny smell coming from PC	29-JUL-2009	ISABELLA
Accidentally deleted Q2.ppt	28-JUL-2009	SHANNON
Smartphone will not sync with Outlook	26-JUL-2009	SCOTT
VPN Client Install Issues	24-JUL-2009	SHANNON
Mouse is not working	23-JUL-2009	ISABELLA
Speakers are too soft	22-JUL-2009	SCOTT
Disk is Full	20-JUL-2009	SHANNON

1 - 13

Figure 8–47. Resulting report generated from dynamic SQL

■ **Note** I used a quoting syntax in the SQL statement that you may not be familiar with. Oracle Database 10g introduced a quoting mechanism for string literals that allows you to define your own string delimiters, removing the need for double up single quotes in strings. Any character can be used as a delimiter, including bracket combinations `() {} [] <>`. The basic syntax is `q'X string X'` where `X` is any single character. The `q'X` opens the literal string and the `X'` closes the literal string. More details on the literal syntax can be found in the *Oracle Database SQL Language Reference*.

Dynamic Actions

One of the new features available with APEX 4 is dynamic actions. They provide a way to define complex client-side behavior declaratively within the APEX framework. APEX will use the information provided to generate the required JavaScript code which is then implemented at runtime. Dynamic actions are event-driven just as manually written JavaScript would be. One of the major advantages of using the declarative configuration is that APEX understands its own objects such as regions and items, allowing easy reference and manipulation.

Major Parts

A dynamic action has three major parts:

- **When:** The event on what type of item will cause the dynamic action to be evaluated. This can be as simple as an item changes when entered by a user. It can also be a complex trigger on multiple items simultaneously.
- **True Actions:** A list of actions to take when the event registers as True. There can be more than one event declared; each event will have its own set of parameters and instructions.
- **False Actions:** A list of actions to take when the event registers as False. Often the opposite of the True actions, but also not required. Some types of actions will logically not have a corresponding negative.

As with other parts of APEX, dynamic actions support conditions, authorizations, and build option features. Another advantage of the declarative nature of dynamic actions in APEX is the inclusion of the actions in the user interface. When a dynamic action is tied to an item event, it shows in the designer as part of the tree. This exposes the usage of the JavaScript events to be easily seen and recognized.

Dynamic Actions in the Help Desk Application

For an example, let's walk through adding a dynamic action to the help desk application. When creating a dynamic action, the first choice is to choose between the two types of standard or advanced. The standard dynamic action is a simpler wizard that defaults the event trigger to the value of change and allows the choice of actions: show, hide, enable, and disable. In contrast, the advanced wizard provides many additional options for events and action choices. Don't be fooled. Regardless of which wizard option you choose, after the dynamic action has been created, the ability to edit any of the settings is the same.

1. Edit page 3 of your application.
2. Right-click the P3_FROM item and choose Create Dynamic Action from the context menu. Using the menu shown in Figure 8–48 is the most direct way to create a dynamic action.

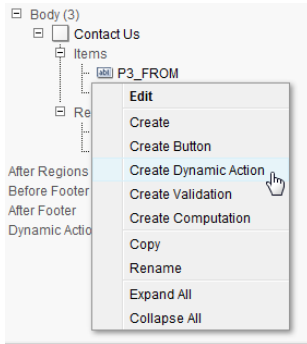


Figure 8–48. Using right mouse shortcut to create a dynamic action

3. Select Standard for the type of dynamic action, as shown in Figure 8–49, and click the Next button.

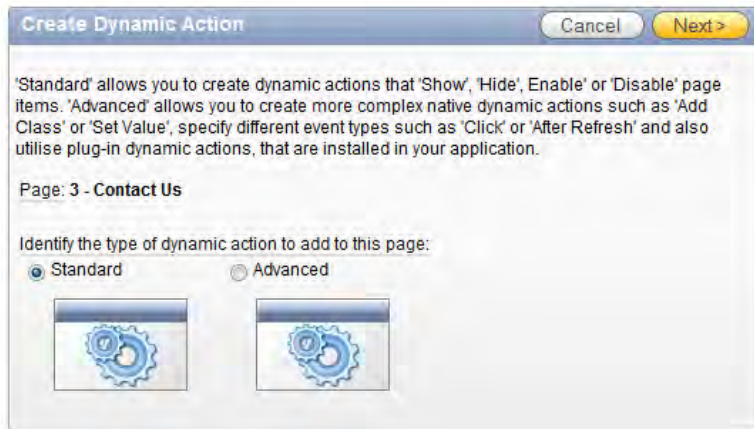


Figure 8–49. Selecting the type of dynamic action to create

4. As shown in Figure 8–50, enter Disable E-Mail Body for the name of the dynamic action. Just as with other components, the more descriptive the name, the easier it is to identify.
5. Click the Next button.

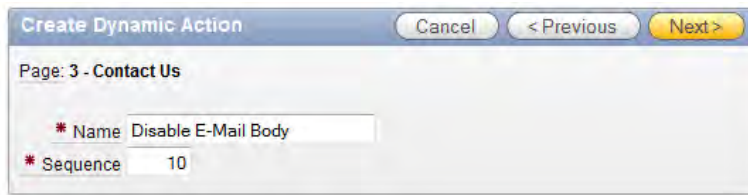


Figure 8-50. Specifying a name

6. Set condition to is null, as shown in **Figure 8-51**, and click the Next button.

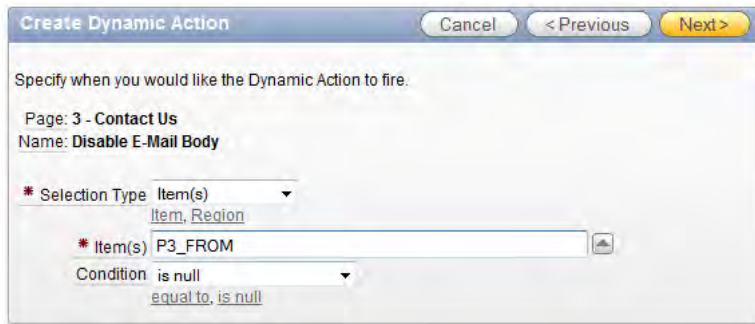


Figure 8-51. Setting a dynamic action's condition for execution

7. Set the Specify the True Action option to Disable, and select the Create Opposite False Action check box, as seen in Figure 8-52. Click the Next button.

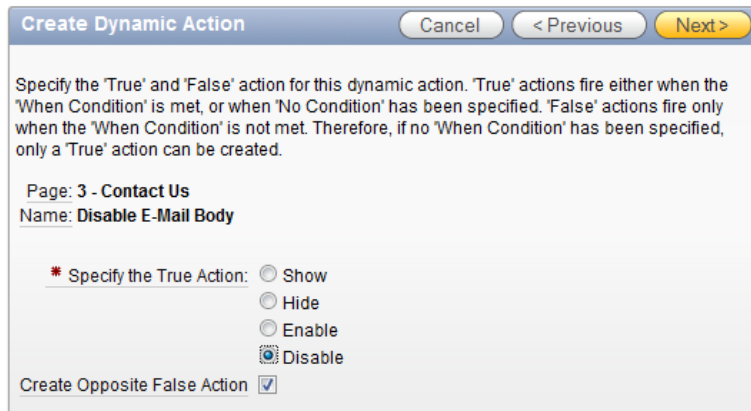


Figure 8-52. Asking to create only the action for when the condition is false

8. Set the selection type to Item(s) and in the resulting item(s) shuttle, add P3_BODY to the right region by double-clicking or using the shuttle buttons. The end result should look like **Figure 8-53**. Click the Create button when complete.

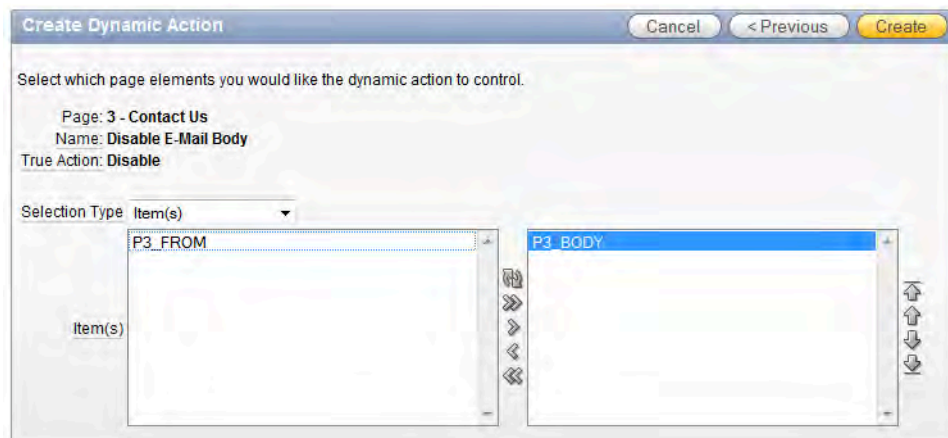


Figure 8-53. The end result before confirming the creation

Now, run page 3. Note that that the message item is disabled until you enter something in the From item and navigate away. Conversely, if you delete all content from the From item, the Message item will again become disabled. This is OK, but it would be nicer if the Message item became enabled as soon as you type anything in the From item. Once a standard dynamic action is created, you can edit its properties and access many more options that were not available in the wizard. Let's set the triggering event to be key release instead of the default change event.

1. Edit page 3 of the application.
2. Double-click on the Disable E-Mail Body dynamic action in the page rendering tree to edit. In **Figure 8-54**, you'll see two dynamic actions with the same name. It's the same action; it is shown in two places because of the reference to the item P3_FROM. This makes it convenient to find actions that are specifically against an item, or a list of all dynamic actions. At the bottom of the tree, all dynamic actions on the page will be listed even if they do not reference a specific item.

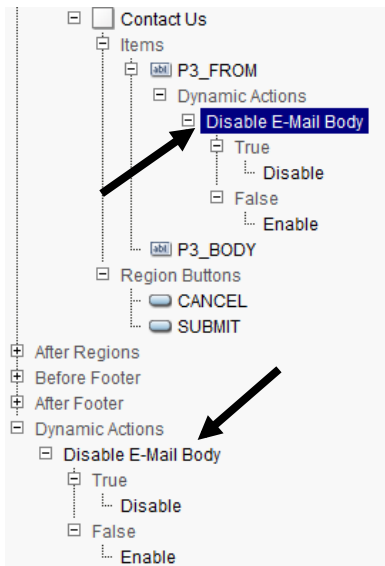


Figure 8-54. Dynamic actions in the page rendering tree

3. In the When region shown in Figure 8-55, change the event to Key Release and click Apply Changes.

Figure 8-55. Specifying key release as the event

To test this change, start by running page 3 again. The page will open looking like **Figure 8-56**. Start typing an address into the From item. As soon as any value is entered, the Message item becomes enabled as in **Figure 8-57**. Conversely, the moment you delete all content, the Message item becomes disabled again.

Figure 8-56. Before a value in the From field

Figure 8-57. After a value in the From field

Here you used the Standard Dynamic Actions wizard to create an event to disable and enable an APEX form item. You then proceeded to modify the event to behave differently using additional options available to the dynamic action. The end result is a form that responds to input from the user allowing better interactivity with the user.

Summary

As with any programming language or framework, learning the basics is the first step. In this chapter, I have touched on a lot of points that could be considered “tips of icebergs.” Each section has capabilities to reach into a vast set of technologies, with the Oracle database being primary among them. The intention here is demonstrate how the APEX framework works through the example application and to provide a starting point for additional detail discovery.



Security

The subject of security has varying degrees of implementation; it's never a black and white answer. The question of how much security is needed is followed up by additional questions regarding the value of what is being protected and the risks, repercussions, and likelihood of it being sought after. For every security measure, there will always be someone trying to circumvent it. In this chapter, we will review the basic security features and an approach to securing the Ticket application. The concepts reviewed here apply to all APEX applications and are specific to the APEX framework.

User Maintenance Navigation

In your help desk application, you have the requirement to allow users to be maintained inside of the application through the web interface. You will add a section to maintain the user accounts and modify the tab structure to navigate to the newly created form. The form needs to be created first as tabs require a page to reference.

1. From the application editor screen, click the Create Page button.
2. Select the option for Blank Page.
3. Set the page number to 600, then click the Next button.
4. Enter Users for the name field and set the breadcrumb selection to Breadcrumb. when the page refreshes, ensure the entry name is Users and click the Next button.
5. Select No for the “Would you like to use tabs for this new page?” option. Click the Next button.
6. Click Finish to complete the creation of the page. The completed page should be empty as shown in Figure 9–1.



Figure 9–1. The newly created page is currently empty with a breadcrumb entry.

Now that you have a Users page, you will make a modification to your navigation tabs. The web design you desire will have links on the right hand side of the top navigation bar in the location of parent tabs on the page template. APEX supports one and two level tab navigation as part of the Shared Components. When the application was created, the option chosen was the one level tab navigation. To

implement two level tabs navigation, you will modify the default page template for the application and modify the tab navigation structure.

■ **Note** Tab maintenance in APEX can be confusing. The easiest change is to go from some level of tabs to no tabs, or from no tabs to one or two levels. In this exercise, you will walk through the steps involved with the converting the existing one level tab to a two level tab configuration to demonstrate the complex configuration change.

Here's the process to follow to implement two levels of tabbed navigation:

1. Navigate to the Shared Components area of your application.
2. Under User Interface, click on Themes.
3. Change the report to list view by clicking on the corresponding display icon as shown in Figure 9–2.

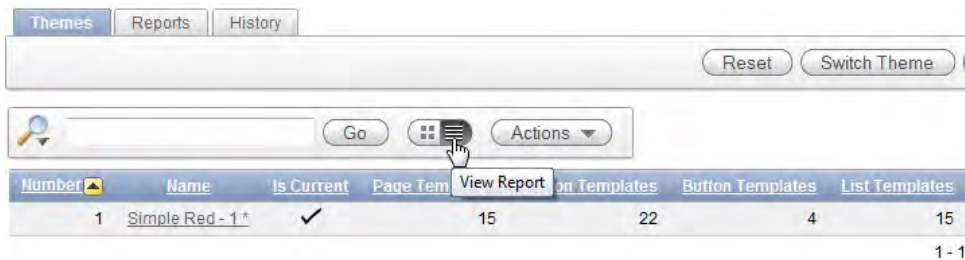


Figure 9–2. Select the list view. The navigation point for the name link will take you to the definition of the theme.

4. Click on the name of the current theme as indicated by the check mark in the Is Current column.
5. Under component defaults, set Page to Two Level Tabs – Right Sidebar (optional / table-based) and click Apply Changes.
6. Navigate back to the Shared Components page.
7. Under Navigation, click Tabs.
8. Click on the Manage Tabs sub-tab shown in Figure 9–3.

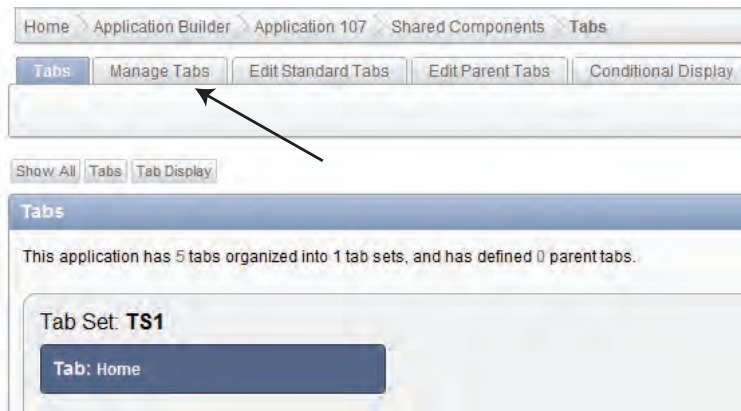


Figure 9–3. The Manage Tabs sub-tab for graphical representation of tabs

9. Add a parent tab by clicking on the Add link in the upper right corner, as indicated in Figure 9–4.

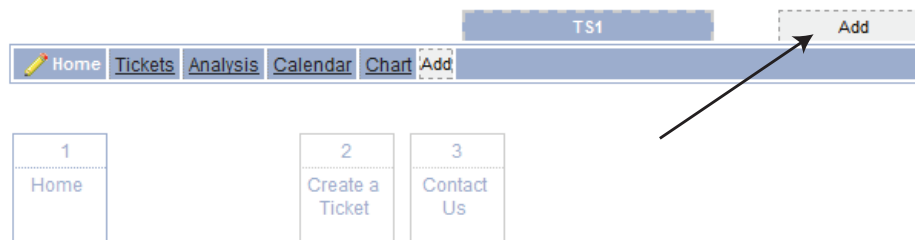


Figure 9–4. The parent tabs are added using the add button.

10. Enter Issue Tracker for the parent tab label and click Next.
11. Enter 1 for the page and click Next.
12. Click Create Parent Tab.
13. Add another parent tab by clicking on the Add link, just as before.
14. Enter Admin for the parent tab label and click Next.
15. Enter 600 for the page and click Next.
16. Set the sequence to 20 and click Create Parent Tab.
17. Click on the Admin tab and then click on the Add button in the lower-left corner indicated in Figure 9–5.

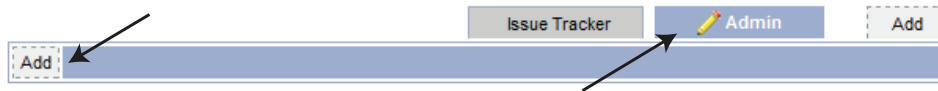


Figure 9–5. After clicking the Admin tab, the child tabs shown only apply to that parent.

18. Enter Users for the tab label and click Next.
19. Enter 600 for Tab Current for Page and click Next.
20. Leave the sequence value as the default. Click Next.
21. You do not need a condition at this time, so leave the default condition type. Click Next.
22. Click Create Tab.
23. Edit the Users tab by clicking on the Edit icon next to the tab. The icon is shown in Figure 9–6.



Figure 9–6. The Edit icon is the small pencil next to the tab name.

24. As shown in Figure 9–7, enter 610 in the Tab Also Current for Pages field.



Figure 9–7. Setting the Tab Also Current for Pages value will show the active tab style when the application is displaying any of the page numbers listed.

25. Under Parent Tab Set, set the Parent Tab Set to TS1as, as shown in Figure 9–8.

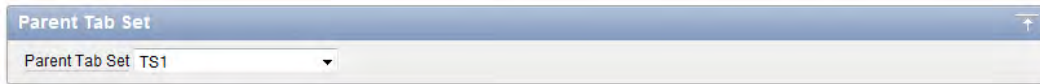


Figure 9-8. The Parent Tab Set value

26. Scroll to the top of the page and click Apply Changes.

Running the application now shows the results seen in Figure 9-9 for the Issue Tracker tab and Figure 9-10 for the Admin tab. The page that is currently active changes the highlight applied to the different tab elements. The parent tab location is dependent on the template used. In the template shown below, the parent tabs are located in the upper right. The standard tabs in this template are the same as they are for the one level tab layout.

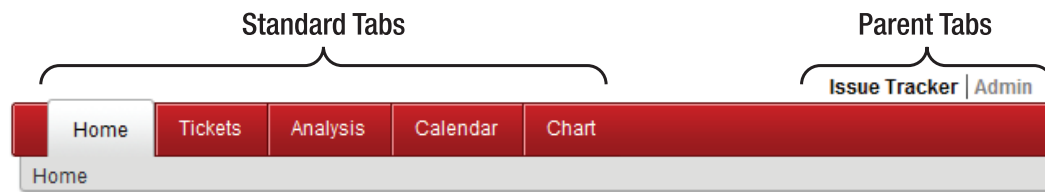


Figure 9-9. Issue Tracker parent tab shows the five standard tabs.



Figure 9-10. Selecting the Admin parent tab changes the standard tabs shown.

You now have a navigational framework to allow clear distinction of items needed to do the administration of the application. This design is extensible. As the application grows with time, additional features requiring administration could also be added to this navigational structure.

User Maintenance Data Entry

As part of the help desk design, you want to be able to maintain the users from the application. You will configure a table in the database as well as a form and report to allow this information to be maintained by any admin. Locate, upload, and run the script `ch9_security_objects.sql`. Refer to Chapter 4 if you need step-by-step instructions. You should see 13 rows, all of which complete successfully.

1. Edit Page 600 of your application.
2. Create a new region by clicking on the Create button and selecting Region on this page.
3. Select Form and click Next.
4. Select Form with Report and click Next.

5. Select your schema for the table/view owner and click Next.
6. Enter USERS for the table/view name and click Next.
7. Set the implementation to Classic, enter 600 for the page number, enter Users for both the page name and region title, and set the region template to Reports Region. The settings will look like those in Figure 9–11. Click Next when finished.

Create Report Page [Cancel] [< Previous] [Next >]

The report page is used to select the rows to be edited. It also includes a button to create a new row. If the page you specify does not exist, the page will be created.

Owner: **APRESS**
 Table Name: **USERS**
 Implementation: Classic
 * Page Number: 600
 * Page Name: Users
 * Region Title: Users
 * Region Template: Reports Region
 Report Template: template: 1. Standard
 Pagination Size: 15
 Breadcrumb: - do not add breadcrumb region to page -

Figure 9–11. Report page setup

8. Select USER_ID and USER_NAME and then click Next. Be sure not to select PASSWORD, as it is not needed in the report.
9. Select any edit link image and click Next.
10. Enter 610 for the page number and Manage Users for the page name and region title, as shown in Figure 9–12. When finished, click Next.

Create Form Page [Cancel] [< Previous] [Next >]

Specify page and region information for the Form Page. The Form Page is used to insert, update, and delete rows from the selected table. If the page you specify does not exist, the page will be created.

Owner: **APRESS**
 Table Name: **USERS**
 * Page Number: 610
 * Page Name: Manage Users
 * Region Title: Manage Users
 * Region Template: Form Region

Figure 9–12. Defining the name on the Manage User form

11. Ensure that the primary key is set to USER_ID and click Next.
12. Select Existing Trigger for the primary key source and click Next.
13. Select the USER_NAME and PASSWORD shown in Figure 9–13 and click Next.

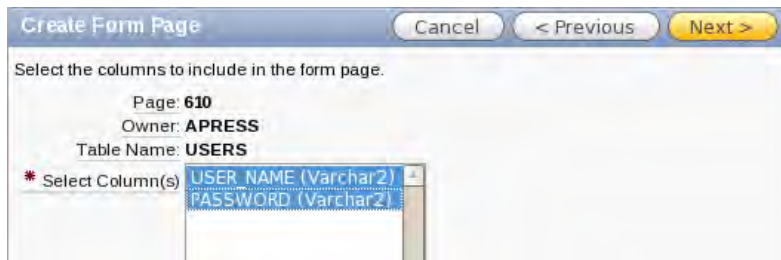


Figure 9–13. Select USER_NAME and PASSWORD as fields to be seen in the form

14. Set Insert, Update, and Delete all to Yes and click Next.
15. Click Finish.

At the completion of these steps, the help desk application has some additional objects. There is the region on page 600 that is the report of the current users. You will also notice a new page that allows editing of the data values, including all of the processes to do the corresponding database transaction. However, there are a few things that you still need to do for the tabs to display properly for Page 610.

16. Edit page 610 of the application.
17. Edit the page attributes by double-clicking on the name of the page.
18. In the Display Attributes region, set the standard tab set to T_ADMIN (Users) and click Apply Changes, as shown in Figure 9–14. This will use the tab set when rendering the page.

■ **Note** This is a different setting than the tab setting that identifies page 610 as being current. If a tab set shows on the page but without a tab highlighted as active, it is because the page is not identified as current in the tab settings. Conversely, if the page is listed as active in the tab settings, but the page does not render any tabs, no tabs will be seen.

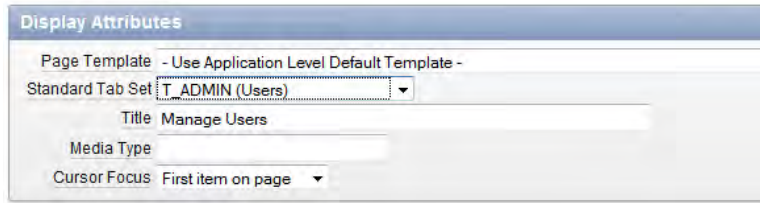


Figure 9-14. Setting the page tab set will use the tab set when rendering the page.

Page 610 does not yet have a breadcrumb entry associated with it. You can quickly add one.

19. Edit Page 600.
20. In the Shared Components region on the page, edit the breadcrumb by expanding the breadcrumbs node in the tree and double-clicking on Breadcrumb:(No corresponding region) seen in Figure 9-15.

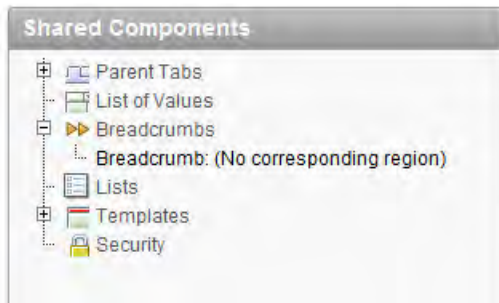


Figure 9-15. Breadcrumbs are shown as a shared component on the page editor.

21. On the filter applied to the report page, select the breadcrumb named Breadcrumb (if it has not already been selected), clear any entry from Page, and click Set. This will show all of the breadcrumbs for the entire application.
22. Click the Create Breadcrumb Entry button.
23. Enter 610 for the page the breadcrumb will associated with. Set the parent name to Users (Page 600), enter Manage Users for short name, enter 610 for page in the Target section, and click Create. The settings are indicated in Figure 9-16.

Figure 9–16. Breadcrumb settings for page 610 as a child of page 600

When completed, page 610 will have a Shared Components breadcrumb entry just as page 600. Running the application will show a breadcrumb entry for the Users Report page and the Manage Users page as shown in Figure 9–17.

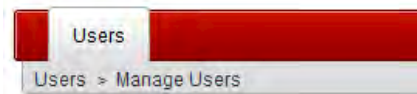


Figure 9–17. Clicking on the Create or Edit buttons of a user record will navigate to page 610, showing the Manage Users breadcrumb.

■ **Note** The breadcrumb entry label has (No corresponding region) as part of the name because there is no breadcrumb region on the current page. So why does it show up when the application is run? In Chapter 5, you moved the breadcrumb region to Page 0 so it shows up on every page rendered.

Finally, you need to change the item type of P610_PASSWORD to Password, so that it will accept a user's input but display * characters as the password is typed. This item type is designed not to retrieve data when a record is edited, despite being bound to a database column. Also, the item type will not save any value in session state, meaning it will not remember the value entered after the page processing is

complete. This is a security feature to prevent data identified as a password from being retrieved inappropriately.

24. Edit page 610.
25. Edit the item P610_PASSWORD.
26. Set Display As to Password, as shown in Figure 9–18. Click Apply Changes.



Figure 9–18. Setting the P610_PASSWORD element to a password field

As a security measure, when a user account is created, it should require a password to be set.

27. On page 610, right click the P610_PASSWORD element and click Create Validation.
28. Select Item Not Null and click Next.
29. Set the validation name to P610_PASSWORD is not null and click Next.
30. Enter “A password must be specified” for the error message and click Next.
31. For a condition, set When Button Pressed to CREATE (Create). Click the Create button when finished.

This completes the navigation and user interface part of the security scheme you will be implementing. With the navigation and maintenance in place, you can now implement the authentication scheme that will utilize the information.

Authentication

The key step in making a secure application is to understand who the accessing user is. APEX refers to this as authentication. Authentication will answer the question of “who are you?” The APEX tool provides a series of pre-defined authentication mechanisms including a built-in authentication framework and an extensible custom framework. At design time, it’s easy to switch between authentication methods by setting the active scheme. There can only be one active authentication scheme at a time for an application. The following are the major types of authentication schemes:

- **APEX Credentials:** Users are managed inside the APEX workspace and are maintained just as workspace developer accounts.
- **LDAP:** The user is an existing LDAP-compliant server such as Active Directory or Oracle Internet Directory.
- **Oracle Single Sign On:** A documented setup to allow authentication to pass between APEX and an existing Oracle SSO server. Logging into the SSO server once will pass the same credentials to all APEX applications.

- **Database Schema Users:** Database user names and passwords determine authentication; not to be confused with data access in an APEX application.
- **Custom:** Allows logic to be determined by the developer. An example of usage is for Internet-facing applications where self-registration may be desired. Another example is when more than one authentication source is used simultaneously such as using two LDAP servers.
- **Open Door:** Used to simulate logging in as different individuals by developer testing. This is not intended to be used as a public authentication scheme.
- **None:** This option is intended to allow all parts of the application to be reachable without needing a user to log in.

Each application has its own set of authentication schemes managed as part of the Shared Components part of the application. Authentication schemes can be copied between applications when needed. This ability to copy is especially useful when a custom authentication scheme has been developed and is desired in more than one application. The authentication schemes also utilize the APEX subscription framework to allow a master copy to be applied to subscribers inside of a single workspace.

Custom Authentication Schemes

In the previous section, the script that was imported included definitions for tables, triggers, and functions. You will use those elements as part of your custom authentication scheme. The key component of the authentication scheme is a function that will compare the given user name and password to the stored values in the USERS table. If there is a match, then the user will be authenticated. You can review the database objects and PL/SQL function code from the SQL Workshop for more details on how this is implemented.

■ **Note** Although the USERS table contains a field named password, it's not the actual password value; it's an encrypted hash of the password. Passwords should not be stored as plain text.

Here's the process to follow to create your authentication scheme:

1. Navigate to the Shared Components of your application.
2. In the Security region, click on Authentication Schemes; it's the first one in the list shown in Figure 9–19.



Figure 9-19. Security settings are part of the shared components of an application.

3. Click the Create button in the upper right of the Authentication Schemes tab.
4. Select From Scratch and click Next.
5. Enter Custom Authentication Scheme for the name and click Next.
6. Leave the Page Sentry function blank and click Next.
7. Leave the Session Verification function blank and click Next.
8. For Invalid Session Target, select the radio button next to Page in This Application. After the page refreshes, set the page to 101 Login, as shown in Figure 9-20. Click Next to continue.

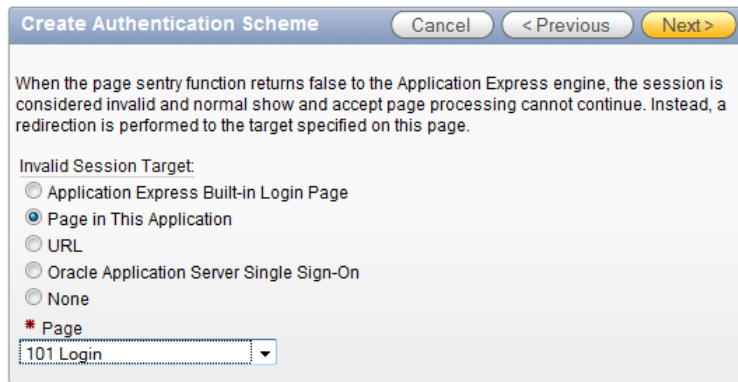


Figure 9-20. Set the login page that will be seen by users when they need to authenticate.

9. Leave the Pre-Authentication Process blank and click Next.
10. Set the Credentials Verification Method to Use my custom function to authenticate. When the page refreshes, enter RETURN authenticate_user into the Authentication Function region and click Next. Figure 9-21 shows the settings applied.

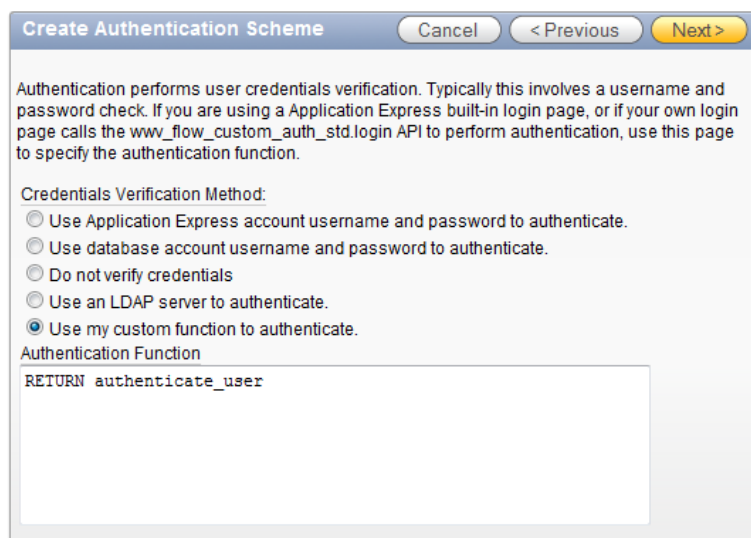


Figure 9–21. Custom authentication function setting

■ **Note** There are no parameters used here or any PL/SQL semicolon. This is part of the definition of how APEX handles custom authentication functions. The `authenticate_user` function that was imported earlier conforms to the expected signature: Function returning a BOOLEAN value with two parameters of `p_username varchar(255)` and `p_password varchar2(255)`.

11. Leave the Post-Authentication Process blank and click Next.
12. Leave the cookie name, path, and domain blank and click Next.
13. The logout value utilizes the standard APEX Logout API. The API is used to invalidate the session cookie and redirect to another APEX page after logout. Because of the complex syntax, a copy of the needed value is included as part of the book files under the name `ch9_logout.txt`. Click Next after setting the logout URL.

`wwv_flow_custom_auth_std.logout?p_this_flow=&APP_ID.&p_next_flow_page_sess=&APP_ID.:1`

■ **Note** The last parameter is the page where the user will be redirected after logout. The given URL will redirect to page 1.

14. Make sure that the settings match those in Figure 9–21 and click Create Scheme.

Create Authentication Scheme

Cancel < Previous Create Scheme

You have requested to create an authentication scheme with the following attributes:

Name	CUSTOM AUTHENTICATION SCHEME
Page Sentry Function	(Built-In)
Session Verification Function	(Built-In)
Invalid Session Target	101
Pre-Authentication Process	(Built-In)
Authentication Function	RETURN authenticate_user
Post-Authentication Process	(Built-In)
Cookie Name	(Built-In)
Cookie Path	(Built-In)
Cookie Domain	(Built-In)
Secure	N
Logout URL	www_flow_custom_auth_std.logout?p_this_flow=&APP_ID.&p_next_flow_page_sess=&APP_ID.:1

Figure 9–22. Authentication scheme settings

Now that you have your custom authentication scheme, you need to activate it, or make it the current scheme.

15. Click the Change Current sub-tab shown in Figure 9–23.
16. Set the Available Authentication Schemes to CUSTOM AUTHENTICATION SCHEME and then click Next.
17. Click Make Current.

Authentication Schemes Subscription **Change Current** History

Change Current Authentication Scheme Next >

Current Authentication Scheme: **Application Express**

Available Authentication Schemes: CUSTOM AUTHENTICATION SCHEME

Figure 9–23. Change current authentication scheme

Run your application and if logged in, log out.

Your application is now using the values stored in the USERS table and will no longer accept your APEX credentials. You can sign on as any of the following users: Scott, Doug, Martin, Karen, Patrick, or Tim; all passwords are “apress” in lower case.

Conditional Security

Many aspects of APEX are conditional. One pair of conditions are particularly applicable to the Authentication status: User is the Public User and User is Authenticated. For applications that expose parts to unauthenticated users, this ability to allow functions to public users is very useful.

By applying security rules to the help desk application, usability can be improved by restricting the display of tabs that are not available to the public. This avoids confusion and improves the overall user experience when accessing the application. Let's walk through the application of this condition.

1. Edit page 1 of the application.
2. In the Shared Components region, edit the Admin tab by expanding the parent tabs node and double-clicking on Admin. Figure 9–24 shows the location of the Admin tab in the navigation tree.

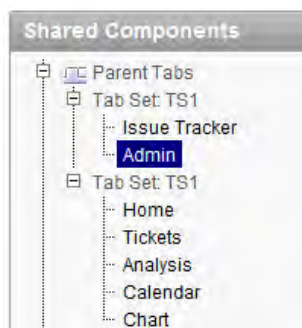


Figure 9–24. The tabs are grouped under tab sets

3. In the Conditional Tab Display region, set the Tab Display Condition to User is Authenticated (not public) and click Apply Changes. Figure 9–25 shows the expected value of the condition.

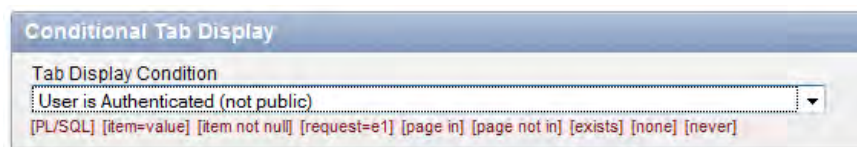


Figure 9–25. Display condition will prevent the tab from displaying until the user is logged in.

4. Repeat steps 2 and 3 for the Tickets, Analysis, Calendar, and Chart tabs.

Run the application now and click the logout link. The Admin parent tab as well as the remaining standard tabs should disappear, with the exception of the Home tab. Logging in again should restore the display of the tabs as they were previously seen.

Access Control and Authorization

APEX includes a built-in feature for creating an access control framework with three roles: Administration, Edit, and View. The wizard is designed to create data structures to store the roles, pages to edit the assignments, and authorization schemes to be utilized throughout an application. This wizard makes the job of creating basic security capability very easy in an application. The summary of the objects created can be seen in Figure 9–27 as the last step in the wizard.

1. Run the application and click Create in the developer's toolbar
2. Select New Page and click Next.
3. Select Access Control and click Next.
4. Enter 620 for the administration page number and click Next
5. Select Use an existing tab set and create a new tab within the existing tab set, allow the page to refresh, and then set the tab set to T_ADMIN (Users), as shown in Figure 9–26.

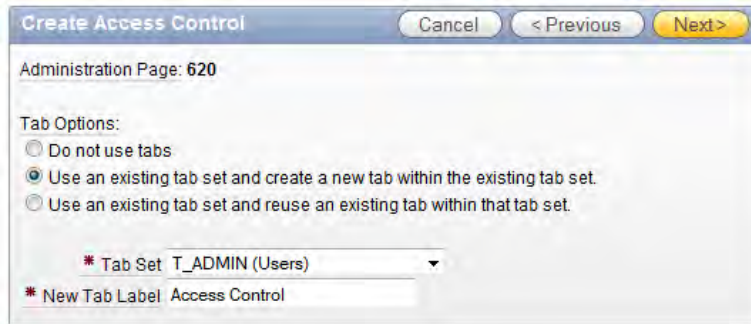


Figure 9–26. Assign page 620 to a new tab under the Admin tab set

6. Enter Access Control for the new tab label and click Next.
7. Click Finish as shown in Figure 9–27.

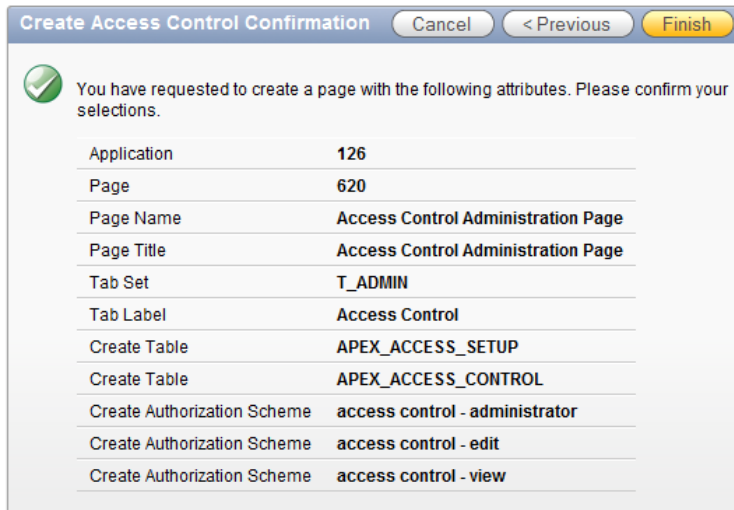


Figure 9–27. A summary of objects that will be created as part of the Access Control wizard

With completion of the wizard, all of the objects have been created and are available for use. Before you enable the security utility, you need to add some users to allow you to use the admin functions. Running the application now, you may notice that the user name is simply an open text field. You should create a list of values (LOV) as a shared component that contains all of the users for whom you want to control access. As the access control page is now part of the application, you can alter it as needed. To increase the quality of the data entered, you will update the user field to be a select list.

8. Edit page 620.
9. Edit the report attributes for the Access Control List report.
10. Edit the ADMIN_USERNAME column.
11. In the Column Attributes region, set Display As to Select List (query based LOV).
12. In the List of Values region, enter the following SQL Statement in the List of Values Definition and click Apply Changes.

```
SELECT user_name d, user_name r
FROM users
```

Next, you need to associate a privilege with each of your users with the Access Control pages.

13. Run your application.
14. Click the Access Control sub-tab.
15. Click Add User.
16. Select Scott for the username, set the privilege to Administrator and click Add User.
17. Select Doug for the username, set the privilege to Edit and click Add User.
18. Select Patrick for the username, set the privilege to Edit and click Add User.

19. Enter Martin for the username, set the privilege to View and click Apply Changes.

Your results should look similar to those in Figure 9–28. Every time a new user is added, the listing in the report will update. You can now use these users to test the application.

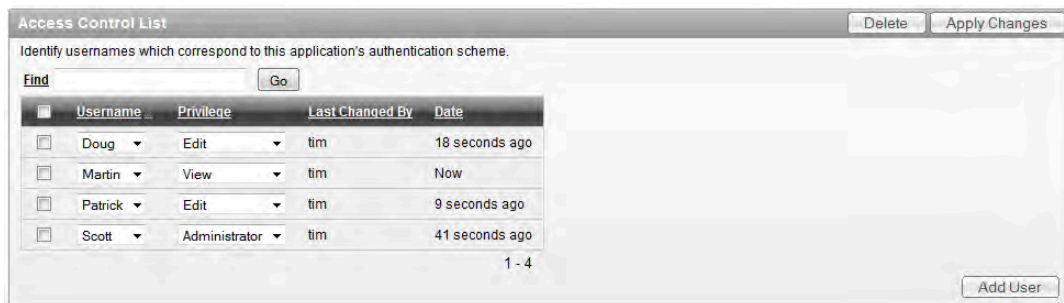


Figure 9–28. The Access Control list with user names and privileges

One of the features of the Access Control utility is the ability to enable or disable the enforcement of the utility itself. Running page 620 will show the header seen in Figure 9–30. By default, the Access Control utility is set to full access. To enable the Access Control features, you set the mode using the following steps:

20. Run page 620.
21. Select the third application mode to Public read only... and click the Set Application Mode button seen in Figure 9–29.

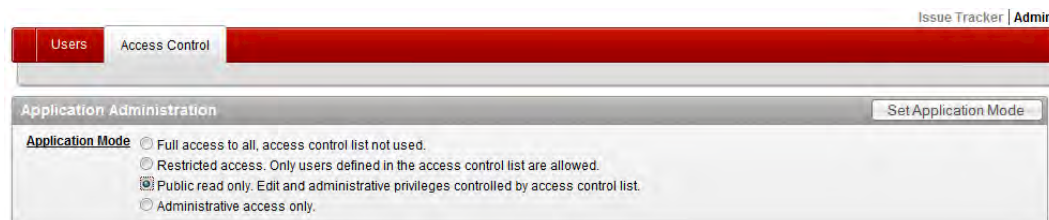


Figure 9–29. The Access Control list enabled as public read only

You now have the editing forms in place and all of the data setup properly, although the application is not utilizing any of the authorization schemes. The last step in this process is to start locking down pages using the authorization schemes.

22. Edit page 620.
23. Edit the page attributes by double-clicking on the page name.
24. In the Security region, set the authorization scheme to access control – administrator, as shown in Figure 9–30. Click Apply Changes when done.

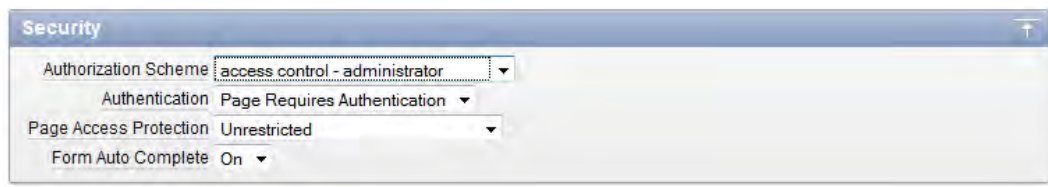


Figure 9-30. Setting the authorization scheme at a page level

25. Repeat steps 23 and 24 above for pages 600 and 610.

Now that the new authorization scheme has been implemented, you can test the security behavior. Only a user setup with the Administrator setting on the Access Control page will be able to use the admin pages 600 through 620.

Log into the application as the user Scott and you will be able to navigate all of the administration functions. Logging in as any other user and clicking on the Admin parent tab will result in the message shown in Figure 9-31.



Figure 9-31. Error message generated when the authorization scheme returns a denied result

The error message in Figure 9-31 is not very friendly. An application should make every effort to avoid the type of event that would cause a privilege error. In this application, the Admin tab should be removed from the page when it does not meet the access restrictions. You will accomplish this using the same authorization scheme applied to the tab itself.

26. Edit page 600 in your application.
27. Expand the Parent Tabs node under the Shared Components region, and double-click on Admin indicated in Figure 9-32.

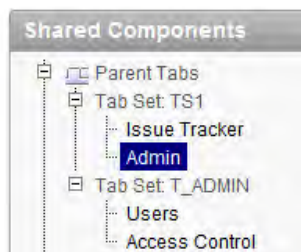


Figure 9-32. When editing page 600, the Admin tab will be in the first tab set group.

28. Under Authorization, set the authorization scheme to access control – administrator and click Apply Changes.

Now when running the application, if the user is not privileged with administrator access, the tab will not display. This avoids the event that would cause the user to see the access denied error message.

You have applied the authorization scheme at the page level and tab level for the administration pages. Next, you will remove the ability for a view-only user to create new records by associating the Edit authorization scheme with the button required to create tickets.

29. Edit page 200 of your application.
30. Edit the Create button by double-clicking its name.
31. Under Security, shown in Figure 9–33, set the authorization scheme to access control – edit and click Apply Changes.

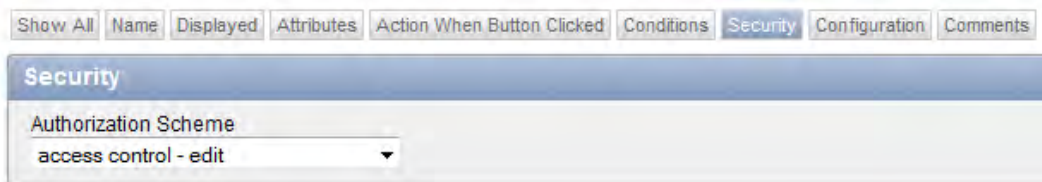


Figure 9–33. Security setting for the buttons

32. Repeat steps 30 and 31 for the Manage Multiple Tickets button.

To test this change, you will log in with the user name Martin. This user has been granted view privileges, so the buttons on page 200 aren't shown. Does this mean that Martin can't create tickets?

Let's review the steps you applied to the Admin pages. Security was first applied to the page itself, and then additional security was applied to prevent the access denied error. In the case of the buttons to create tickets, security to remove the buttons does not prevent the page from being run directly either from the developer tool or from changing the page number in the URL to page 210 or 230.

■ **Important** Removing or hiding a button, tab, or other link doesn't secure the target it was pointing at; it only helps reduce errors seen by users on components that are already secure.

The design for the help desk application only has the Multiple Tickets page only available to users with edit privileges, so the entire page will be secured at the edit level. The single record view of a ticket will continue to be visible to all authenticated users, but without the buttons related to record manipulation.

33. Edit page 210 of your application.
34. Edit the Create button in the Tickets region by double-clicking its name.
35. Under Security, set the authorization scheme to access control – edit and click Apply Changes.
36. Repeat steps 34 and 35 for the Delete and Save buttons as well as the second Create button located in the Ticket Details region.

37. Edit page 220 of your application.
38. Edit the Create button by double-clicking its name.
39. Under Authorization, set the authorization scheme to access control – edit and click Apply Changes.
40. Repeat steps 40 and 41 for the Delete and Save buttons.
41. Edit page 230 of your application.
42. Edit the page attributes by double-clicking on the page name.
43. In the Security region, set the authorization scheme to access control – edit and click Apply Changes.

Review the application now with different users. Notice how the user Martin can still navigate from the Tickets report to view the details of the ticket, but there are no buttons to modify the records in the database. Even though the form elements are editable, they are not written back to the database without the proper form submission.

Read-Only Items

Normally, users can edit the contents of an item in APEX. There are instances where you want to prohibit them from doing so, but you don't want to hide the item entirely. At the conclusion of the previous step, the user Martin does not have the ability to edit the ticket information even though the form looks like it can be changed. To assist in preventing changes, each item in APEX has a read-only attribute that you can programmatically set. The approach is similar to how item conditions are managed. Since the read-only attribute can't make use of an authorization scheme, you can use the APEX API `APEX_UTIL.PUBLIC_CHECK_AUTHORIZATION` to determine whether or not a user has the ability to edit the data. This API takes a parameter of the authorization scheme name and will run the verification returning a Boolean result that can be used in PL/SQL logic.

1. Navigate to and edit items indicated in Table 9–1 by double clicking on the name on the respective page.
2. Under the Read Only section, set the condition type to PL/SQL Function Body Returning a Boolean, as shown in Figure 9–34. Set the value for Expression 1 to the following:

```
RETURN NOT APEX_UTIL.PUBLIC_CHECK_AUTHORIZATION('access control - edit');
```

Table 9–1. Items That Require The Read-Only Attribute

Page Number	Page 210	Page 220
Items to Update	P210_SUBJECT	P220_DETAILS
	P210_DESCR	P220_ATTACHMENT
	P210_ASSIGNED_TO	P220_CREATED_BY
	P210_CREATED_BY	
	P210_CLOSED_ON	
	P210_STATUS_ID	
	P210_PUBLIC_FLAG	

■ **Note** I recommend that you use the Page Edit Selection button in the following steps for editing the read-only setting shown in Figure 9–34. The settings for Conditions and Read Only regions are close to each other and contain similar options. Using the selection buttons at the top of the page editor makes navigation much more efficient and prevents confusion with the Conditions region. Clicking on the > icon to advance to the next item is a shortcut for moving between successive items on a page. Your changes will be saved when using the arrows to switch items. Figure 9–34 shows the location in the upper right corner.

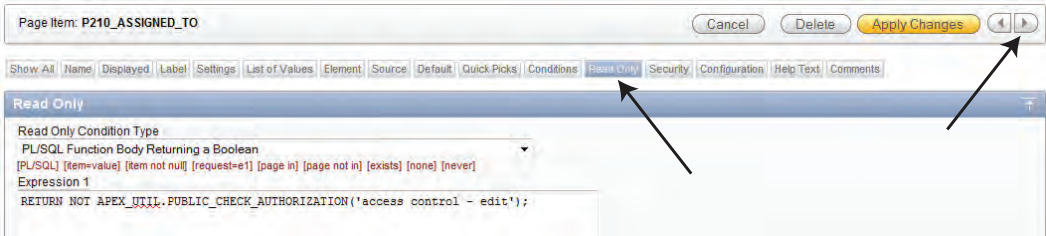


Figure 9–34. Editing the read-only condition to utilize the authorization scheme

When running the application as Martin, information about a ticket on page 210 shows data without the confusion of form elements. Authenticating as any other user will show the data in form elements and display the corresponding buttons. Results of the read-only view are shown in Figure 9–35; compare them to the form in edit mode shown in Figure 9–36.

Tickets

Subject PC will not turn on

*Description The user's PC will not turn on when the power button is pressed.

Assigned To Created On 07-AUG-2009 Created By - Select A User -

Closed On 03-NOV-2010

Status CLOSED Public Flag ☒ N ☐ Y

19 of 20

Figure 9–35. Ticket record in read-only mode

Tickets

Subject PC will not turn on

The user's PC will not turn on when the power button is pressed.

*Description

Assigned To - Select a Tech - Created On 07-AUG-2009 Created By - Select A User -

Closed On 03-NOV-2010

Status CLOSED Public Flag ☒ N ☐ Y

19 of 20

Figure 9–36. Ticket record in edit mode

Data Security

At this point, the majority of the application is relatively secure. What you don't have is data security applied to segregate the data between application users. Any authenticated user can see and make changes to any other user's records. APEX does not provide any built-in construct for securing data. APEX does support and work well with other Oracle technologies that secure data such as Virtual Private Database, Oracle Label Security, and Transparent Data Encryption.

While there are a number of ways to deal with data segregation and security, one of the simpler methods is to use a view to enforce the data available to a user in place of all references to the base table. This method is effective and works with all versions of the Oracle database. The process works by adding a securing function to the view that utilizes the current APEX user name filtering out the data from other users.

To implement the data security, you will run a script that will create a new view named `ticket_SECURE_V` and then re-create the other two views, `TICKET_ACTIVITY_V` and `TICKET_V`, so that they point to the secured view rather than the `TICKETS` table directly. Then you will make modifications to the other key components of the pages accessing ticket data to also use the new secure views.

1. Locate, upload, and run the script `ch9_data_security_script.sql`. Refer to Chapter 4 if step-by-step instructions are needed. You should see three rows, all of which complete successfully.

2. Once the script completes, run your application and navigate to the Analysis page. You should notice that only tickets or ticket details that are assigned to the user that you're logged in as appear.

Next, you'll make changes to the source of your reports and tabular form.

3. Edit page 200 of your application.
4. Edit the Tickets report by double-clicking on it.
5. Locate and open the file `ch9_report_p200.txt` and copy its contents into the Region Source, replacing everything that is there and click Apply Changes.
6. Run page 200 and notice that you can only see the tickets that are assigned to the current user.

You need to make a similar change on the Manage Multiple Tickets page.

7. Edit page 230 of your application.
8. Edit the Manage Multiple Tickets report by double-clicking on it.
9. Locate and open the file `ch9_report_p230.txt` and copy its contents into the Region Source, replacing everything that is there and click Apply Changes when complete.
10. Run page 230 and notice that you can only see the tickets that are assigned to the current user.

Next, you'll modify the Calendar report.

11. Edit page 400 of your application.
12. Right-click the Ticket Activity Calendar and click Edit Calendar.
13. Under the Tasks region on the right side of the page, click Convert to SQL Based calendar. Figure 9–37 shows the location of the link. Clicking the link will generate a conformation message and remain on the same page.

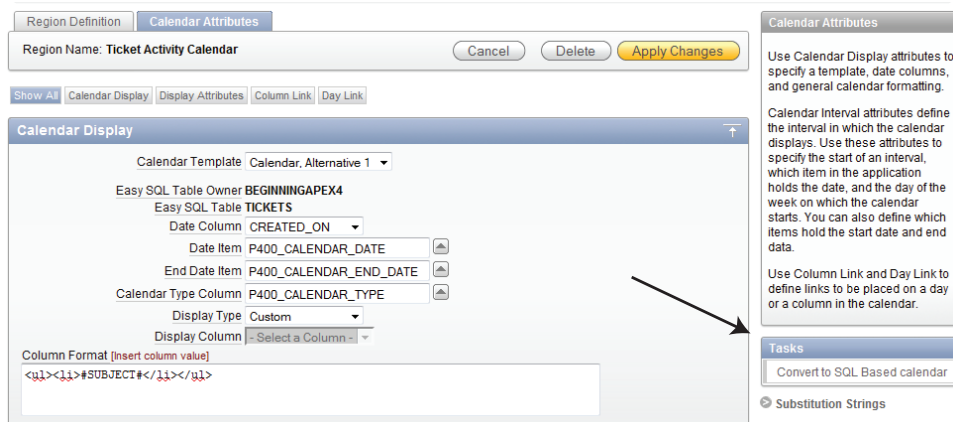


Figure 9–37. Tasks region on the right side of the Calendar Attributes tab

14. Click the Region Definition tab.
15. Locate and open the file `ch9_report_p400.txt` and copy its contents into the Region Source, replacing everything that is there. Click Apply Changes when complete.

Lastly, you should also apply this rule to the chart, as it is still allowing you to see the status from all records in the system, which is inaccurate.

16. Edit page 500 of your application.
17. Edit the chart attributes by right-clicking on the chart name and selecting Edit Chart from the context menu.
18. Edit Series 1 by clicking on the corresponding icon indicated in Figure 9–38.

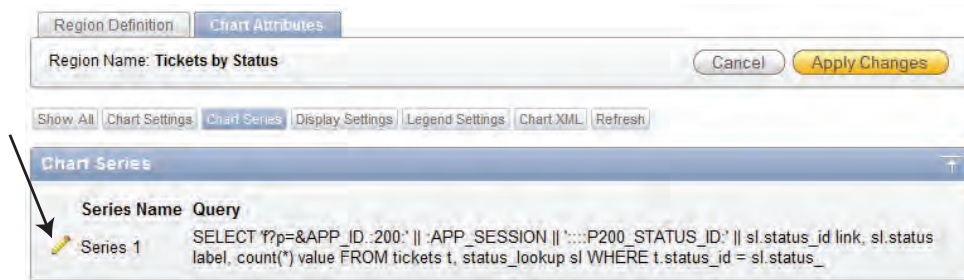


Figure 9–38. Each chart data series is edited independantly

19. Locate and open the file `ch9_report_p500.txt` and copy its contents into the SQL region, replacing everything that is there and click Apply Changes.
20. Run page 500 and notice that the chart only reflects the status of either unassigned tickets or tickets that are assigned the current user.

This is a huge leap forward in data security, but you're not quite finished yet. You may have noticed that if you edit one of your records on page 210, you can use the Next and Previous buttons in the upper right corner to see records that belong to other users. Thus, you need to plug this security hole as well.

21. Edit page 210 of your application.
22. The location of the process Get Next or Previous Primary Key Value is shown in Figure 9–39. Edit the process by double clicking the name.

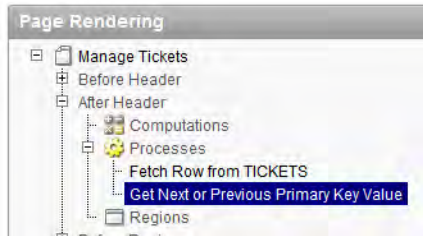


Figure 9–39. The location of the Get Next or Previous Primary Key Value process

23. Change the value of table name to TICKETS_SECURE_V as shown in Figure 9–40. Click Apply Changes when completed.

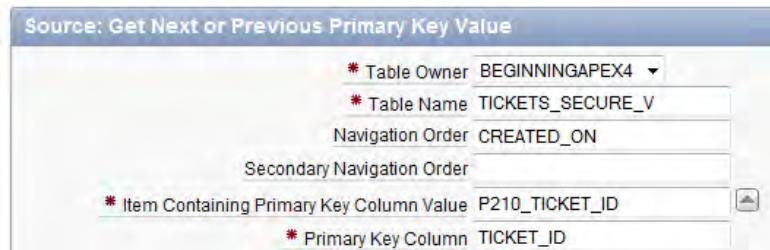


Figure 9–40. Update the source for fetching next record

Now all of your data is secured based on who is signed on to the system. Or is it?

Session State Protection

One of the most common ways to compromise a web application is through a form of attack known as URL tampering. You don't need to be a programmer or hacker to launch this type of attack—all you need to do is alter the URL in your browser. APEX introduced the session state protection feature in release 2.2. When enabled, it adds a checksum value to the URL. If any portion of the URL is altered, the resulting checksum will not match what is expected and the page will simply not render. Implementing session state protection is simple and recommended for any report based on sensitive data.

In the previous exercise, you secured the data from the report pages and a navigation component. However, you did nothing to protect pages 210 and 220 where the actual changes are made. Thus, if a user were to tamper with the APEX URL, it would be quite trivial for them to view and edit other user's tickets. This is easily visible from the page 200 report. The end of the URL link on the report clearly shows P210_TICKET_ID:10 where 10 in this case is the ticket number. Changing that number in the URL directly will cause APEX to fetch the new record ID automatically.

You can prevent this from happening in just a few easy steps, using APEX's session state protection functionality.

1. Go to the Shared Components region of your application.
2. In the Security region, click Session State Protection.
3. Click Page.

- Click on the link for page 210 – Manage Tickets.
- Set Page Access Protection to Arguments Must Have Checksum, set the Display Item Type to Data Entry Items, set the Item Session State Protection for P210_TICKET_ID to Checksum Required – Session Level and click Apply Changes. Figure 9–41 shows the session state protection settings for page 210.

Item Name	Region ▲	Sequence	Label	Item Session State Protection
P210_TICKET_ID	Tickets	10	Ticket Id	Checksum Required - Session Level ▼
P210_SUBJECT	Tickets	20	Subject	Unrestricted ▼
P210_DESCR	Tickets	30	Description	Unrestricted ▼
P210_ASSIGNED_TO	Tickets	40	Assigned To	Unrestricted ▼
P210_CREATED_ON	Tickets	50	Created On	Unrestricted ▼

Figure 9–41. Session state protection settings for page 210

Now run the Tickets report on page 200 in your application. Hover your mouse over the Edit icon and examine the URL. Notice the &cs= portion of the URL. The &cs= parameter is the checksum that was automatically generated by APEX. Alter the value for P210_TICKET_ID in the URL, or remove &cs= and everything to the right, and try to run the page. You will receive an error message similar to that shown in Figure 9–42.



Figure 9–42. Checksum error message as a result of URL tampering

Summary

In this chapter, you have applied new security to the help desk application by utilizing the key features of APEX. You implemented a new custom authentication scheme to allow control over users who access the sensitive parts of the application. You also reviewed conditional security with both authenticated and un-authenticated individuals and added parameters to allow the application to be used by both.



Application Deployment

This chapter will cover how to get an application out of the system in a form that can be easily migrated or stored and how to transfer that application back into the system using the same process. The concept of application migration and deployment is always something to consider when planning your application. In the case of APEX, it's very straightforward.

Exporting

An application export is simple to obtain. The interface includes a process designed to generate scripts for migrating APEX applications. As APEX is a set of metadata in the APEX workspace, creating a script that allows that metadata to be transferred is all that is needed. Let's make an export of the help desk application.

1. Export capability is located on the Application Builder main page shown in Figure 10-1 or on the application's Edit page. Both options navigate to the same location. You'll use the one on the Application Builder main page. Click the Export button.

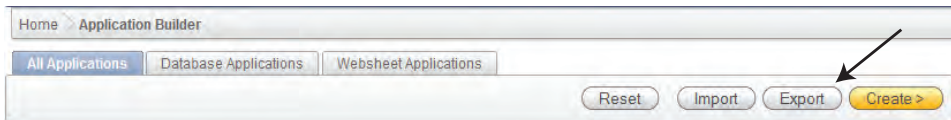


Figure 10-1. The export and import buttons are located on the main button bar inside an application

2. Select the Application tab or the Database Application link, as indicated in Figure 10-2. Select the help desk application and click the Set Application button.

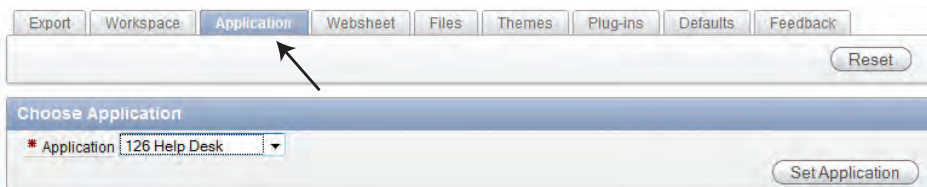


Figure 10-2. Any application in the workspace can be selected for export

3. A number of options will be available for the export of an application. The default settings shown in Figure 10–3 will work for the purpose of this exercise. After adjusting the settings, click the Export Application button at the top of the form.

The screenshot shows the Oracle APEX 'Export Application' dialog box. At the top, there are tabs: Export, Workspace, Application (selected), Worksheet, Files, Themes, Plug-ins, Defaults, and Feedback. Below the tabs are 'Reset' and 'Export Application' buttons. The 'Choose Application' section shows a dropdown menu with 'Application 126 Help Desk' and a 'Set Application' button. The 'Export Application' section displays the following information: Selected Application: Help Desk, Page Count: 15, Owner: BEGINNINGAPEX4, File Format: UNIX, Owner Override (empty), Build Status Override: Run and Build Application, Debugging: No, As of (empty) minutes ago (~ 5 min delay), and File Character Set: Unicode UTF-8. The 'Export Preferences' section shows five options: Export Supporting Object Definitions (Yes), Export Public Interactive Reports (Yes), Export Private Interactive Reports (No), Export Interactive Report Subscriptions (No), and Export Developer Comments (Yes). The 'Recent Exports to Database Repository' section shows 'no data found'.

Figure 10–3. Options for the export application

4. Depending on your browser and OS settings, you will be prompted to download and save the resulting file. Place the file somewhere easy to retrieve.

The downloaded file contains a large script that defines all of the contents of the application built in APEX. With the application pages are the shared components, including authentication schemes, authorization schemes, themes in the application, user interface settings, and reports. This script can, in turn, be imported into the same workspace, a different workspace, or even a different server.

It is important to realize that the export is the application only. The database objects, database permissions, static files, images, and cascading style sheets need to be migrated separately. You can create scripts for the database objects by using the SQL Workshop DDL generator tool.

Importing

APEX applications can be imported by simply providing the application export script. You can import into a different workspace or into the original workspace. Using the previous export file, you will re-import the same application into the original workspace.

1. From the Application Builder home page, click on the Import button.
2. When prompted to specify the file, as shown in Figure 10–4, click the Browse button and locate the application export file. The file type will be identified as database application. Click the Next button once you’ve specified the file.

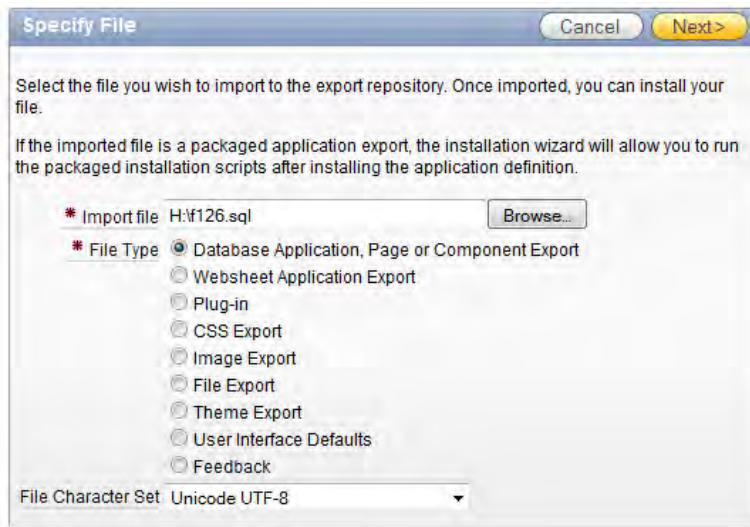


Figure 10–4. Import file identified as a database application

The message you receive in Figure 10–5 indicates that the file has been uploaded from your computer to the server. Remember that the application file is a script. At the moment, it has not been run; therefore the application is not installed.

3. To install the application, click the Next button.

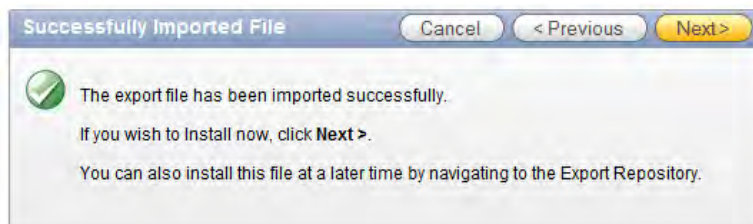


Figure 10–5. File upload success. Continue to install the application.

4. APEX prompts for a few key pieces of information, as shown in Figure 10–6. The parsing schema can be any of the database schemas available to the workspace. The build status allows the application to be set to a runtime mode, which is useful for production environments; the default allows run and build (or edit) mode. The final category concerns application ID values; the default option is to assign a new application ID when installed, which allows the same application to exist in the workspace multiple times—each time under a different ID. Click Install to continue.

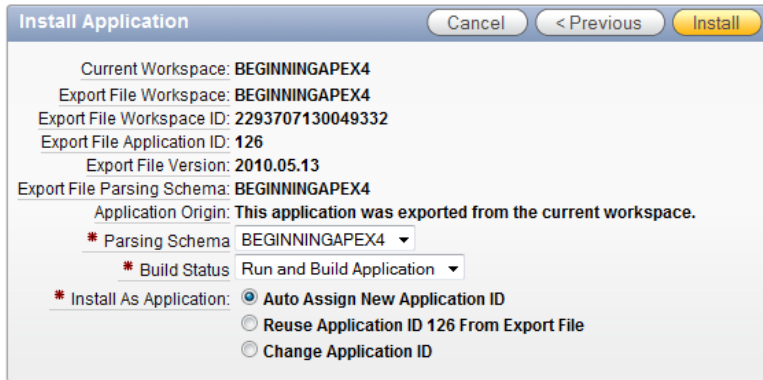


Figure 10–6. Installing the application into the workspace

5. After a short wait, you should see a screen like Figure 10–7 indicating that the application has been installed successfully. This completes the wizard.

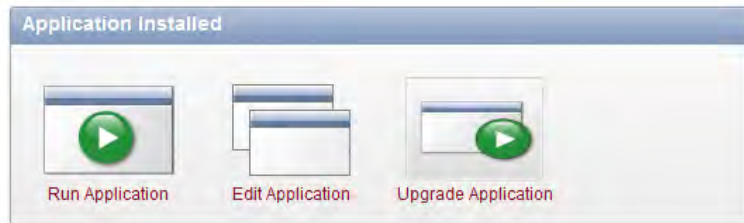


Figure 10–7. Successful installation of an application

At the end of the wizard, the third option may seem a little silly as the application came from the same version of APEX that you are currently running. However, the upgrade application option is handy when importing an application created in a previous version of APEX. In such cases, the upgrade application option can be a useful tool for quickly identifying areas that may benefit from the new technology in the current version. Don't worry if you bypass the upgrade application option; it is available at any time under the Utilities menu of the application editor.

Your results as seen in the application builder should be similar to those in Figure 10–8. You should notice two applications with the same title but different application values. The time of update is a good indication that application 127 is the newer version.

It is a good practice to distinguish applications with the same name within a workspace in some manner. Approaches include using group assignments, changing the name of the application to reflect the purpose, or utilizing the application version. The version and other attributes can be seen by utilizing an interactive report.


Application 	Name	Updated	Version
126	Help Desk	43 hours ago	release 1.0
127	Help Desk	16 seconds ago	release 1.1

Figure 10–8. Two applications with the same name but different ID values

Both applications will operate independently. Keep in mind that any data activity operates against the database user. Changes to data or database structures are common for both applications. For example, you can enter a help request from one application and view that same request from the other application.

■ **Note** Applications can also be duplicated using the copy application process under the Tasks menu of a specific application.

Supporting Objects

The method outlined during the export process is simply for exporting the application. It does not take into account the database objects. APEX does provide additional capability for installing database objects used by an application. Such objects are referred to as supporting objects. You can make use of a feature called Supporting Objects Scripting when you require a complete installation process that includes both application and database objects.

Figure 10–9 shows the main interface for the supporting object script definition. It consists of three major parts: installation, upgrade, and deinstallation. When an application is exported, there is an option to include the supporting objects. When the option is set, the export file will include the scripts and messages of the shared component and allow them to be used during import or removal of the application.

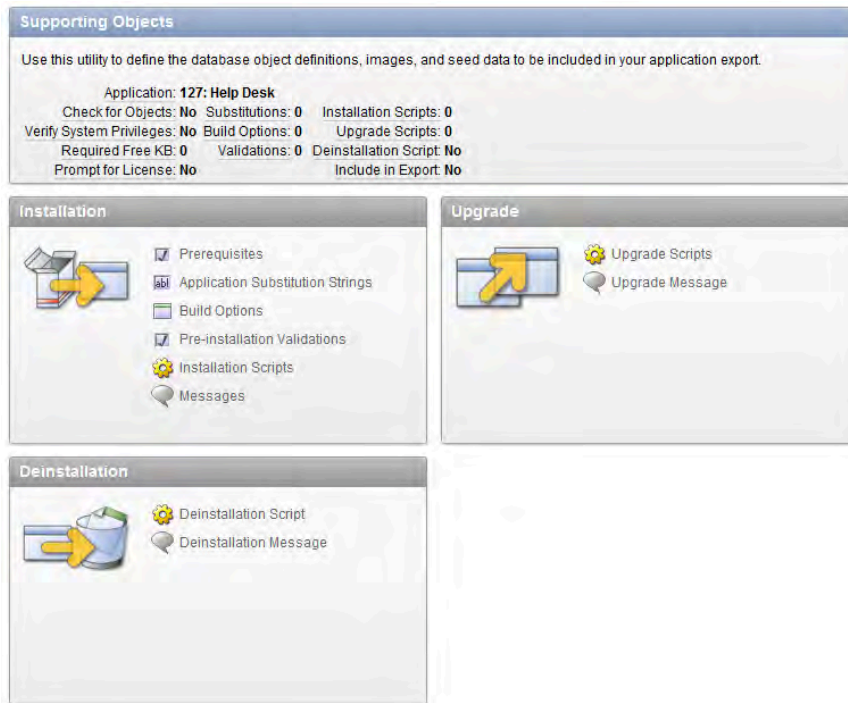


Figure 10-9. Supporting object options

After an application import, APEX will prompt the user to install the supporting objects, as shown in Figure 10-10. This is an optional step and can also be executed manually from the supporting objects screen in the application. When an application requires an upgrade, existing objects need to be examined to determine if a different set of upgrade scripts are needed.

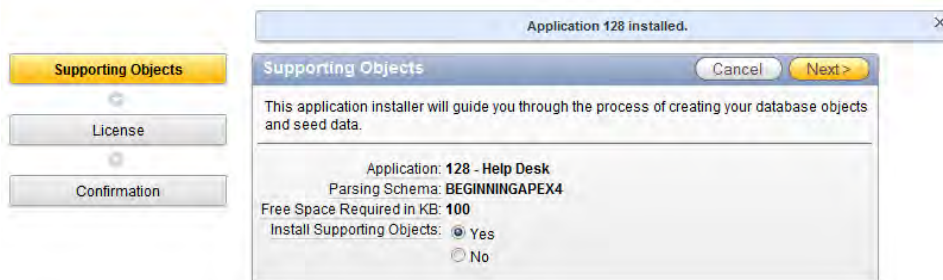


Figure 10-10. Supporting object installation after application import installation

Depending on the complexity of the application and the value of the existing data, it may be necessary to develop complex scripts. When an application has successive revisions, it may be necessary

to move or alter data structures for one version differently than from another version. This possibility needs to be part of the design consideration when developing installation scripts.

Deinstall scripts also require special consideration. When you ask to delete an application, APEX will also prompt you to deinstall supporting objects. When writing a script that will remove objects, it is a good practice to articulate the extent of that script, especially if it will result in the loss of data. The Supporting Objects Scripting feature allows you to construct messages that effectively communicate with the user.

■ **Note** If the intent is to design an application that will be distributed, it is important to create table names that are as unique as possible. For example, in your help desk application, the table USERS may already exist in some environments, resulting in a name conflict. This type of conflict needs to be considered when doing any object scripting.

Summary

APEX has built-in migration capability. The export and import capabilities are easy to use and very functional. The additional capability of constructing installation scripts to manage the database side of an application goes a long way toward the ability to deploy self-standing applications in one process.



Understanding Websheets

Websheets, a new marquee feature of APEX 4.0, deliver end user control over both web content and structure. In the early days of APEX, when it was still known as Project Marvel and later HTML DB, some people thought that end users could use APEX to develop their own applications. While this was true for simple spreadsheet-like applications, most end users were not comfortable building web applications that needed an underlying normalized database together with snippets of SQL, PL/SQL, and JavaScript. Websheets now fulfill the early promise of end user development for web content like blogs, wikis, and very simple business applications. Websheets give the end users this power without forcing them to learn how to normalize a database and write code. Everything in websheets, except a few optional advanced features, is declarative.

Websheets have been designed so that they are easy to use. However, like all computer tools, there is an associated learning curve. That's the bad news. The good news is that the learning curve is very shallow. The tool relies heavily on wizards that lead you intuitively through the content creation processes.

This chapter outlines the underlying structure of websheets, describes the navigation style, and highlights some of the handy features that will make you productive. This chapter concentrates on *what* websheets can do; Chapter 12 concentrates on *how* websheets are built by leading you through some step-by-step scenarios. After reading this chapter and working through the next chapter you will be able to quickly create professional looking web content.

■ **Note** As you read through this chapter, you may find yourself wondering how to create some of what is discussed. Not to worry. The examples in the next chapter provide an in-depth look at the major tasks involved in creating a worksheet. This chapter provides the background that will enable you to follow along with and fully understand the upcoming examples.

Websheet Structure

The fundamental building blocks of a worksheet (see Figure 11–1) are simple to envision. A worksheet is a container for web pages. The web pages, in turn, are containers for sections. A section, which is similar to a region in an APEX database application, contains your content. Annotations are used to enhance the content and the search functionality.

There are four section types.

Text: Text sections contain text that is easily formatted. Links to other content and images are embedded within the text by using very simple markup syntax.

Navigation: Navigation sections help you navigate through your hierarchy of pages. Creating these sections requires very little thought or effort on your part. You can also set up navigation within a long page by using section navigation.

Data: Data sections are used to display data in a row and column format that is similar to a spreadsheet. There are two types of data sections: report and data grid. A report is used to display read-only data from outside your websheet. Data grids are spreadsheet-like objects that you build. You are responsible for defining the columns, adding data entry business rules, providing default values, etc. If you have used spreadsheets, you will find this work relatively easy to do.

Chart: Chart sections are used to display graphs. Chart sections get their data from data sections. You link a chart section to a data section by using a simple intuitive wizard.

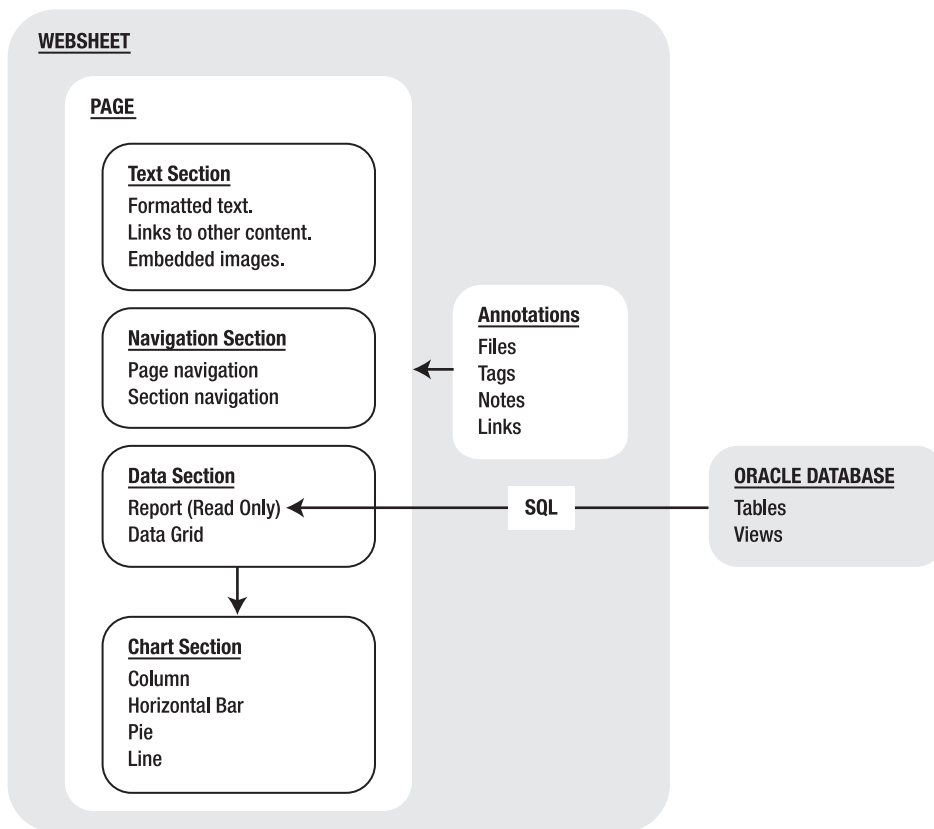


Figure 11–1. Websheet structure

Navigation

I will speak of websheet navigation in two contexts. First, I will discuss navigating through a websheet's content. Second, I will discuss navigating through the pages that are used to build a websheet. Be mindful that, in practice, you will frequently flip back and forth between these two contexts.

In both contexts, there are usually several ways to navigate to a given page or section. The duplicate navigation choices might cause you a bit of confusion at first. However, after you work with websheets for a while you will find that the navigation choices are helpful and intuitive. This chapter does not document every possible navigation path; instead, it shows you where to look for navigation links so that you can quickly find a comfortable navigation style that works for you.

Content Navigation

Content navigation enables you to quickly go to pages and sections within a page. Page navigation is mandatory. Section navigation is optional and is useful on long pages that require a lot of vertical scrolling.

Page navigation is created in two ways. First, the websheet itself automatically creates page navigation by adding breadcrumbs and right hand side sections that contain links to child and peer pages (see Figure 11.2). For small websheets, the breadcrumbs and the right side navigation sections might be all you need.

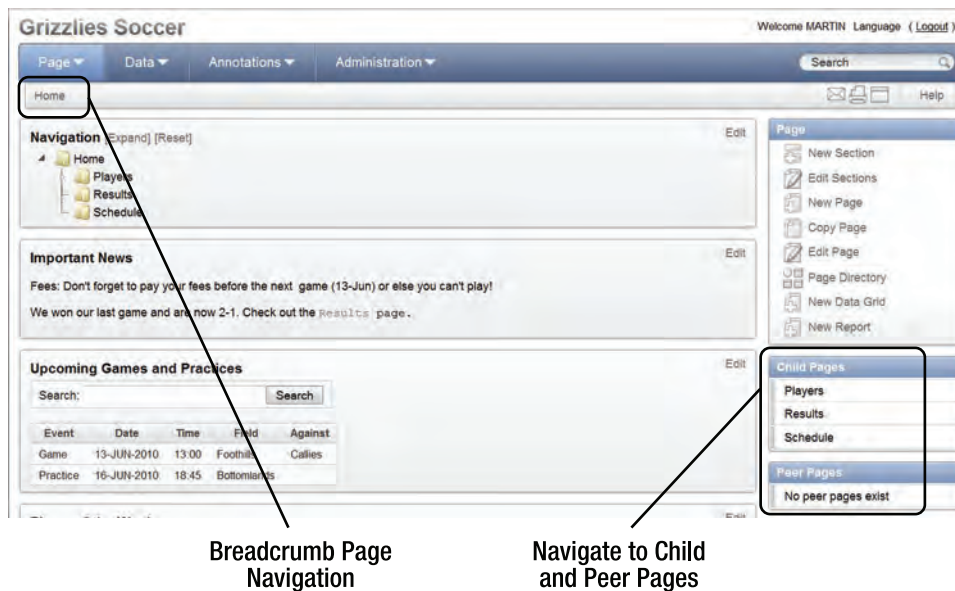


Figure 11–2. Page navigation created by the websheet

Second, you can add page navigation manually by creating a page navigation section. You can also embed explicit page links in the page content (see Figure 11–3). The details for adding a page navigation section is discussed later under the heading "Navigation Sections." Embedded links are discussed in detail in the "Markup Syntax" section.

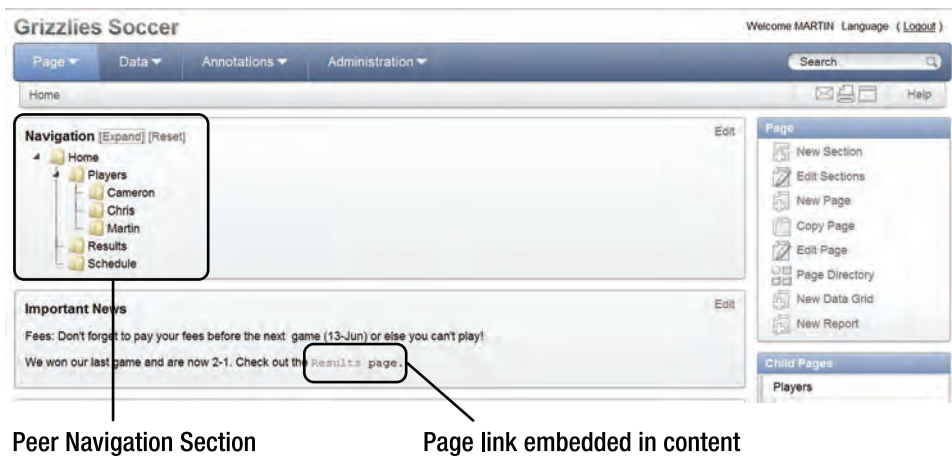


Figure 11–3. Page navigation created by the user

Section navigation is optional. It's good for content-heavy pages that require a great deal of scrolling to reach the bottom of the page. Section navigation is almost identical to page navigation (see Figure 11–4). The main difference between page and section navigation sections is the lack of hierarchy in section navigation; sections have no children.



Figure 11–4. Section navigation created by the user

Structural Navigation

Structural navigation is used to access the pages that are used to build and update the structure and content of websheets. On most websheet pages, there are two areas that enable you to access the structural pages (see Figure 11–5). The first area contains the drop-down menus at the top of the pages. These menus do not change from page to page. The second area is located on the right side of every page. This area contains a set of sections that, in turn, contain links to the various structural pages. These sections and links vary from page to page and are tailored to the page's context.

In addition to these areas, some structural links are embedded in the content sections. These embedded links are convenient for getting to the Edit page for the content that is currently being displayed.



Figure 11–5. Structural navigation

Help

Do not overlook the Help link (see Figure 11–6). The help is clear, concise, and useful. Clicking the Help link invokes a pop-up page that contains mostly static information that you can read at your leisure. I strongly recommend that you do so; it will take only a few minutes of your time.

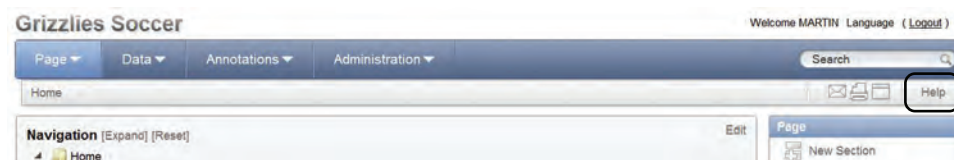


Figure 11–6. Help link

In addition to the static information, there is one tab that contains dynamic information. The Application Content tab contains complete lists of all the websheet's pages, sections, files, images, data

grids, and reports (see Figure 11–7). These lists are presented in interactive reports that you can tailor to suit your needs.

All of the interactive reports contain a column that displays the explicit markup syntax that enables you to embed links to the listed objects directly in your content. This saves you the effort of having to remember the details of the markup syntax and typing it manually; you also avoid the aggravation of debugging typos.

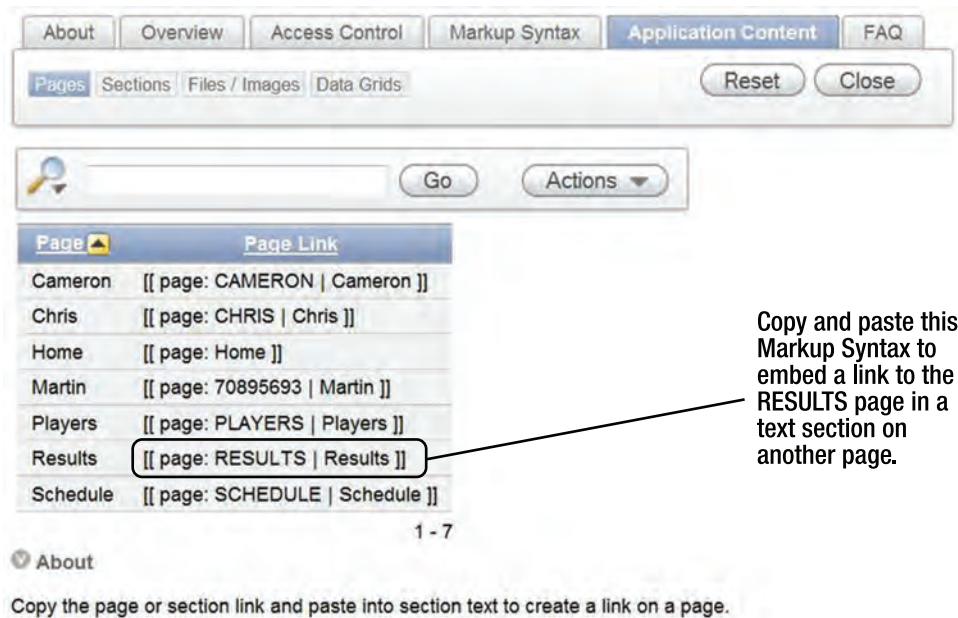


Figure 11–7. Help - Application Content tab

An example of how the markup syntax is used is shown in Figure 11–8. A link to the Results page is embedded in the content in the Important News text section. Pressing the Edit link in the Important News text section takes you to the corresponding Edit page (see Figure 11–9) where the underlying markup syntax for this example is illustrated.

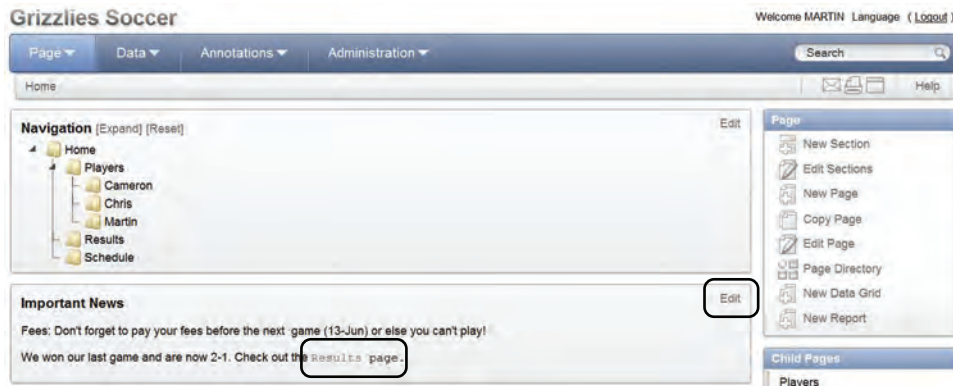


Figure 11-8. Resulting content of markup syntax

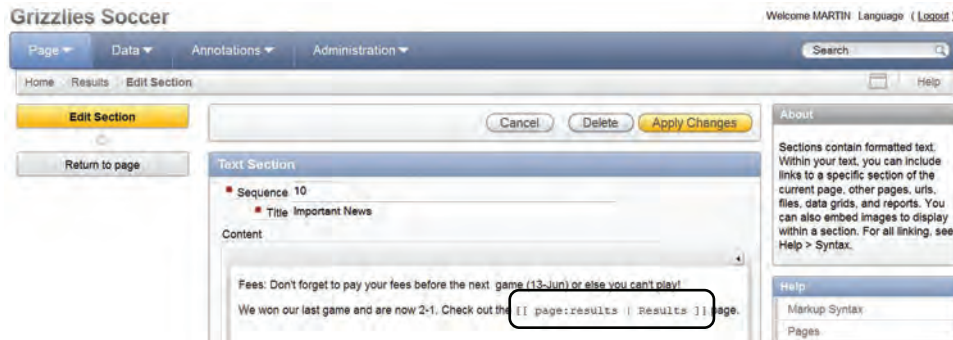


Figure 11-9. Embedded markup syntax

Markup Syntax

The markup syntax that is used to embed links in your web content looks a bit like a computer language. End users might find the syntax a bit intimidating; however, the syntax structure is simple, forgiving, and well documented on the Help page.

The markup syntax is simple:

```
[[ LINK_TYPE: LINK_TARGET | LINK_NAME ]]
```

The opening and closing delimiters are two square brackets that are easy to read. The LINK_TYPE is a keyword with a trailing colon. The available LINK_TYPES are described in Table 11-1.

Table 11–1. LINK-TYPEs and Descriptions

page:	Links to a page in the websheet.
section:	Links to a section in the websheet
url: and popupurl:	Links to a URL
file:	Downloads the target file to the user's computer.
image:	Displays an image on the page in a text section.
data grid: or datagrid	Links to a data grid's Edit page.
report:	Links to a read-only report page.
sql:	Displays the result of a SQL statement in a grid.
sqlvalue:	Displays a single value from a SQL statement.

The `LINK_TARGET` specifies the object that is displayed when you click on the link. For a page link, the `LINK_TARGET` is the page alias. For a file, the `LINK_TARGET` is the file name or alias. The only exception to this pattern is the `sql: LINK_TYPE`. The `sql: LINK_TYPE`'s `LINK_TARGET` is not a link; it's a SQL statement that returns data in rows and columns. The SQL data is automatically displayed when the page is displayed. The `sql:` syntax is also referred to as SQL tags and this feature must be turned on by an administrator in the application properties area. This is covered later in the "Reports - Setup" section.

A vertical bar character separates the `LINK_TARGET` and the `LINK_NAME`. The only fussy part of the syntax is the single spaces that must precede and follow the vertical bar.

The `LINK_NAME` contains the text that is embedded in the page's content. The user clicks on this text to follow the link. There are two exceptions to this pattern. First, the image `LINK_NAME` is optional and can be replaced by HTML markup. For example, the HTML markup can be used to resize the image. Second, the `sql LINK_TYPE` has no `LINK_NAME` at all. The `LINK_NAME` is not required because the SQL data itself is automatically embedded in the page content.

The markup syntax is forgiving. It is case insensitive and the websheet code makes several friendly assumptions. For example, if you omit the `LINK_TYPE`, the websheet scans its metadata for the `LINK_TARGET`. If an exact match is found, the websheet assumes that this is the target for which you were looking. In other words, you can be a bit sloppy with the syntax and still get the correct result.

User Authentication

User authentication governs how users log on to a websheet. There are four options.

Application Express Account: The websheet users log on to the websheet by using the IDs and passwords that have been set up in the APEX workspace that hosts the websheet.

Single Sign-On: Oracle's Single Sign-On technology enables users to sign into their computing environment one time and then access all of their applications, like websheets, without having to re-enter their user name and password. This is an advanced feature that is out of scope for a beginning book.

LDAP: Lightweight Directory Access Protocol (LDAP) is used by websheets to provide single sign-on capability in computer environments that use non-Oracle authentication schemes. This is an advanced feature that is out of scope for a beginning book.

Custom: This advanced feature is explicitly explained and illustrated in the Chapter 12.

You set up user authentication by first navigating to the Websheet Properties page and selecting the Authentication tab. Here, the current authentication option is displayed in a read-only section that also contains an Edit Authentication button (see Figure 11–10).

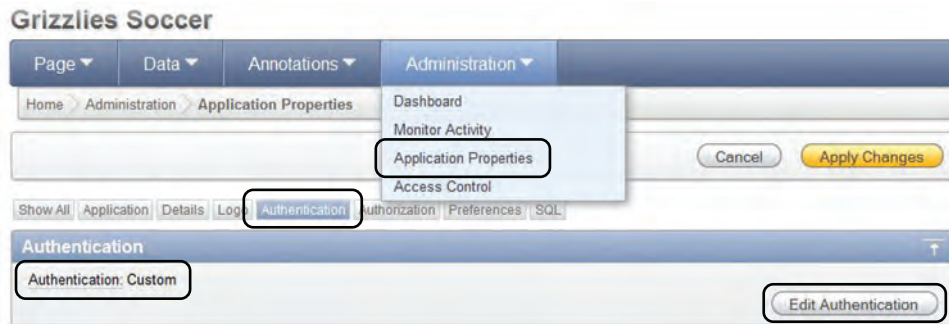


Figure 11–10. Read-only user authentication option displayed in a websheet

When you press the Edit Authentication button, you are redirected to the websheet's Application Properties page in the Application Builder (see Figure 11–11).

When you are using the Application Express Account authentication option, pressing the Edit Authentication button takes you out of the websheet and directly into the Application Builder. This transition may not be obvious to you at first because the page bodies are similar. You need to verify your context by looking at the top of the page and the menus.

When you use a Single Sign-On, LDAP, or custom authentication option to get into your websheet, pressing the Edit Authentication button redirects you to the login page for the Application Builder. Here you must log in as an APEX administrator or developer. After you log in, you are taken directly to the websheet's Application Properties page where you pick the desired authentication scheme and configure it. The details of the example shown in Figure 11–11 are discussed in Chapter 12.

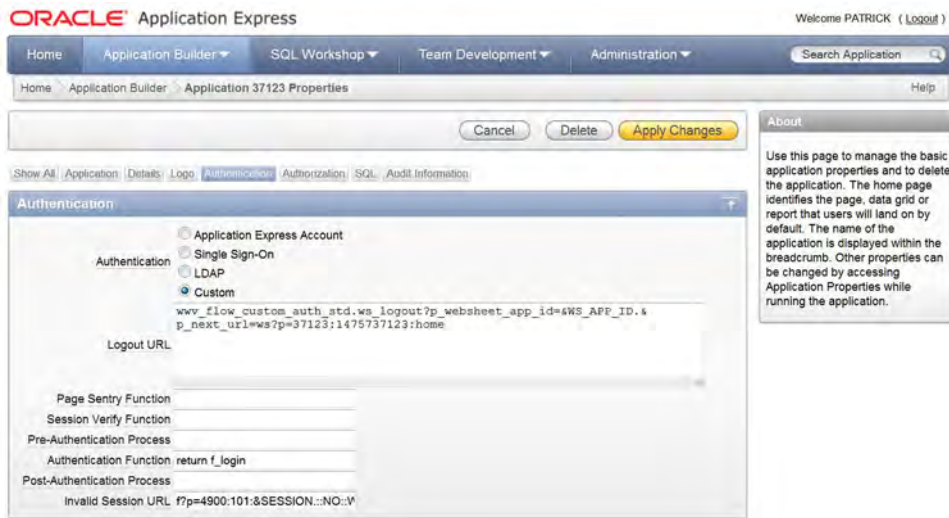


Figure 11–11. User authentication option displayed in the Application Builder

User Authorization

Websheets have three authorization roles.

Reader: This is the read-only role. Figure 11–12 is a websheet home page as seen by a reader. The page contains content together with navigation objects. When you drill down into data pages, you see the data but not the buttons that are used to add, change, and delete data.

Contributor: This role is allowed to add, change, and delete a websheet's content plus manipulate the structure. Figure 11–13 is the websheet home page as seen by a contributor. The figure highlights the rich set of functionality that is added for this role. The top drop-down menus contain links to the structural pages. The text sections contain an Edit link. The right side sections contain links to the structural pages. When you drill down into the data pages, you see the buttons that allow you to add, change, and delete the content.

Administrator: This role creates and deletes websheets. It is responsible for maintaining a websheet's global properties and maintains the list of users who can access the websheet. Figure 11–14 is the administrator's view of the websheet home page. The only addition to the contributor's view is the Administration drop-down menu at the top of the page.

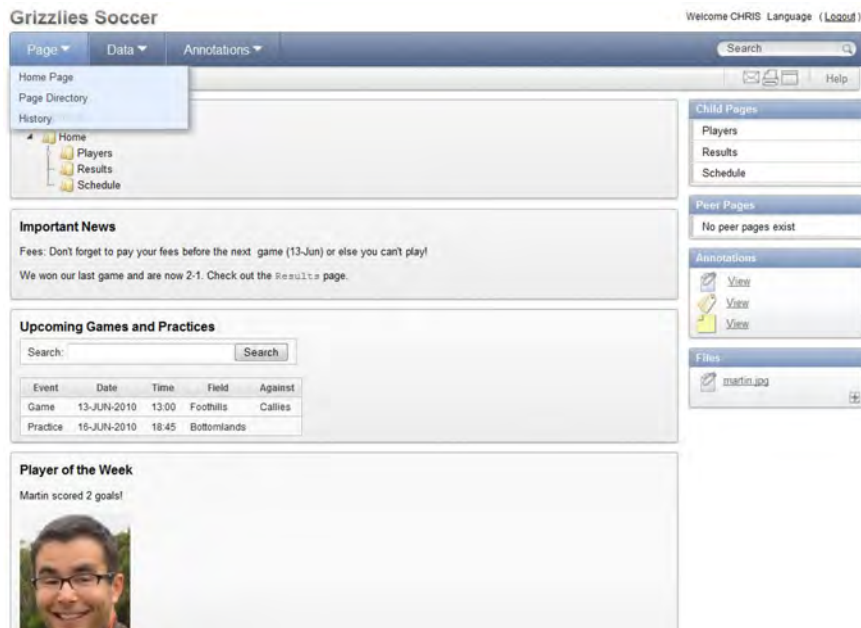


Figure 11–12. Websheet home page as seen by a reader

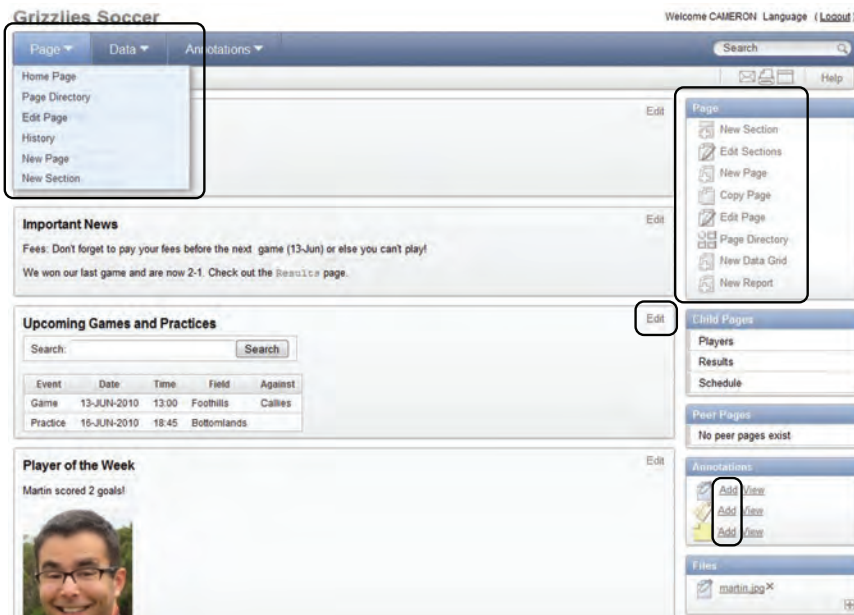


Figure 11–13. Websheet home page as seen by a contributor

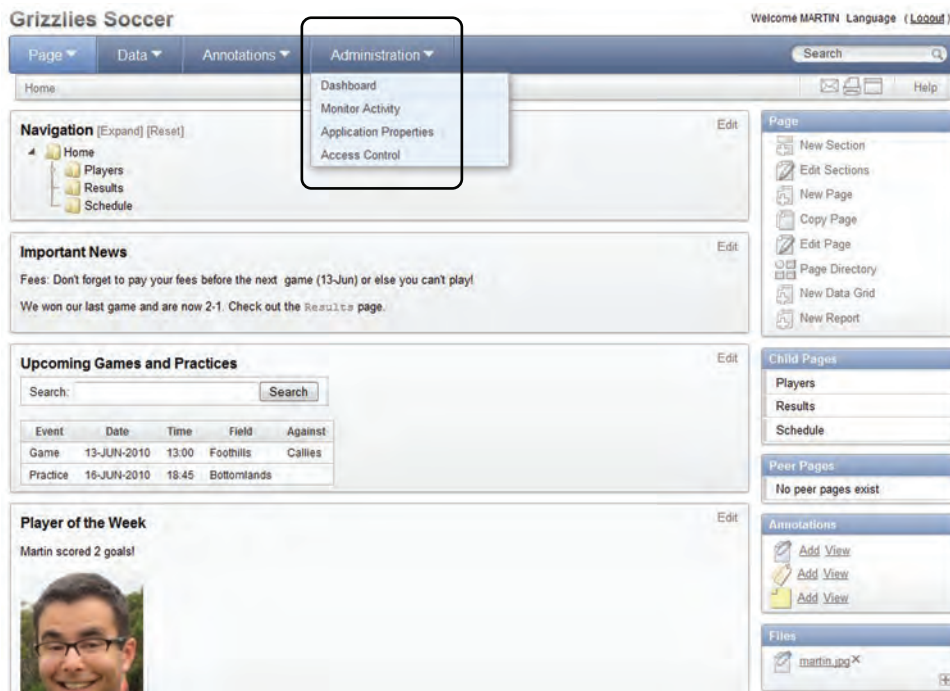


Figure 11–14. Websheet home page as seen by an administrator

The websheet administrator configures the user privileges through the Access Control list, which is found under the Administration drop-down menu (see Figure 11–15). This task is usually done after you set up the authentication scheme.

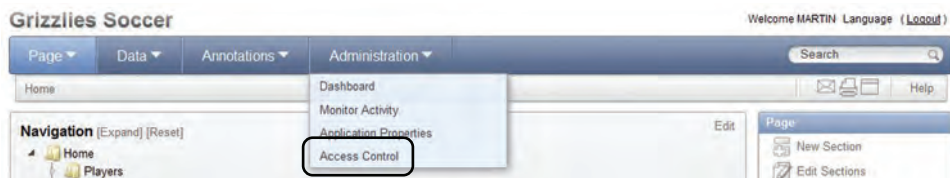


Figure 11–15. Navigating to the Access Control list

The Access Control list is simple to create and maintain (see Figure 11–16). Create a new entry by pressing the Create Entry button. Change an existing entry by clicking on the pencil icon in the list. In both cases, you are taken to the Entry Details page that has only two fields: the user name and the privilege level (see Figure 11–17).

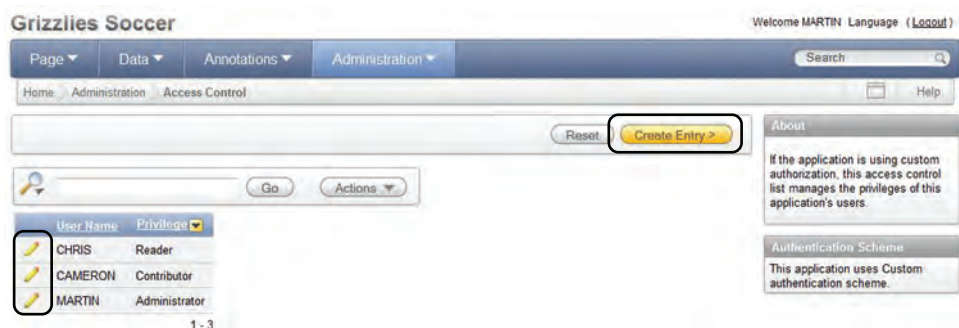


Figure 11–16. Access Control list



Figure 11–17. Entry Detail page

The Access Control list is somewhat sensitive to the authentication scheme that is used. When you use Single Sign-On, LDAP, or a custom authentication scheme, the Access Control list is mandatory. In this context, it's easy to understand and build. You build the Access Control list as a duplicate of the list of users in the authentication scheme. The hook between the two lists is the user name. The websheet user name must match the user ID in the authentication scheme.

When you use the Application Express Account authentication scheme, things get a little bit harder to understand. In this instance, using the Access Control list is optional. Since the websheet is inside an Application Express workspace, the websheet can directly use the existing APEX user accounts. The websheet privileges are inferred from the APEX user account privileges. Table 11–2 illustrates the translation between the APEX workspace privileges and the websheet privileges. You are able to override the default translation by adding the APEX users to the Access Control list. For example, you might want an APEX workspace administrator to have the reader privilege on a given websheet. To do this, you add the APEX workspace administrator's ID to the Access Control list and set the privilege to reader.

Table 11-2. Access Control Configuration

Authentication Scheme	No Access Control List	With an Access Control List
Application Express Account	APEX Administrator = Websheet Admin APEX Developer = Websheet Contributor APEX End User = Websheet Reader	The Access Control list overrides the inferred APEX websheet privileges
Single Sign-On	NA - Access Control list is mandatory	Access Control ID must match SSO ID
LDAP	NA - Access Control list is mandatory	Access Control ID must match LDAP ID
Custom	NA - Access Control list is mandatory	Access Control ID must match Custom ID

Websheets can be set up for public access. This means that anyone who invokes the websheet's URL is allowed to access the websheet with the reader role. Logging in with an ID and password is not required. You set this up by going to the Application Properties page and pressing the Edit Authentication button in the Authentication section (see Figure 11-18).

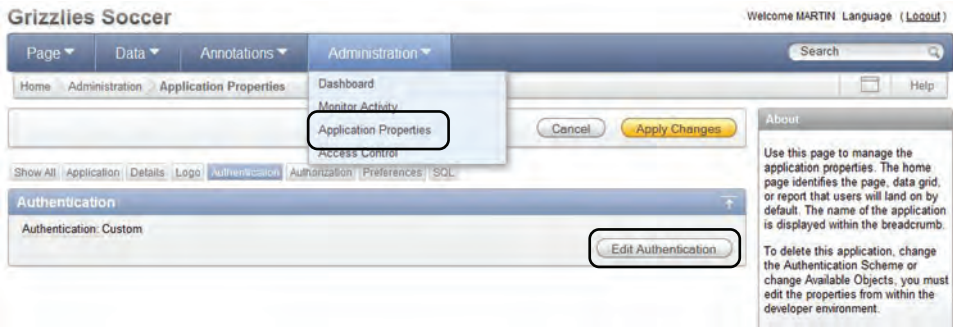


Figure 11-18. Navigating to the Application Properties page

Once you are logged into the Application Builder as a workspace administrator or developer, you navigate to the Authorization region on the Application Properties page. You then select Yes in the Allow Public Access drop-down menu and press the Apply Changes button (see Figure 11-19).

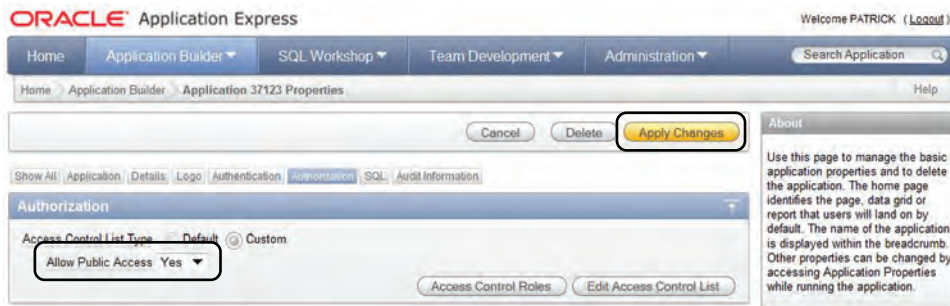


Figure 11–19. Setting up a websheet for public access

Now when you run your websheet application, you are automatically logged in as the user “nobody” with reader privileges. Administrators and contributors who need to update the websheet's content can log in by using the Login link that now appears in the upper right of all the websheet's pages (see Figure 11–20).



Figure 11–20. Public websheet

Sections

Sections contain your content. The following chapter sections illustrate useful features in the websheet structural environment. The features are illustrated by showing you the Edit pages for existing objects. Step-by-step procedures for creating new websheet objects are covered in Chapter 12.

Text Sections

Text sections contain text, embedded links, and images. Text sections can be used to create wikis and blogs. To start a wiki, the original author creates a text section and then invites contributors to edit the text section's content. To start a blog, the original author creates a text section and then invites contributors to add more text sections in reply to the first section.

To access a text section's Edit page, you click on the Edit link in the upper right corner of the text section (see Figure 11–21).



Figure 11–21. Navigating to a text section's Edit page

The Edit page for a text section is simple and clean when it is invoked. By default, the collapsible regions are collapsed (see Figure 11–22). The upper right collapsible region link, a small arrow icon, contains the text formatting controls. The lower left link, Page Sections, contains a list of all the sections on the page. The lower right link, a small triangle icon, enables you to drag the corner of the text area to enlarge it. This is a handy feature when you have a large amount of content.

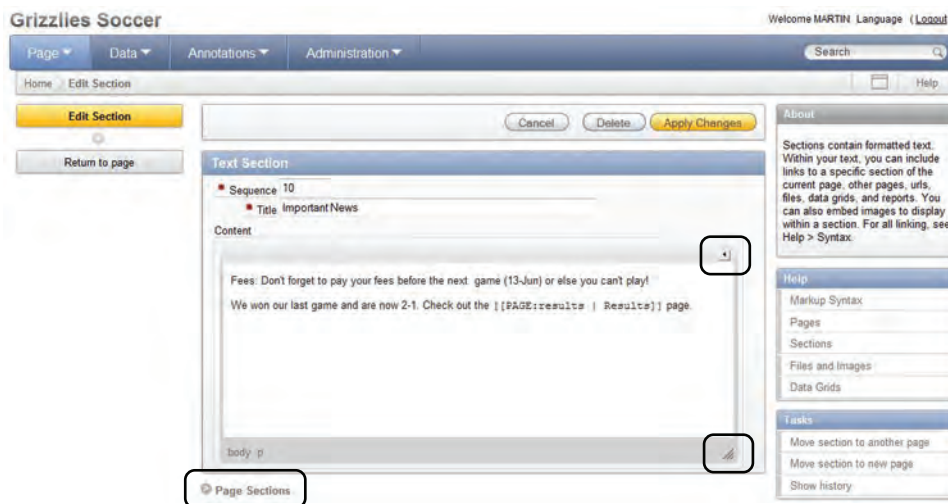


Figure 11–22. Edit page for a text section - collapsed

When you expand the collapsible regions, you can see the considerable scope available for enhancing your content (see Figure 11–23). The upper right icon expands into a section that contains a number of formatting icons. I am not going to describe each formatting icon's function in detail. I feel that you will find them intuitive because they are similar to other tools, like a word processor, that you probably use regularly.

Other elements that you'll see include:

- The list of page sections at the bottom of the page shows you where the current section fits within the page relative to the other sections.
- The Help section contains direct links to the Help page tabs.
- The Tasks section contains links to processes that automate moving the section to an existing or a new page.

The Show History link is explained in the "Administration" section later in this chapter.

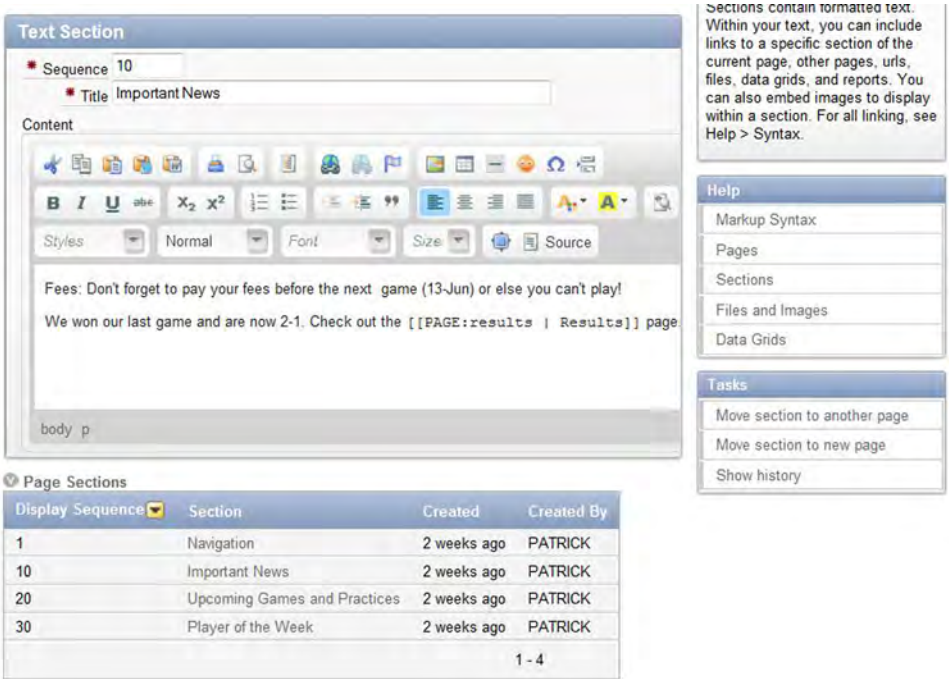


Figure 11-23. Edit page for a text section - expanded

Note Many pages contain collapsible regions at the bottom of the page. Some of the regions contain help text; others contain lists of things that help give you a sense of place and context which, in turn, helps you to get your content right. In all cases, the collapsible regions that are found at the bottom of pages are useful.

Navigation Sections

The most important aspect of adding navigation sections to your web content is the fact that adding them takes very little effort or thought on your part. Navigation sections are easy to use because they are declarative and the websheet takes care of virtually all the details like the page links and formatting. The details of adding navigation sections are discussed in Chapter 12.

A page navigation section is illustrated in Figure 11–24. Pressing the Edit link takes you to the Edit page (see Figure 11–25). The Edit page has four inputs.

Sequence: Positions the section amongst the other sections on the page.

Title: The title of the section.

Starting Page: The starting page field enables you to start the navigation tree on pages that are below the home page in the page hierarchy. In this example, Cameron could add a page navigation section to the top of his personal page that shows only the pages below his personal page in the hierarchy.

Maximum Levels: The maximum levels field is used to limit the number levels that are displayed in a page navigation section. In this example, if you set the maximum levels value to three, you would see only the pages that are illustrated in Figure 11–24 even if Cameron added child pages under his personal page.

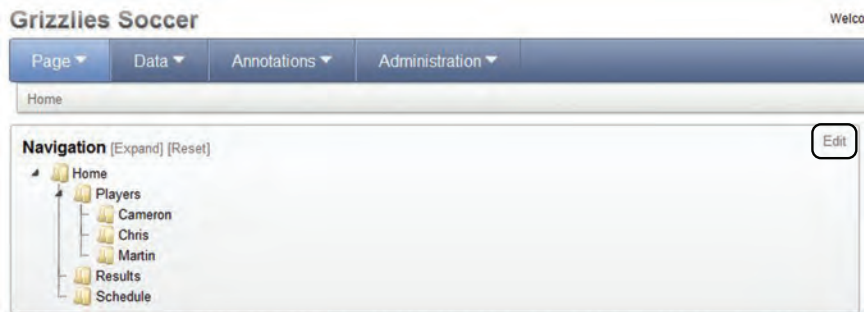


Figure 11–24. Page navigation section

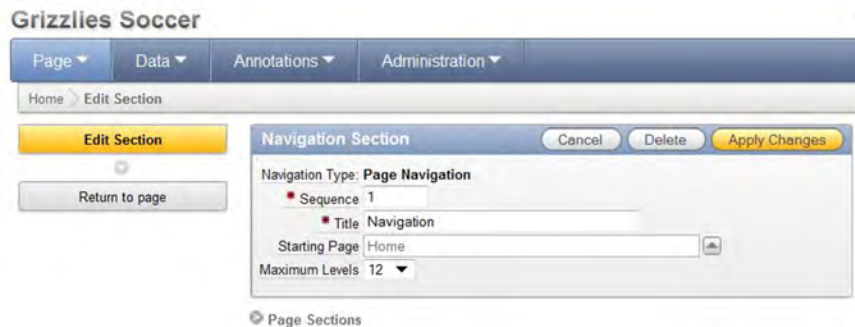


Figure 11–25. Edit page for a page navigation section

A section navigation section is illustrated in Figure 11–26 together with the Edit page in Figure 11–27. The only inputs are the sequence and title. In this case, the websheet takes care of all the other details. You don't have to do any work.

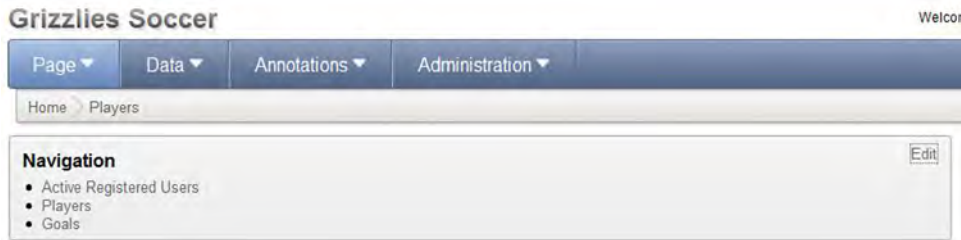


Figure 11–26. Section navigation section



Figure 11–27. Edit page for a section navigation section

Data Sections

There are two types of data sections: reports and data grids. Reports display read-only data from external database tables or views that are located outside the websheet. Data grids are spreadsheet-like objects that you create entirely within a websheet.

Reports - Setup

The Report feature and the related SQL Tags feature require a small amount of setup by the administrator before the contributors can use them. You start the setup process by navigating to the SQL section on the administrator's Application Properties page (see Figure 11–28). Pressing the Edit SQL Access and Suggested Objects button takes you to the Application Builder (see Figure 11–29) where you do the actual configuration.

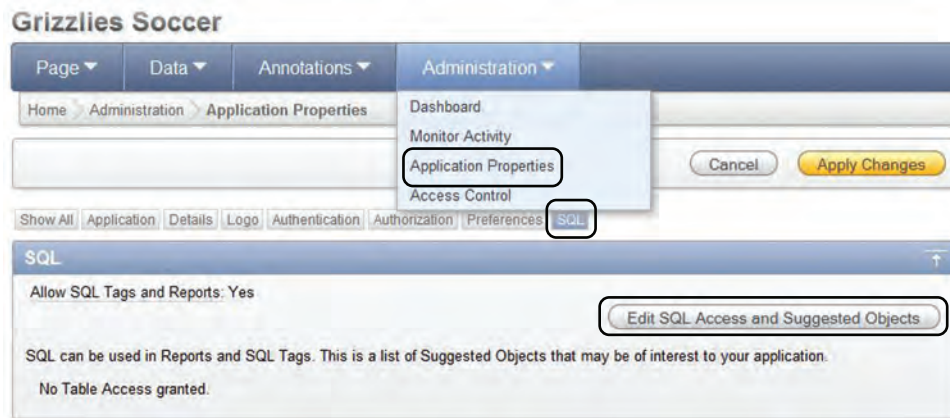


Figure 11–28. Websheet report and SQL Tag setup

In order for your websheet to see all of the database objects in the default schema, you set the Allow SQL Tags and Reports to Yes. After you do this, you press the Add Object button that takes you to the page illustrated in Figure 11–30. This page enables you to create a list of suggested objects. The list of suggested objects is an optional convenience that automatically creates a drop-down list of database objects together with helpful comments.

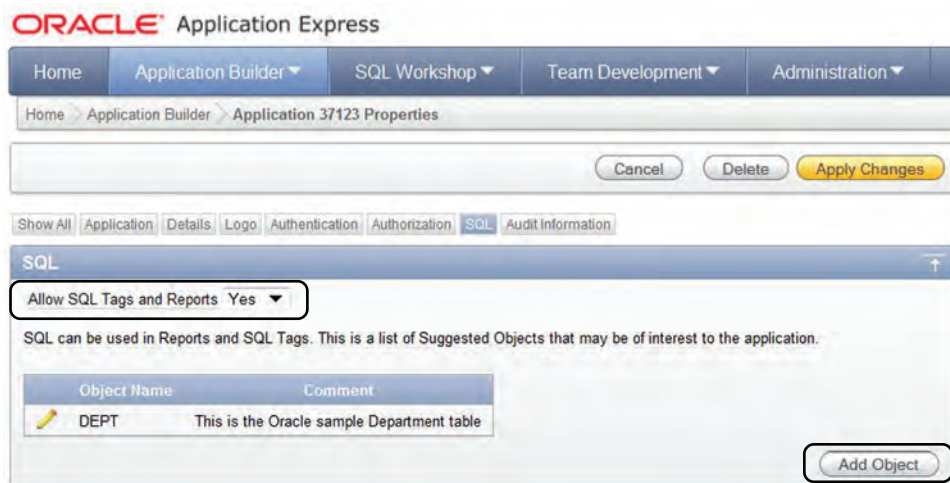


Figure 11–29. Application Builder report and SQL Tag setup

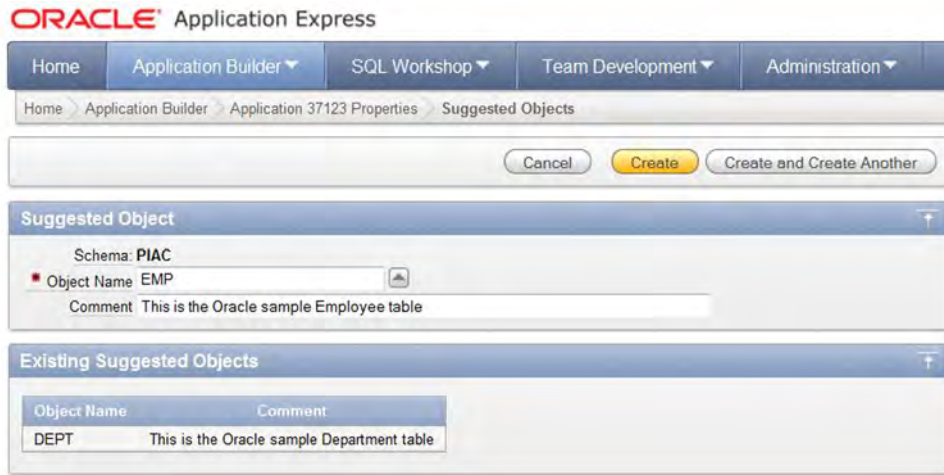


Figure 11–30. Create the list of suggested database objects

■ **Note** When you set Allow SQL Tags and Reports to Yes, you are giving the contributors access to all of the database objects in the websheet's default schema. This is a potentially serious security issue. It is imperative that you chat with your Oracle database administrator (DBA) before you use this feature to make sure that sensitive data is not accidentally exposed.

Reports - Creation

After you have set up the Allow SQL Tags and Reports feature in the Application Builder, you return to your websheet. You create a report by navigating to the Data Report page (see Figure 11–31) and pressing the Create Report button.

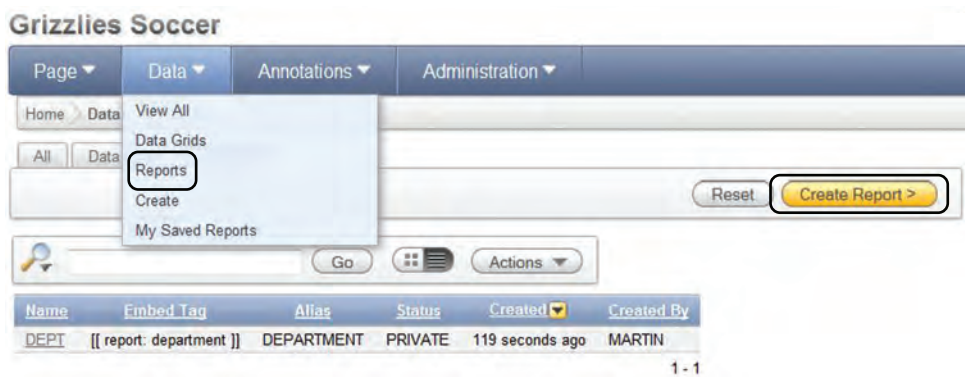


Figure 11–31. Navigate to the Data Reports page

The Create Report page prompts you for one of two report sources (see Figure 11–32). The Table Report Source creates a report that contains all of the columns in the selected table or view. The SQL Query Report Source (see Figure 11–33) creates a report based on a SQL statement. Using a SQL statement gives you a tremendous amount of flexibility in tailoring a report to your needs. In both cases, you press the Next button, which takes you to a confirmation page so you can check your input before creating the report.

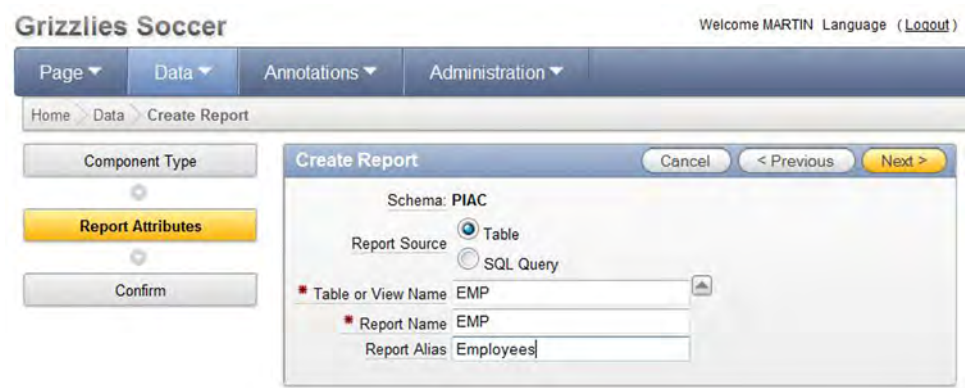


Figure 11–32. Create a report based on a table or view

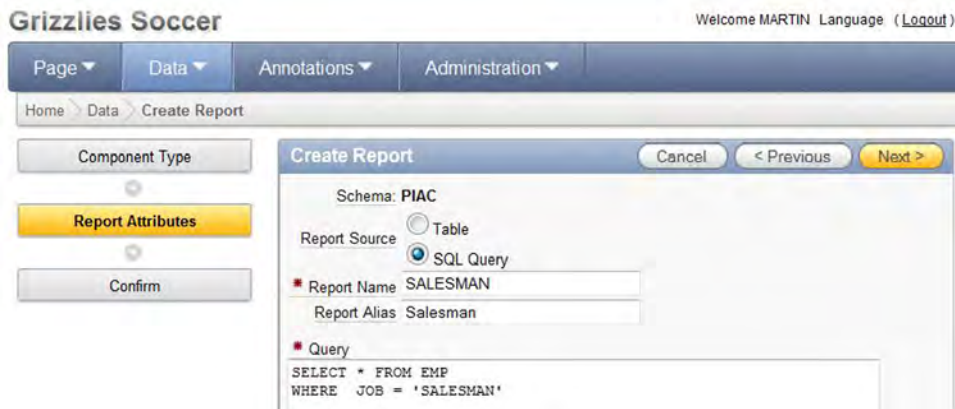


Figure 11–33. Create a report based on a SQL statement

Reports - Accessing the Data

Users will want, of course, to see the report data. Report data is exposed in three ways.

Navigate to a report: Figure 11–34 shows you where to find the list of your reports under the top drop-down menu. Clicking on the Name link in the list takes you to the Report Data page. Figure 11–35 illustrates the Salesman report in this example.

Embed a link to the report in a text section: The list of reports in Figure 11–34 contains an Embed Tag column. This column contains the markup syntax that you need to add a link to the report inside a text section. You copy and paste the markup syntax into the text section to add a link that takes you to the Salesman report that is shown in Figure 11–35.

Create a data section: Creating a data section that is based on a report is easy. A wizard walks you through the steps. You first navigate to the page that will contain your report and click on one of the New Section links (see Figure 11–36). This starts the Create Section wizard. Select the Data icon on the first page and press the Next button (see Figure 11–37). You now link the data section to its data source (see Figure 11–38). In this example, you are linking the data section to the Salesman report. This page allows you to link your data section to any data grid or report that you have previously created. Press the Next button to continue. The next page contains a single drop-down list called Saved Report Settings to Use. This is an advanced feature that is not covered in this book; for now use the default and press the Next button, which takes you to the confirmation page. When you press the Create button on the Confirmation page, you will find yourself back on your Content page and your report is displayed in the new data section (see Figure 11–39).

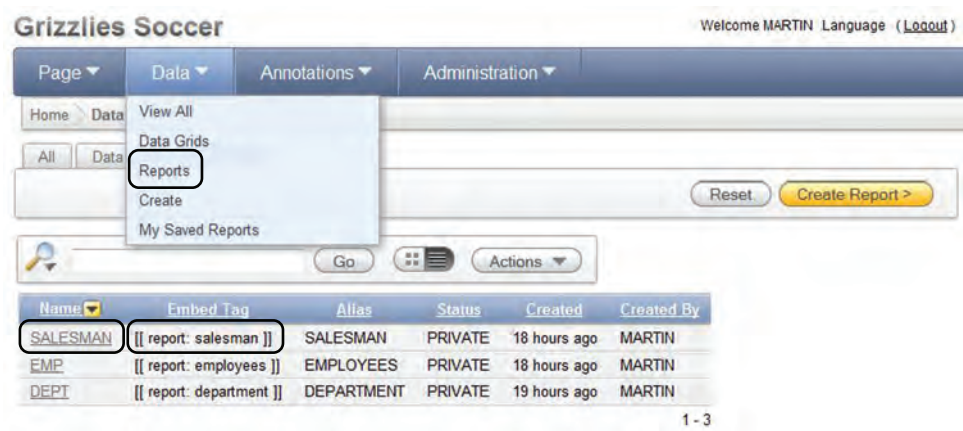


Figure 11–34. Navigate to a report



Figure 11–35. Report Data page

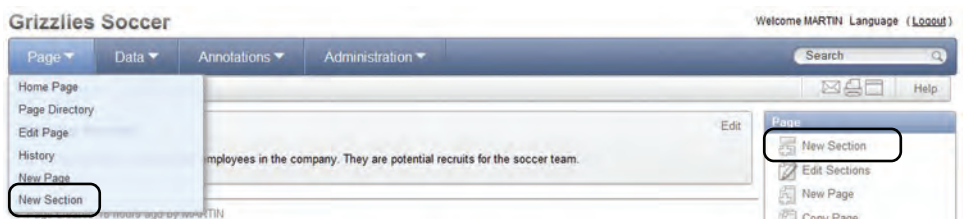


Figure 11–36. New Section link

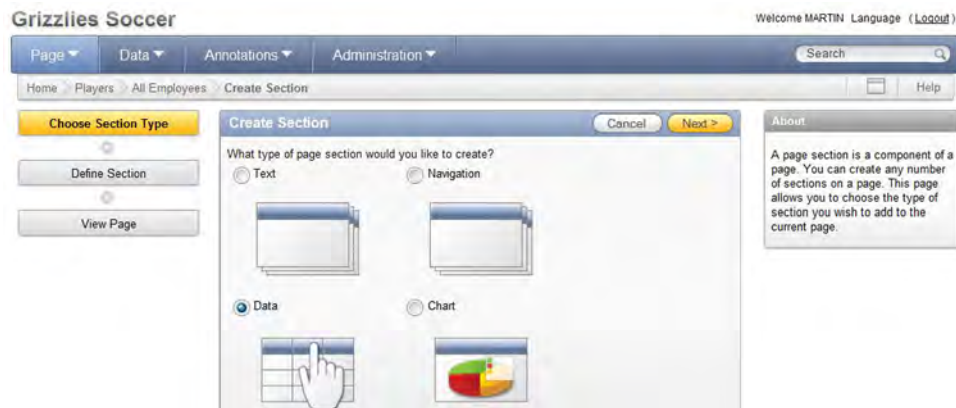


Figure 11-37. New Section wizard - choose section type



Figure 11-38. New Section wizard - data source

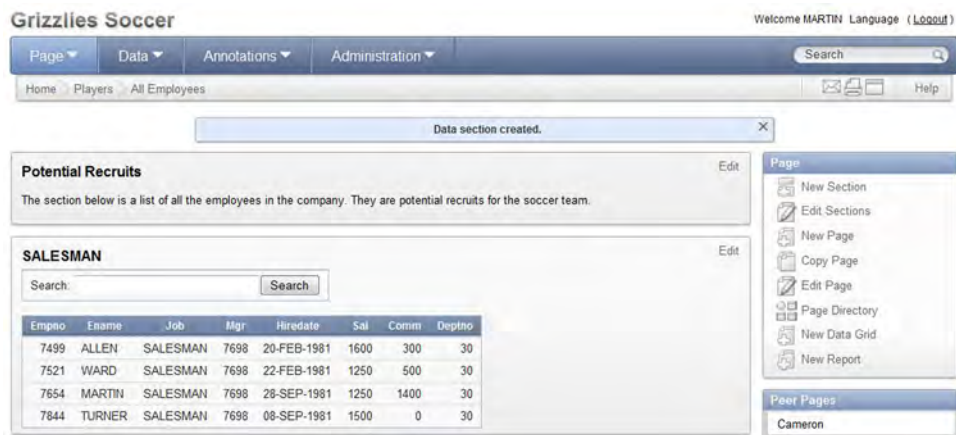


Figure 11-39. Data section containing a report

Data Grids

A data grid is the most complex part of a websheet. However, if you have had a bit of experience with a spreadsheet, you probably won't have much difficulty in learning how to use a data grid in a websheet.

This section will highlight some of the features of data grids. Chapter 12 walks you through the steps that are required to create a data grid from scratch.

Data grids are used to organize data in a column and row format. Both the design and data are held within the websheet environment; no external configuration required.

Data grids are closely related to reports. Like reports, you put them into data sections and you can embed links to them in text sections. Figure 11–40 illustrates a data grid that is displayed in a data section. In this context, the data grid is read-only, like a report. The Edit link goes to a page that allows you to edit the data section, not the data grid itself.

The screenshot shows the Grizzlies Soccer websheet interface. At the top, there's a navigation bar with tabs for Page, Data, Annotations, and Administration. A search bar is on the right. Below the navigation bar, there's a Home link and a Help icon. The main content area is divided into several sections:

- Navigation:** A tree view showing the site structure: Home, Players, Results, and Schedule. An 'Edit' link is next to it.
- Important News:** A section with two news items. An 'Edit' link is next to it.
- Upcoming Games and Practices:** A section containing a search bar and a table of upcoming events. An 'Edit' link is next to it.

The 'Upcoming Games and Practices' table has the following data:

Event	Date	Time	Field	Against
Game	13-JUN-2010	13:00	Foothills	Calles
Practice	16-JUN-2010	18:45	Bottomlands	

On the right side of the websheet, there's a sidebar with links for Page, Child Pages, and Peer Pages. The 'Page' section includes links for New Section, Edit Sections, New Page, Copy Page, Edit Page, Page Directory, New Data Grid, and New Report. The 'Child Pages' section lists Players, Results, and Schedule. The 'Peer Pages' section states 'No peer pages exist'.

Figure 11–40. Data grid displayed in a data section

You navigate to the context where you maintain the data grid's data by using the Data drop-down menu (see Figure 11–41). This list of data grids is identical to the list of reports. You navigate to in individual data grid by clicking on its link in the Name column.

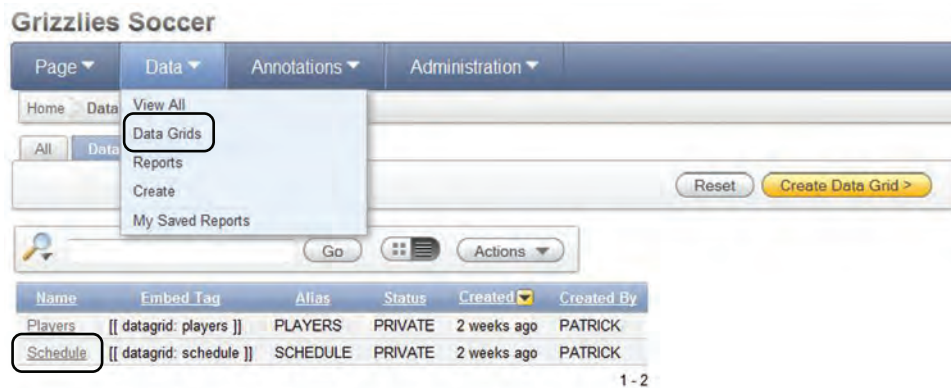


Figure 11–41. Navigating to a data grid's data entry context

Figure 11–42 shows you this example's Schedule data grid. The search text box, the Go button, the Reports drop-down list, and the Actions drop-down menu are standard interactive report features that have been discussed in a previous chapter.

You can change the data directly in the data grid. When you click on a cell, the cell turns white and you can type data directly into the cell. If you have configured a column as a date, a pop-up calendar automatically appears to help with accurate date entry. You can also configure a column to have a list of values; in this case, the cell contains a drop-down list. You can also use the pencil icon on the grid's left side to link to a Form page that edits an individual row (see Figure 11–43). This is convenient when the data grid contains many columns and is too wide for your computer screen.

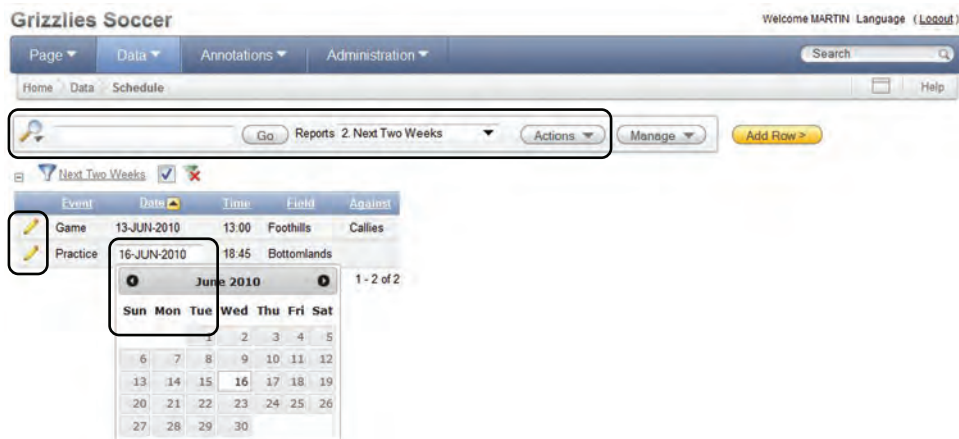


Figure 11–42. Editing data in a data grid

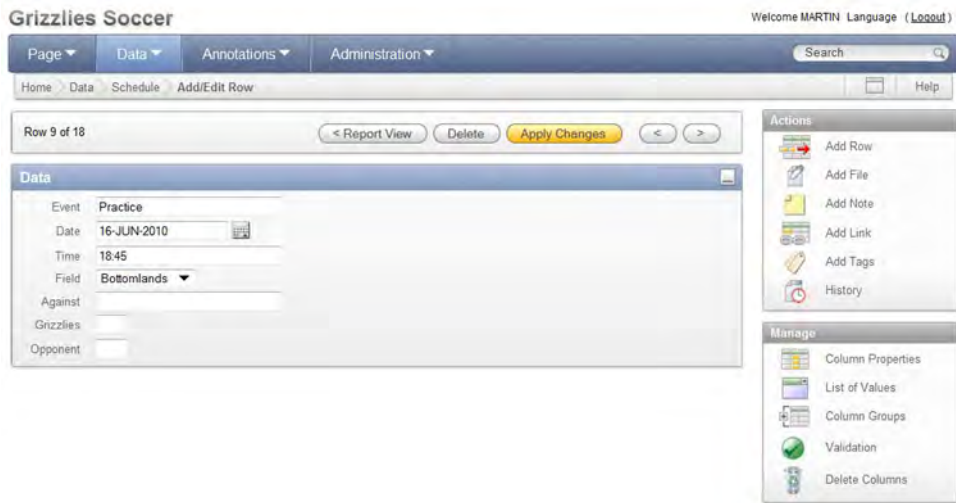


Figure 11–43. Editing data on a Form page for one row

You manage the structure of a data grid by selecting options from the Manage drop-down menu on a data grid. Figure 11–44 and Figure 11–45 illustrate all of the data grid configuration options.

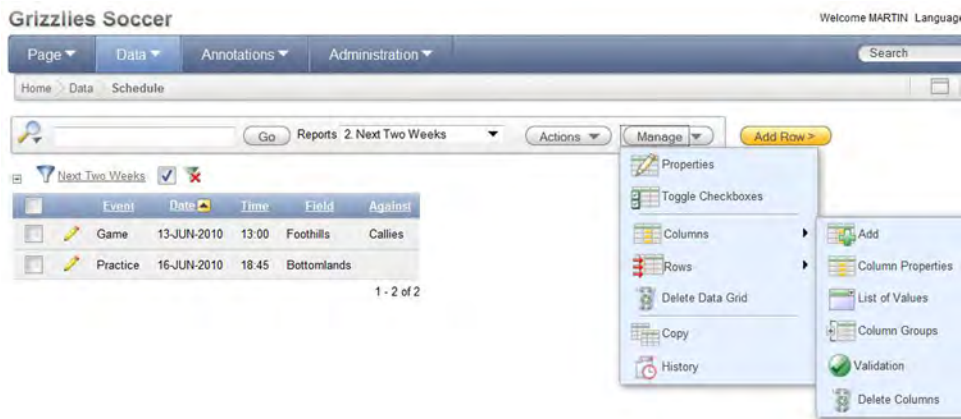


Figure 11–44. Data grid management - column options expanded

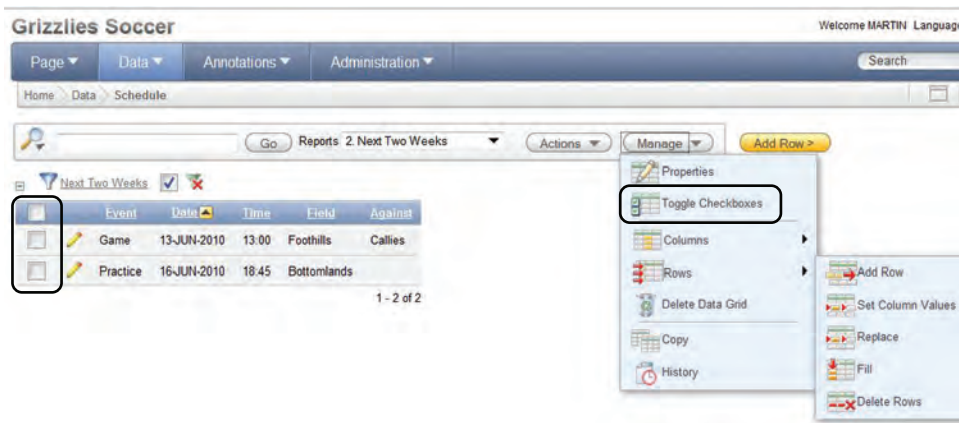


Figure 11-45. Data grid management - row options expanded

An interactive edit section is displayed above the data grid when you select one of the Manage menu options. This is illustrated in Figure 11-46. The edit sections all contain their own Cancel and Apply buttons. You must press the Apply button to save your change.

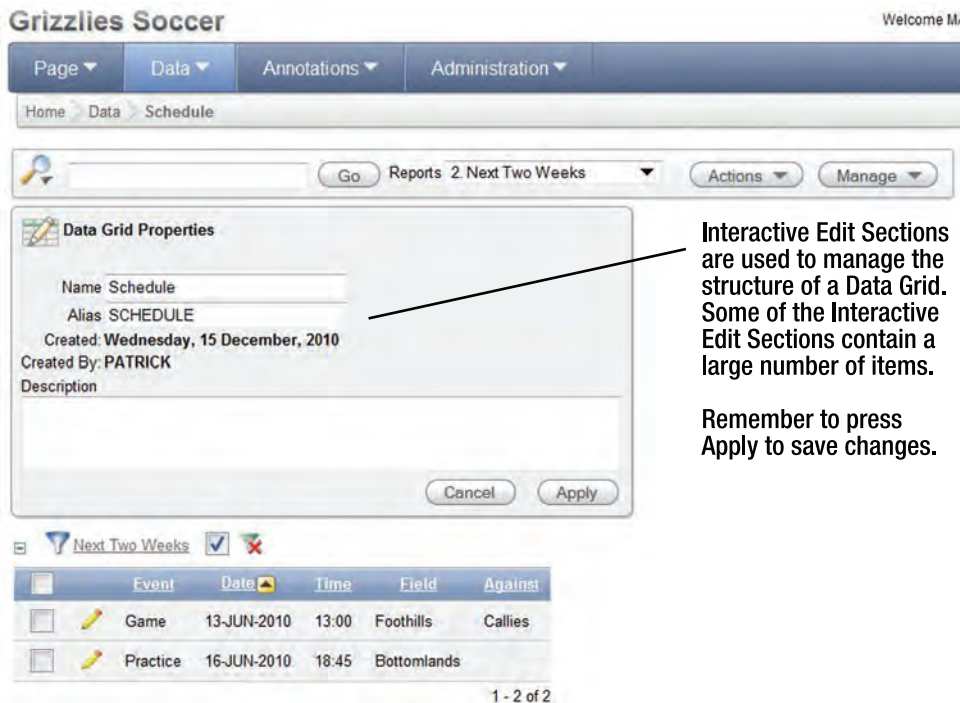


Figure 11-46. Data grid management – Data Grid Properties menu option

The Manage drop-down menus give you a rich set of options that allow you to use a data grid as a simple and friendly spreadsheet-like application. Most of the options are simple to use; they are illustrated in more depth in Chapter 12. The Manage menu options are summarized below:

- **Properties:** Edit the data grid properties.
- **Toggle Checkboxes:** Toggle the row selection check boxes on and off. The row selection check boxes are used to perform bulk updates on selected rows. This feature is highlighted in Figure 11–45.
- **Columns:**
 - **Add:** Add a new column to the data grid.
 - **Column Properties:** Change the properties of a column after it has been created.
 - **List of Values:** Create a named list of values. A named list of values can be used in more than one data grid.
 - **Column Groups:** Create column groups.
 - **Validation:** Add data entry validations. You choose validations from a defined list of business rules. You can use several validations simultaneously to achieve a result. For example, to make sure a number is greater than zero, you use the "Column specified is NOT zero" validation together with the "Column specified does not contain any of the characters in Expression" validation where you enter a minus sign character in the Validation Expression text area. This last point illustrates the fact that all of the underlying data in a data grid is text—and that sometimes you need to use more than one validation rule to achieve a single result.
 - **Delete Columns:** Delete one or more columns.
- **Rows:**
 - **Add Row:** The data entry page is displayed. New data is entered.
 - **Set Column Values:** Data is entered into many rows in a single column. A value can be set for All Rows, Selected Rows, or Empty Rows.
 - **Replace:** This is a find-and-replace feature that is similar to text editors and word processors. It can be applied to All Rows or Selected Rows.
 - **Fill:** Fill a column's null cells with the value that is found above them.
 - **Delete Rows:** Rows are deleted from the data grid. This is done for All Rows, Selected Rows, or Rows with Empty Columns.
- **Delete Data Grid:** Delete the data grid from the websheet.
- **Copy:** Make a copy of the data grid.
- **History:** This is an audit trail for the data.

Chart Sections

Chart sections are an easy way to add graphics to your content. First, you must create a report or data grid that contains at least one numeric column. Second, you run the Create Chart wizard. The wizard links the report or data grid to the chart and sets up the axis labels. The details and visual result of this simple process are covered Chapter 12.

Annotations

Annotations are used to add additional content to your pages and individual rows in data grids. There are four types of annotations.

Files: Two types of files can be uploaded into your websheet. Image files are displayed within your text sections. Other file formats, like PDF, can be uploaded to a websheet and then downloaded to the end user's computer. In both cases you use markup syntax to achieve your result.

Tags: Tags are free-form text words that are attached to your content to enhance the websheet's Search feature.

Notes: Notes are like a sticky note. When you can add a note to a page, it appears in a section on the right side of the page.

Links: Links allow the users to navigate to any valid URL. Annotation links are associated with rows in a data grid; they can't be associated with a page. To add a link to a page, you use markup syntax, not an annotation.

Figure 11–47 illustrates the Annotation section that appears on a websheet page. Clicking on the various links allows you to add, change, and delete your annotations.

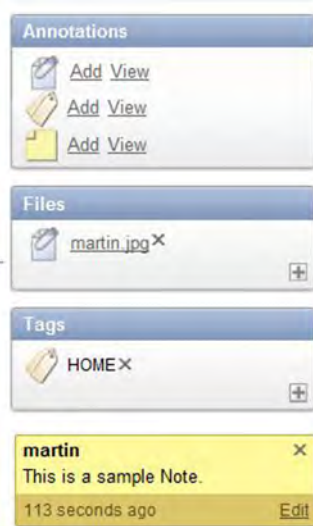


Figure 11–47. Annotation section on the lower right side of a page

Administration

Websheets, like any blog or wiki, should be reviewed periodically by a moderator who has been given the administrator role. This allows the moderator to view the Dashboard and Monitor Activity pages (see Figure 11–48). These pages and the underlying reports give the moderator the tools to see which pages are the most and least popular, how long it takes to render a page, who is using the websheet, and many other parameters that help the moderator to make sure that the websheet is working optimally.

As you work through this chapter and Chapter 12, you will notice links that are labeled History. The History links are placed throughout the websheet's structural navigation areas. They take you to context-sensitive report pages that contain comprehensive audit trails of websheet changes. The contributors, when they know that their changes are audited, will make an effort to hold their contributions to a high standard.

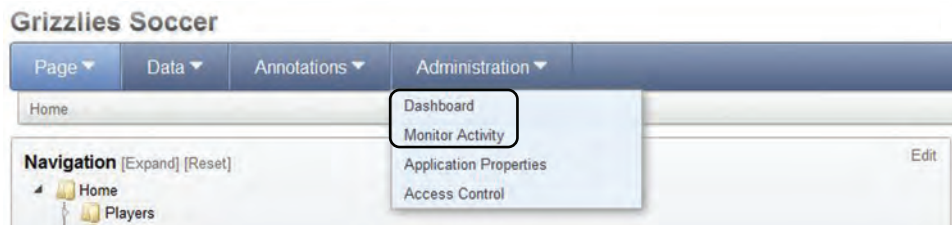


Figure 11–48. Navigating to the Dashboard and Monitor Activity pages

Summary

This chapter has given you a good look at websheets so that you can now use them in innovative and creative ways. Chapter 12 complements this chapter by working through, on a step-by-step basis, the construction of a websheet from the ground up.



A Websheet Example

The previous chapter covered all the different features in websheets. In this chapter, you will use these features to build an application from scratch in websheets.

You will build an application that will manage your corporate soccer team. Currently, the player roster and schedules are maintained in spreadsheets. When the schedule is updated, you e-mail the spreadsheet to all the players. It has been very frustrating to manage the team this way.

A websheet is a good way to manage the soccer team because websheets can be built with minimal developer or DBA assistance. All the files, along with a copy of the final application, can be found in the example download described in the introduction to this book.

Setup

In order to highlight the capability of websheets to interact with objects in your database, you will create some database objects that will be referenced throughout the chapter. These objects will simulate an existing users table and login function for your organization. Run the following script, which is included in the example download, in SQL*Plus or directly in APEX using the SQL workshop:

```
-- Create Users Table
CREATE TABLE tusers (
  user_id      NUMBER (5, 0) PRIMARY KEY,
  user_name    VARCHAR2 (10) NOT NULL UNIQUE,
  password     VARCHAR2 (10) NOT NULL,
  active_flag  VARCHAR2 (1) NOT NULL
);

-- Create sequence for IDs
CREATE SEQUENCE sn_users;

-- Create Users
-- Note: You should not store passwords in clear text.
-- This was done for demonstration purposes.
INSERT INTO tusers ( user_id, user_name, password, active_flag)
  VALUES (sn_users.NEXTVAL, 'martin', 'martin', 'Y');

INSERT INTO tusers ( user_id, user_name, password, active_flag)
  VALUES (sn_users.NEXTVAL, 'chris', 'chris', 'Y');

INSERT INTO tusers ( user_id, user_name, password, active_flag)
  VALUES (sn_users.NEXTVAL, 'cameron', 'cameron', 'Y');

-- Authentication Function
```

```

CREATE OR REPLACE FUNCTION f_login (p_username IN VARCHAR2, p_password IN VARCHAR2)
RETURN BOOLEAN
AS
    v_count    PLS_INTEGER;
BEGIN
    SELECT COUNT (user_id)
    INTO v_count
    FROM tusers
    WHERE LOWER (user_name) = LOWER (p_username)
        AND password = password
        AND active_flag = 'Y';

    IF v_count = 1 THEN
        RETURN TRUE;
    END IF;

    RETURN FALSE;
END f_login;
/

COMMIT;

```

Creating and Configuring a Websheet Application

To create a websheet application you need to have access to the APEX builder. Once you have logged in, proceed with the following steps to create the websheet application:

1. Click the Application Builder icon.
2. Click the Create button
3. Select Websheet for the application type, as shown in Figure 12–1, and click the Next button.

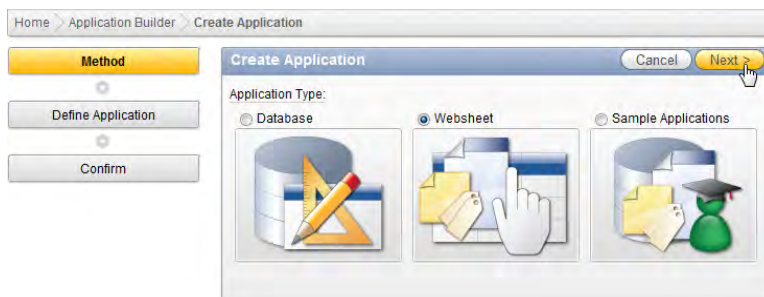


Figure 12–1. Create websheet application

4. On the next screen enter Grizzlies Soccer for the application name and use the default application ID. Leave the Description and Home Page section blank. Click the Next button.
5. Click the Create button to finish the wizard.

6. You should now see a success page with the options to run the application or edit properties (see Figure 12–2).

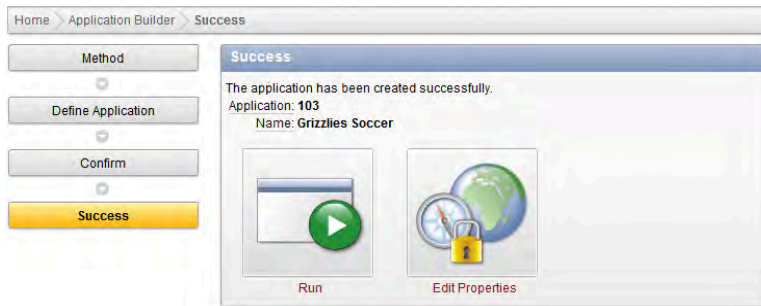


Figure 12–2. Create websheet success

As the Grizzlies are your corporate soccer team, you'd like to be able to have users login using their current corporate accounts. To use the corporate authentication, you will need to configure the application and modify the authorization scheme.

Since you have not defined an authentication scheme, you will need access to the APEX builder to modify the application properties. This will be the last portion of the process to require APEX builder access. The following steps describe how to modify the application properties:

1. If you are still on the Success page, click on the Edit Properties icon. If you changed screens, then click on Application Builder. Select the new application that you created (Grizzlies Soccer), then click the Edit Properties icon.
2. Modify the items in the following sections:
 - Application: Enter DD-MON-YYYY for the application date format, as shown in Figure 12–3.

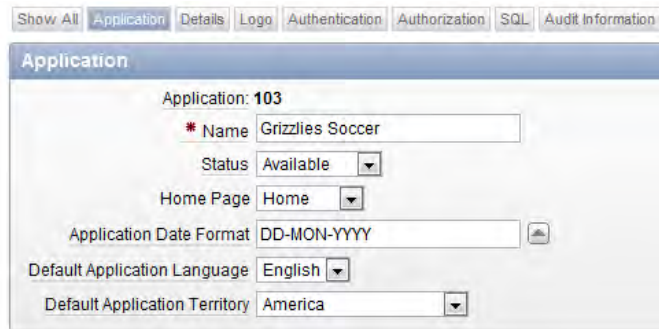


Figure 12–3. Application date format

- Authentication: Select Custom and enter the single statement `return f_login` in the authentication function field, as shown in **Figure 12-4**. Note that `f_login` refers to the function you created in the setup section of this chapter.

Authentication configuration page showing the Custom authentication method selected. The Authentication Function is set to `return f_login`.

Figure 12-4. Custom authentication

- SQL: Select Yes for Allow SQL Tags and Reports. This will allow you to reference tables and views in the schema. After selecting Yes, an Add Object button will appear. Click the Add Object button and enter TUSERS for the object name, as shown in **Figure 12-5**. This will allow users to quickly select the TUSERS table when creating reports. Click the Create button, which will bring you back to the application properties page.

Suggested Objects configuration page. The Suggested Object section shows the Schema as `BEGINNINGAPEX4` and the Object Name as `TUSERS`. The Existing Suggested Objects section shows "No objects associated."

Figure 12-5. Adding an object

3. Scroll back to the top of the properties page and click the Apply Changes button to save your changes.
4. Authorization: Before you can use a custom authentication scheme, you'll need to define an administrator for the application. To define an application administrator, edit your websheet application properties again. Scroll down and click the Edit Access Control List button.

On the new page, click the Create Entry button. In the username field, enter Martin and select Administrator, as shown in Figure 12–6. Click the Create button to register Martin as an administrator. You will be brought back to the Access Control List page. Click the Cancel button to return to the application properties page. Click the Apply Changes button to save your changes.

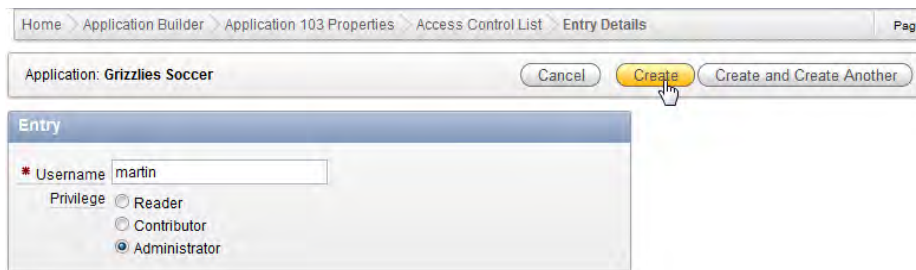


Figure 12–6. Access Control list: adding an administrator

Adding Content to a Websheet

In the previous section, you created and configured a websheet application to manage your corporate soccer team. In this section, you'll create data grids and add content to the application.

Run the websheet application and login as martin/martin the site administrator. Once logged in, the page should look like **Figure 12–7**. You're now ready to create your first data grid.

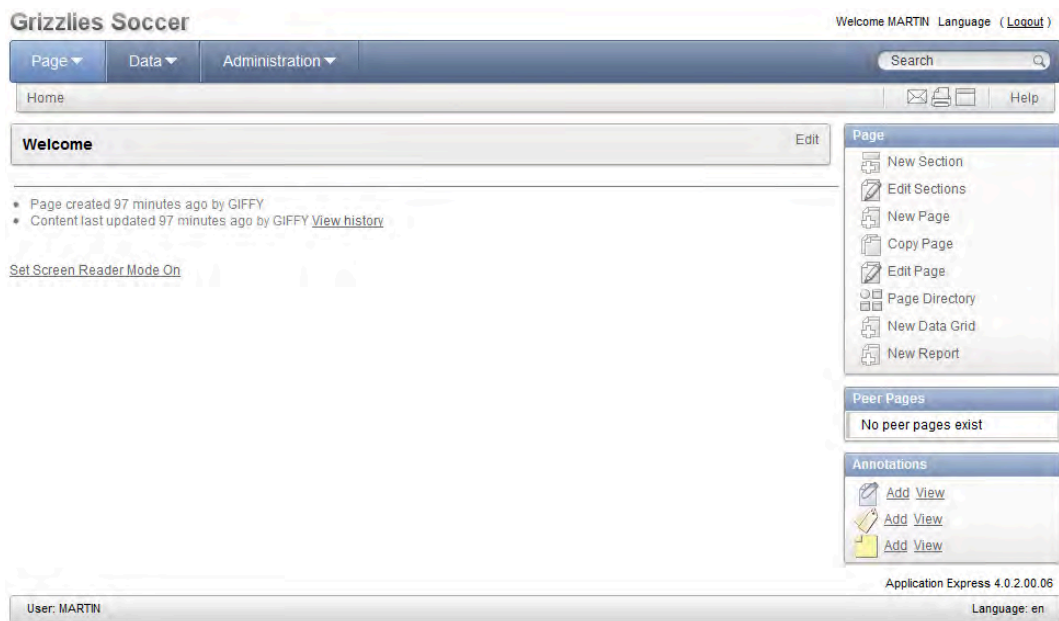


Figure 12–7. Initial websheet application

■ **Note** The URL to run your websheet application is `<apex_url>/ws?p=<web_sheet_id>`. For example: `http://apex.clarifit.com/apex/ws?p=103` where 103 is the websheet application ID.

Creating Data Grids


The first thing you'll do is create some custom tables called data grids. There are two ways to create data grids: by pasting existing data in from a spreadsheet, and from scratch by manually defining each column. You'll create data grids from both sources.
















You currently keep your game and practice schedule in a spreadsheet. You can import data and simultaneously create a data grid using copy and paste. Here is the process to follow:

1. Click the Data tab at the top of the application.
2. Click the Create option in the drop-down menu.
3. Select Data Grid and click the Next button.
4. Select Copy and Paste as the input method and click the Next button.
5. Enter Schedule in the name and alias fields.

6. Open Grizzlies_Schedule.csv, which can be found in the sample code for this chapter, and select all the fields including the header. Copy these values and paste them into text area. Ensure that the First Row Contains Column Headings checkbox is checked and click the Upload button.
7. You should now see the data in an interactive report, as shown in Figure 12–8.

Home > Data > Schedule



	Event	Date	Time	Field	Against	Grizzlies	Opponent	delete
	Practice	11-MAY-10	18:45	Bottomlands				delete
	Practice	18-MAY-10	19:00	Bottomlands				delete
	Practice	22-MAY-10	10:00	Forrest Lawn				delete
	Game	26-MAY-10	20:15	Airdrie	Airdrie	2	1	delete
	Game	02-JUN-10	20:15	Tom Brooks	Chinooks	2	3	delete
	Practice	04-JUN-10	21:00	Bottomlands				delete
	Game	08-JUN-10	19:00	Bottomlands	Blizzard	4	1	delete
	Game	13-JUN-10	13:00	Foothills	Callies			delete
	Practice	16-JUN-10	18:45	Bottomlands				delete
	Game	30-JUN-10	19:15	Bottomlands	Darts			delete
	Game	07-JUL-10	18:45	Glenmore	Vipers			delete
	Game	18-JUL-10	15:30	Mount Royal	Darts			delete
	Game	30-JUL-10	19:00	Bottomlands	Wolves			delete
	Practice	01-AUG-10	11:00	Bottomlands				delete
	Game	09-AUG-10	20:00	Oakridge	ASP			delete

1 - 15 of 18 >

Figure 12–8. Data grid result

8. The last column is tagged as “delete”. This is a workaround for a bug in the cut and paste mechanism within websheets¹. To delete this column, click the Manage button and select Columns ► Delete Columns, as shown in Figure 12–9.

¹ The bug mentioned does not load data correctly if the last column(s) in the data have multiple null entries. By appending a dummy column to the end that has a non-null value, APEX will load all of the data correctly. This bug is present through at least version 4.0.2.0.6.

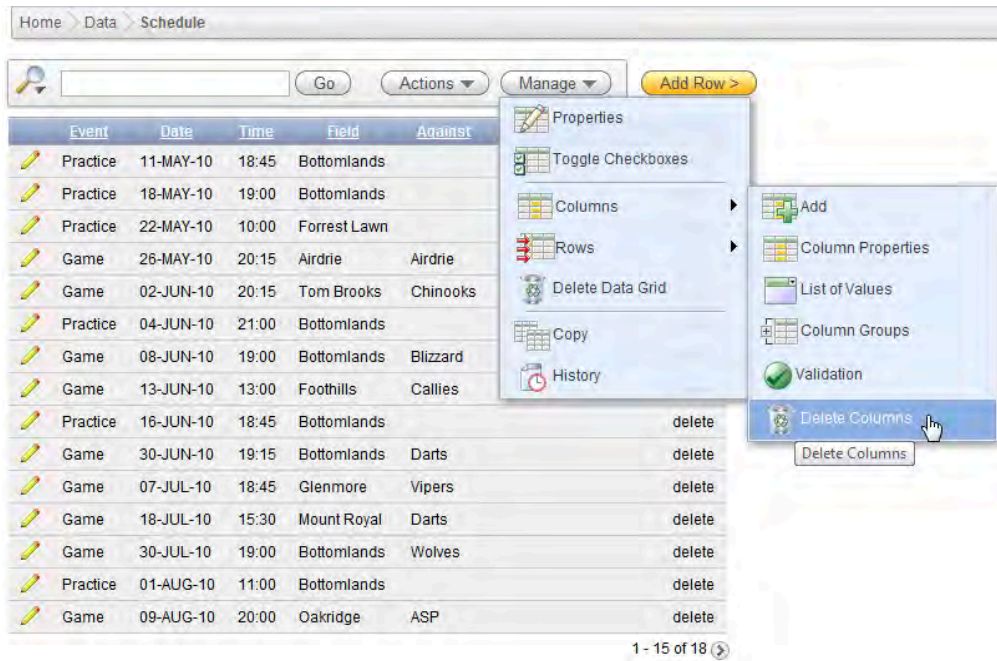


Figure 12–9. Delete Column option

9. Select the Delete Column option, check Confirm to perform this delete action, and click the Delete button, as shown in Figure 12–10.

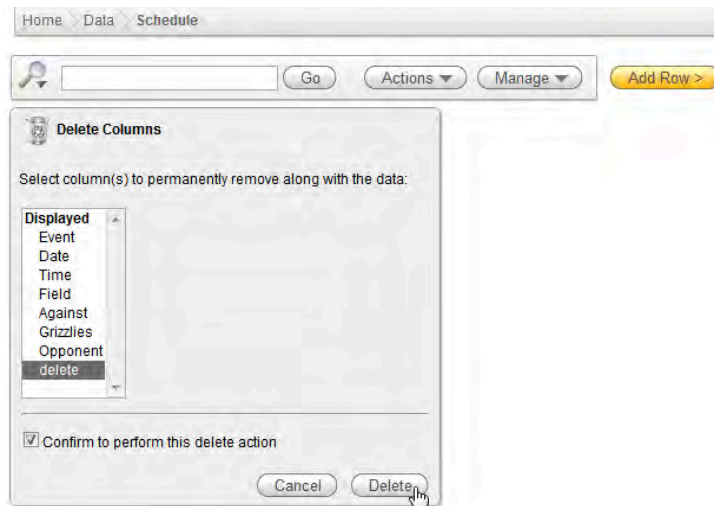


Figure 12–10. Delete column confirmation

You also need to create a data grid to keep track of the number of goals each player scores. You don't have any existing data ready to copy and paste from a spreadsheet, so this time it makes sense to create your data grid manually. Here are the steps to follow:

1. Click the Data tab at the top of the application.
2. Click the Create menu option.
3. Select Data Grid and click the Next button.
4. Select From Scratch as the input method and click the Next button.
5. Fill out the data grid definition as shown in Figure 12–11 and click the Create button to finish creating the data grid.

Column Name	Type	Move
Name	String	▼ ▲
Goals	Number	▼ ▲

Figure 12–11. Create data grid from scratch

Applying Constraints

Now that you have some data grids, you'll need to add some constraints to them. Since data grids are not database objects, you must use the websheet interface to apply constraints.

In the Players data grid, you need to ensure that all the fields contain data, and that the default for the Goals column is 0. To apply these constraints, follow these steps:

1. Click the Data tab.
2. Click the Players data grid icon.
3. Click the Manage button and select Columns ► Column Properties, as shown in Figure 12–12

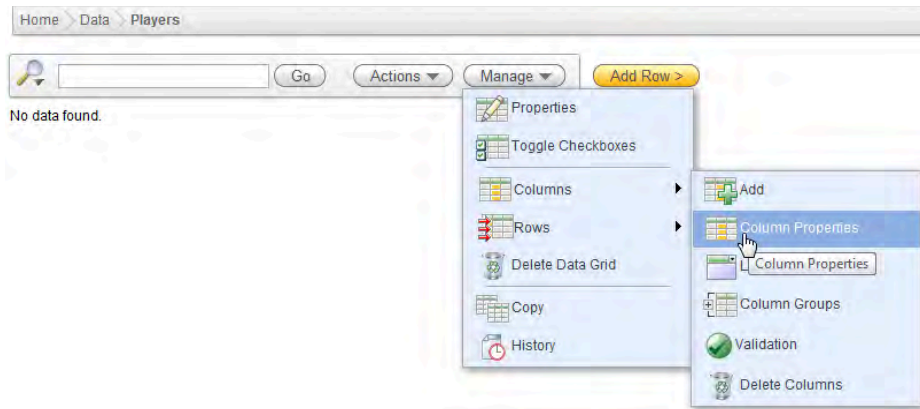


Figure 12–12. Data grid column properties

4. Select Name in the Column Name select list.
5. Select Yes for value required.
6. You must explicitly save your changes before modifying another column. Click the Apply button at the bottom.
7. Open the Column Properties section again (repeat the third step).
8. Select Goals in the Column Name select list.
9. Select Yes for value required.
10. In the Default Text field, enter 0.
11. Click the Apply button.

When you create players, both the Name and Goals are required values. Goals will be defaulted to 0.

Adding Players

To add players to the Players data grid, click the Add Row button. For this demo, add the players and their goals as shown in **Figure 12–13**.

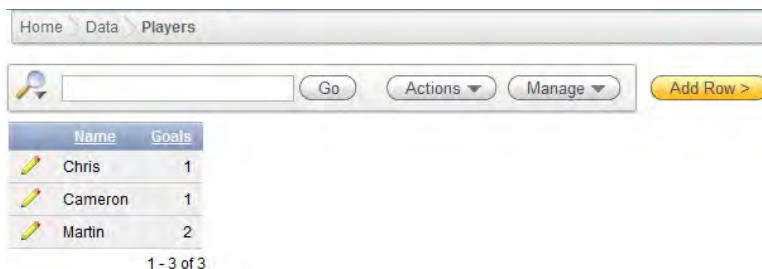


Figure 12–13. Players data

Creating Alternate Default Reports

Now that you have some data in the data grids, you'll create some alternate default reports, which you will reference when creating sections. Data grids allow you to save reports just like interactive reports do. Alternate default reports are saved data grid reports that can be displayed throughout your websheet application.

■ **Note** Some of the reports that you will create are date sensitive. Normally, you'd use SYSDATE as a reference point. Instead, you'll use a static date of 10-Jun-2010 to simulate a common SYSDATE. This will ensure that you see the same data that is shown in this book.

The first alternate report you'll create will be to highlight the games and practices for the next two weeks. To create this report, follow these steps:

1. View the Schedule data grid.
2. Click the Actions button ► Filter.
3. Select Row as the filter type.
4. Enter Next Two Weeks as the name.
5. Enter the following as the Filter Expression: `B >= to_date('10-jun-2010', 'dd-mon-yyyy')` and `B < to_date('10-jun-2010', 'dd-mon-yyyy') + 14`. Click the Apply button.
6. Order the Date column as ascending by clicking on the Date column heading and choosing the Ascending sort icon.
7. Hide the Grizzlies and Opponents columns since you don't have scores for future games. To do this, use the Select Columns option from the Actions menu.
8. Click the Actions button ► Save Report.
9. Select Default Report in the Save select list.
10. Select Alternate as the Default Report Type and enter Next Two Weeks as the name.
11. Click the Apply button. The report should now look like Figure 12-14.



Figure 12–14. *The next two weeks*

Since you have already learned how to manipulate and create saved interactive reports, create the following alternate default reports for the Schedule data grid:

Remaining Games: This report lists all the games left in the season.

Remaining Practices: This report lists all the practices left in the season.

Results: This report lists all the games that have been completed along with the scores.

Creating Page Sections

In this section, you'll modify existing pages and create new sections. You'll cover some of the different types of content that you can add to websheets.

To start, you'll modify the home page. You'll create several sections that will help the players get the most important information right away. Let's modify the Welcome section to contain important news by following these steps:

1. Click the Page tab at the top.
2. Select Home Page.
3. Click the Edit link on the right side of the Welcome region.
4. Change the tile from Welcome to Important News.
5. Enter the following in the content section:

Fees: Don't forget to pay your fees before the next game (13-Jun) or else you can't play!
We won our last game and are now 2-1. Check out the `[[results | Results]]` page.

The special notation involving the square-brackets will create a link to the Results page that you'll create later in this chapter.

6. Click the Apply Changes button.

The home page should now look like **Figure 12–15**.

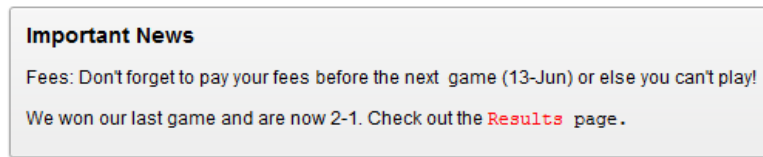


Figure 12–15. Important news

■ **Note** You'll notice that the link to the results page is in red. That's because you created an invalid link. Once the Results page is created, the link will turn grey.

Next, you'll create a new section on the home page to highlight the upcoming games and practices. This section will reference one of the alternate default saved reports that you created earlier. To create the section, follow these steps:

1. From the home page, click the New Section link on the right side located in the Page region.
2. Select Data for the page section type and click the Next button.
3. Select Schedule for the data grid and change the title to Upcoming Games and Practices. Select a style (use 2 for all sections in this demo) and click the Next button to continue.
4. In the next section, you can select which saved report to display. Select the Next Two Weeks alternate report and click the Next button.
5. On the confirmation page, click the Create button.

The new section should look like **Figure 12–16**.



Figure 12–16. Upcoming games and practices

Each week the coach likes to highlight a player of the week. The coach wants to include a picture along with some text in this section. This week I'm the lucky recipient of the Player of the Week award. To create the Player of the Week section, follow these steps:

1. The first thing you'll do is upload the player's picture. Under the Annotations region on the right, click on the Add File link, as shown in **Figure 12–17**.

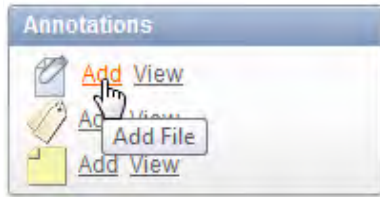


Figure 12–17. Add file

2. In the File section, upload `martin.jpg` and click the Add button. You will be brought back to the home page.
3. Click the Add New Section link.
4. Select Text as the section type and click the Next button.
5. Enter Player of the Week for the title.
6. Enter the following in the content text area:

Martin scored 2 goals!

[[image: martin.jpg]]

7. Click the Create button

The Player of the Week section should now contain an image, as shown in **Figure 12–18**. Each week the coach can easily upload a new picture and modify this section.

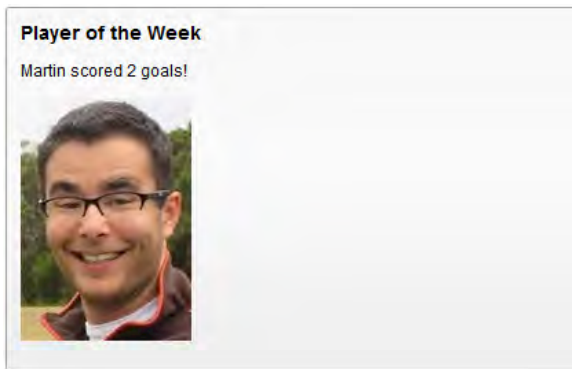


Figure 12–18. Player of the Week section

You also need a page to display the list of players and include a graph to view the top scorers on the team. To create and modify the player's page, follow these steps:

1. Click the New Page link on the right side.

2. In the name field, enter Players. For the page alias, enter PLAYERS. Select Home as the parent page. Click the Create Page button.
3. On the new Players page, remove the Welcome section by clicking its Edit link and clicking Delete button.
4. Add a new section called Players, which is a data grid report referencing the Players data grid.
5. To add the graph, click on the Add New Section link.
6. Select Chart as the section type and click the Next button.
7. Select Column as the chart type and click the Next button.
8. Select Players for the data grid and enter Goals as the section title. Click the Next button.
9. On the next page, you should only have the Primary Report option available since you didn't create any alternate default reports. Click the Next button to continue.
10. Modify the chart section as shown in Figure 12–19 and click the Next button.

Figure 12–19. Chart definition

11. On the confirmation screen, click the Create button.
- The new chart region should look like **Figure 12–20**.

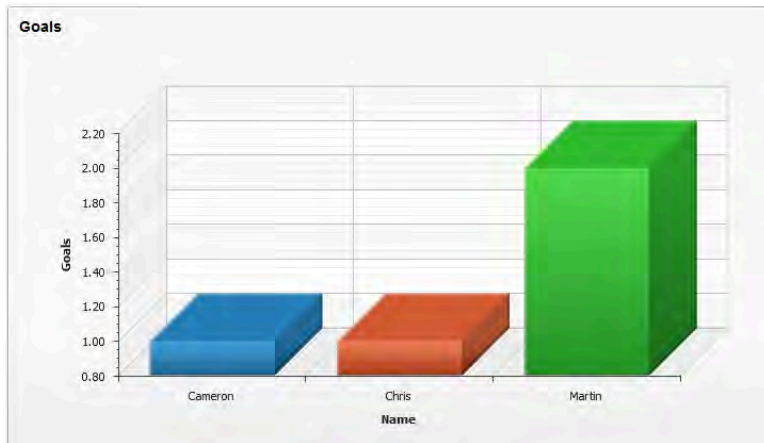


Figure 12–20. Goals chart

The last modification you need to make for the players page is to add a navigation section. This will allow users to quickly go to each section on the page rather than having to scroll down the page. To add the section navigation section, follow these steps:

1. Click the Add New Section link.
2. Select Navigation as the section type and click the Next button.
3. Select Section Navigation for the navigation type and click the Next button.
4. Enter a title for the section and 1 for the sequence. Setting the sequence to 1 will make it the first section on the page. Click the Create button.

When someone views the page they'll be able to quickly navigate to each section via the navigation section.

You should now be comfortable creating and modify pages. Before creating the final section, create the following pages, which are child pages of the home page:

Results: This page will display the Results saved report. The Results page should look like **Figure 12–21**.

Schedule: This page contains two sections. The first section will display the Remaining Games saved report and the other section will display the Remaining Practices saved report which you created earlier. The Schedule page should look like **Figure 12–22**.

Home > Results					
Results					
Search: <input type="text"/>		<input type="button" value="Search"/>			
Date	Time	Field	Against	Grizzlies	Opponent
26-MAY-2010	20:15	Airdrie	Airdrie	2	1
02-JUN-2010	20:15	Tom Brooks	Chinooks	2	3
08-JUN-2010	19:00	Bottomlands	Blizzard	4	1

Figure 12–21. Results page

Home > Schedule				
Remaining Games				
Search: <input type="text"/>		<input type="button" value="Search"/>		
Event	Date	Time	Field	Against
Game	13-JUN-2010	13:00	Foothills	Callies
Game	30-JUN-2010	19:15	Bottomlands	Darts
Game	07-JUL-2010	18:45	Glenmore	Vipers
Game	18-JUL-2010	15:30	Mount Royal	Darts
Game	30-JUL-2010	19:00	Bottomlands	Wolves
Game	09-AUG-2010	20:00	Oakridge	ASP
Game	20-AUG-2010	21:00	Tom Brooks	Storm
Game	25-AUG-2010	19:15	Bottomlands	Tigers
Remaining Practices				
Search: <input type="text"/>		<input type="button" value="Search"/>		
Event	Date	Time	Field	Against
Practice	16-JUN-2010	18:45	Bottomlands	
Practice	01-AUG-2010	11:00	Bottomlands	
Practice	23-AUG-2010	19:15	Bottomlands	

Figure 12–22. Schedule page

The next section you'll create will provide a list, along with links, to all the pages. To create this navigation section, follow these steps:

1. Go to the home page.
2. Click the Add New Section link.
3. Select Navigation as the section type and click the Next button.
4. Select Page Navigation as the navigation type and click the Next button.
5. If you want, modify the title and set the sequence to 1, then click the Create button.

The new section should look like **Figure 12–23**.

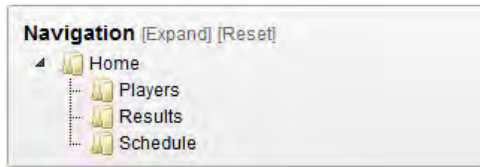


Figure 12–23. Page navigation

SQL Tags

Websheets allow administrators and contributors to query tables and views in the schema. They can create reports that are similar to data grids, except they are read-only. They can also include query results directly in sections that are called SQL Tags.

On the Players page, you'll add a section to display the number of registered users who have access to the application. You'll include a SQL tag in the section. To create the section, follow these steps:

1. Go to Players page.
2. Click the Add New Section link.
3. Select Text as the section type and click the Next button.
4. Set the sequence to 5 and the title to Active Registered Users.
5. Enter the following text in the content section and click the Create button:

```
We currently have [[sqlvalue: select count(*) from tusers where active_flag = 'Y' ]] active
registered users. They are: [[sql: select initcap(user_name) "Name" from tusers order by
user_name ]]
```

The new section should look like **Figure 12–24**. Notice the `select count` query in the first line of the preceding code. That query generates the value 3 shown as the number of active users in the figure. Similarly, the second `select` statement in the preceding code generates the list of player names.

When using SQL tags, you need to explicitly define if the query will return a single value or multiple rows and columns. A SQL tag defined as `sqlvalue:` means that a single value will be returned and be embedded within a sentence such that the single value appears as a word in the sentence. Using `sql:` means that multiple rows and columns will be returned. When a query returns multiple rows, its results are displayed in the spreadsheet-like format that you see in Figure 12–24.

The Search box that you see in Figure 12–24 is a result of the second query returning multiple rows. Whenever a query's results are displayed as a spreadsheet-like grid, that grid is preceded by a search box that you can use to quickly find specific result rows.

Active Registered Users

We currently have 3 active registered users. They are:

Search:

Name
Cameron
Chris
Martin

Figure 12–24. Section with SQL tags

Access Controls

The last thing you need to do for your application is give the other players on the team access to it. You already gave access to Martin when you created the application. You need to give the other players, Chris and Cameron, access to view the application. To grant access to the application, follow these steps:

1. Click on the Administration tab at the top of the screen.
2. Click the Access Control link.
3. Click the Create Entry button.
4. Enter Chris as the username and select Reader as the privilege. Click the Create and Create Another button.
5. Enter Cameron as the username and select Reader as the privilege. Click the Create button.

Now Chris and Cameron can log into the application. They will not be able to modify any of the sections or data grids. If you need to give someone access to modify the application, you can grant them the Contributor role in the access control section.

Summary

The last two chapters introduced websheets and what they can do. From here you can go on to make complex websheet applications without having to know much about databases or SQL. Now that you have a base knowledge of websheets, installing and analyzing the websheet sample application would be a good next step to understanding the capabilities of a websheet.



Extended Developer Tools

While developing the sample application in the previous chapters you saw many features in the APEX development tool. This chapter will highlight advanced development features in APEX that were not covered in the previous chapters. These features—or tools, as I prefer to call them—may help when developing large applications in a corporate environment.

■ **Note** This chapter assumes that you are comfortable with APEX and understand the fundamentals. If you are still not comfortable developing an APEX application, I strongly recommend that you redo the examples from chapters 1 through 10 in order to become more comfortable with APEX and its development environment.

Page Locks

When developing in larger teams, development conflicts occur. A development conflict is when two developers are working on the same object at the same time and overwrite the other's changes.

■ **Note** For the remainder of this chapter, references to APEX objects imply page items, regions, lists, pages, etc.

Conventional web development tools, such as ASP, PHP, and JSP, contain multiple files that each represent a page or a set of functions within the web application. When developing with these tools, it's common practice to use a source control tool, such as Subversion, to manage all the changes. That's because source control tools can easily manage development conflicts between multiple developers as they are isolated to a single file.

APEX is different than the scripting languages just mentioned because developers do not work with files. All of the information is stored in tables within the database. Currently, these tables reside in the APEX_040000 schema. When you create an export of an application you get a single SQL file that contains many insert statements into these tables. Since APEX stores its content within the database, you can't use traditional source control tools to manage conflicts when developing in teams with multiple developers.

APEX Conflicts

To demonstrate a development conflict in APEX, imagine that you have two developers, Mina and Natalie, working on the same page within an application. If they are both adding and modifying page components at the same time, the page may not behave as expected for either of them.

APEX will prevent developers from modifying the same object at the same time by performing *optimistic locking*. **Table 13–1** shows the sequence of events that will occur as the two developers edit the same object. You can see at the end how APEX prevents Natalie from overwriting the changes made by Mina.

Table 13–1. Optimistic Locking Scenario

Step	Mina	Natalie
1	Edit P1_EMPNO	--
2	--	Edit P1_EMPNO
3	Edit help text to: "Mina's Help"	--
4	--	Edit help text to: "Natalie's Help"
5	Apply Changes	--
6	--	Apply Changes
7	--	Receive error message: ORA-20503: Current version of data in database has changed since user initiated update process. current checksum = "9A4B0C15B18DA7A89A3F15A0B982B500" application checksum = "4A157EAF6B2C7E67DA3BBDD655C4467C"

Optimistic locking only works when developers modify the same object. The problem occurs when multiple developers are modifying different objects on the same page at the same time. Modifying one object may affect the process of the entire page, which other developers may not be aware of. *Pessimistic locking* helps prevent trouble in that scenario. I'll discuss how to do pessimistic locking next.

Locking an APEX Page

The easiest way to prevent issues from occurring when developing an application with multiple developers is to lock a page before working on it. Locking a page will prevent other developers from modifying the page while you're working on it. Developers can still view the page and its components while a page is locked; they just can't make any modifications to the page.

The following process will lock a page:

1. On the page edit screen, click the lock icon in the top right corner, as shown in Figure 13–1



Figure 13-1. Lock page

2. You will see a report of page locks with a filter for your current page (Page 1 in this example). Check the checkbox for the current page and then click the Lock Checked button.
3. Enter in a page a comment (all page locks require a comment) and click Lock Page(s).
4. You will be brought back to the Page Locks report where you'll see your current page marked as locked.

Entering meaningful page lock comments is important since a history of page locks is maintained. If you use a case management tool, it's smart to reference the case number that you are working on when locking a page.

If another developer views a locked page, they'll see a notification message and that the lock icon is locked, signaling that the page cannot be modified. **Figure 13-2** shows an example of a locked page. The page lock comment is displayed above the Page Render region.

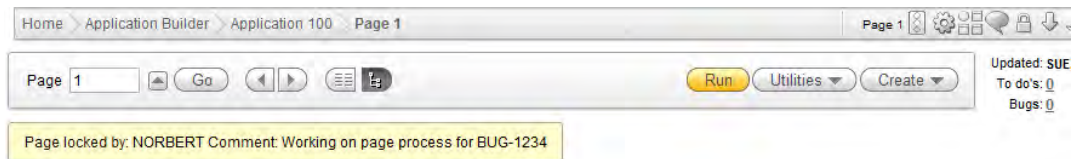


Figure 13-2. Locked page

Unlocking a Page

Only APEX administrators and the developer that locked a page can unlock it. If you are the developer that locked a page, the following process demonstrates how to unlock it:

1. Go to the locked page (Page 1 from the previous example) and click the lock icon in the top right corner.
2. You will see a report of page locks with a filter for your current page. Check the checkbox for the current page then click the Unlock Checked button.
3. You will be brought back to the Page Locks report page. You should see a confirmation message that the page was unlocked.

Administering Page Locks

Developers may want to see all the pages that are locked, or they may want to lock/unlock multiple pages at the same time. APEX provides tools to handle multiple page lock requests. To view the Page Locks report go to Utilities ► Cross Page Utilities ► Page Locks as shown in **Figure 13-3**.

Status	Page	Name	Locked By	Date Locked	Page Updated	Lock Comment	View Lock
	0	Page Zero	-		6 months ago		
	1	Sample Application	-		6 months ago		
	2	Customers	GIFFY	12/06/2010 01:39:55 PM	6 months ago	BUG-45	
	3	Products	GIFFY	12/06/2010 01:40:16 PM	6 months ago	BUG-34	
	4	Orders	-		6 months ago		

Figure 13-3. Multiple page locks tool

From the Page Locks report you can view all the pages that are locked and unlocked. You can lock and unlock multiple pages from the Page Locks report. Workspace administrators can unlock any page, while developers can only unlock pages that they locked. Developers can also view the lock history for each page by clicking on the View Lock icon that looks like a magnifying glass.

■ **Note** Page locks are not maintained to the new version when an application is copied or exported. This means that any page locks and comment history are only relevant to the specific application in the workspace. If you copy an application into another workspace, nothing will be locked in that new copy.

Application and Page Groups

Developing large APEX applications may require you to group applications and pages. APEX allows you to declaratively group applications and pages within applications. Grouping pages and applications can help avoid the need to have strict application and page naming and numbering schemes.

Application Groups

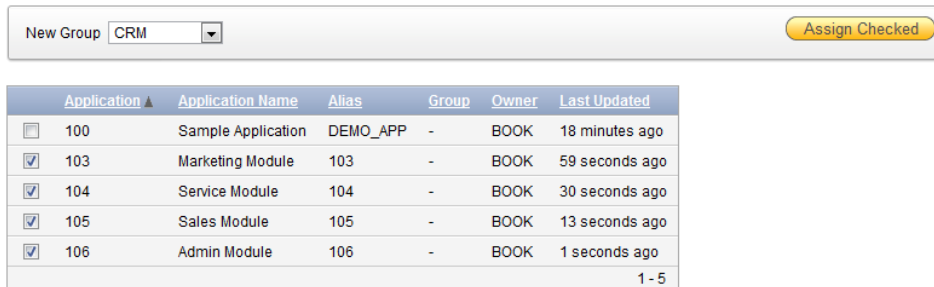
When you have multiple applications within a workspace, you may want to group associated applications to help developers visualize which applications are related. For example, suppose you develop a large CRM system that consists of three modules: Marketing, Service, and Sales. For various reasons you may want to create each of these modules in their own applications and have a common Admin module that links the applications together.

If your workspace contains other suites of applications, developers may get confused as to which suite of applications they should be working on. To resolve this issue, you can create application groups. Here is the process to create an application group:

1. Go to Home ► Application Builder.
2. Click on Application Groups on the right side under the Tasks region.
3. Click the Create button.
4. Enter a name and description for the group. In this example, you'll use CRM. Click the Create button.

The following steps demonstrate how to add individual applications to an application group:

1. In the Application Groups page, click on the Manage Unassigned link on the right side in the Tasks region.
2. Select the applications and the application group and click the Assigned Checked button, as shown in **Figure 13–4**.



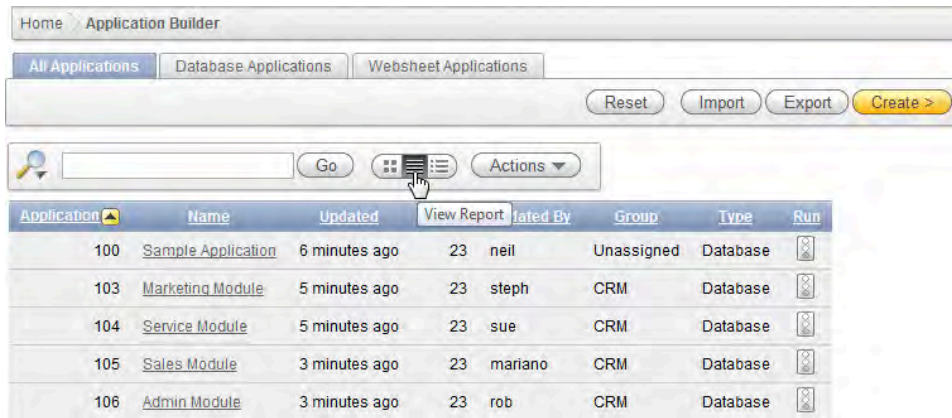
The screenshot shows a web interface for managing application groups. At the top, there is a 'New Group' section with a dropdown menu set to 'CRM' and an 'Assign Checked' button. Below this is a table listing applications with columns for Application, Application Name, Alias, Group, Owner, and Last Updated. The table contains five rows of data, with the first row (Application 100) having an unchecked checkbox and the subsequent four rows (Applications 103-106) having checked checkboxes. The bottom right of the table shows a pagination indicator '1 - 5'.

Application	Application Name	Alias	Group	Owner	Last Updated
<input type="checkbox"/> 100	Sample Application	DEMO_APP	-	BOOK	18 minutes ago
<input checked="" type="checkbox"/> 103	Marketing Module	103	-	BOOK	59 seconds ago
<input checked="" type="checkbox"/> 104	Service Module	104	-	BOOK	30 seconds ago
<input checked="" type="checkbox"/> 105	Sales Module	105	-	BOOK	13 seconds ago
<input checked="" type="checkbox"/> 106	Admin Module	106	-	BOOK	1 seconds ago

1 - 5

Figure 13–4. Assign application groups

3. Go to the Application Builder main page (Home ► Application Builder) and view the applications in a report format. You can view the list of applications along with their application group, as shown in **Figure 13–5**.



Application	Name	Updated	View Report	Created By	Group	Type	Run
100	Sample Application	6 minutes ago		23 neil	Unassigned	Database	
103	Marketing Module	5 minutes ago		23 steph	CRM	Database	
104	Service Module	5 minutes ago		23 sue	CRM	Database	
105	Sales Module	3 minutes ago		23 mariano	CRM	Database	
106	Admin Module	3 minutes ago		23 rob	CRM	Database	

Figure 13–5. Application report

Page Groups

Page groups are similar to application groups except that they group pages together. They are application specific, which means that a page group is only valid for a particular application. Page groups are very useful to help group common pages together within an application.

An alternate approach to using page groups is to use a numbering scheme to group pages together. The page number approach may not always work as you may run out of numbers. For example, imagine that you grouped logical pages in sets of 10. What happens when you have 11 pages in a group? Of course, a workaround is to create large intervals for each logical page group, but even so you may run into a situation where a page does not conform to your numbering standards.

■ **Note** You may also leverage page groups for purposes other than the development environment. Suppose you grouped all of the admin pages into an Admin page group. If you had a Page 0 region that you only wanted to appear on admin pages, you could put a condition that would check that the current page was associated to the Admin page group.

To create and manage page groups, go to Utilities ► Cross Page Utilities ► Page Groups. On the Page Groups page you can create and assign page groups similar to how you created applications groups in the previous section.

APEX Views and the APEX Dictionary

In traditional web development tools, if you need to search through all your code you must search through multiple text files. APEX is different because APEX stores code in the database. Thus, you can run queries to search through your code.

The APEX Schema

A common misconception about APEX is that it is an extra piece of software that you need to install. In fact, APEX is a framework that is stored in the APEX_040000 schema within the database. At a very high level, each time you request a page, APEX queries tables within its schema and executes many invocations of the *HTTP* procedure to produce the HTML that is sent to the browser.

Each time you create an object in APEX, it's stored in a table within the APEX schema. For example, when you create a page item, it's stored in the APEX_040000.WWV_FLOW_STEP_ITEMS table.

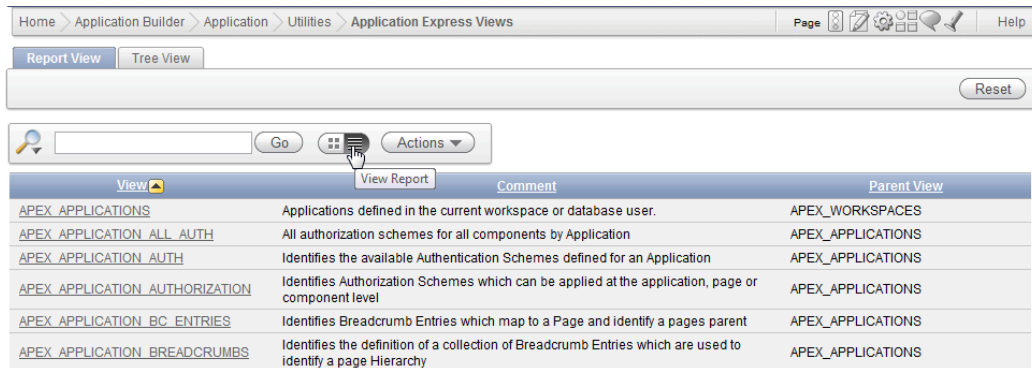
■ **Note** As mentioned in the first chapter, APEX was originally called *FLWS* which is why some of the table names contain “FLOW.”

Storing code within the database has its advantages and disadvantages. An advantage is that you can query the database to quickly find what you are looking for in an organized fashion. For example, you can easily search through all your reports for a certain table or column reference. A disadvantage is that it is harder to search through all the objects within an APEX application at the same time. Searching through an entire application is easier in file-based web applications since you can do a simple text search.

APEX Views

There are several ways to access the data from the APEX views. In this section, I'll discuss how to use these views to search within the APEX development environment.

To access the APEX views within the development environment go to Utilities ► Application Express Views. If you are new to APEX, I recommend that you change the default view mode to View Report, which provides a detailed description of each of the reports (see **Figure 13–6**).



Home > Application Builder > Application > Utilities > Application Express Views			
Page [Icons] Help			
Report View Tree View			
Reset			
[Magnifying Glass] [Go] [Actions]			
View	View Report	Comment	Parent View
APEX_APPLICATIONS		Applications defined in the current workspace or database user.	APEX_WORKSPACES
APEX_APPLICATION_ALL_AUTH		All authorization schemes for all components by Application	APEX_APPLICATIONS
APEX_APPLICATION_AUTH		Identifies the available Authentication Schemes defined for an Application	APEX_APPLICATIONS
APEX_APPLICATION_AUTHORIZATION		Identifies Authorization Schemes which can be applied at the application, page or component level	APEX_APPLICATIONS
APEX_APPLICATION_BC_ENTRIES		Identifies Breadcrumb Entries which map to a Page and identify a pages parent	APEX_APPLICATIONS
APEX_APPLICATION_BREADCRUMBS		Identifies the definition of a collection of Breadcrumb Entries which are used to identify a page Hierarchy	APEX_APPLICATIONS

Figure 13–6. APEX views

The following example demonstrates how the APEX views can assist you. Suppose you need to compare the help text for items which reference *PRODUCT_ID*. Follow these steps to compare the help text for these items:

- 1. Go to Utilities ► Application Express Views.
- 2. Click on the APEX_APPLICATION_PAGE_ITEMS view. You may need to search for this view in the Interactive Report or paginate to the next page.
- 3. On the Select Columns screen, add ITEM_NAME and ITEM_HELP_TEXT to the selected column and click the Filter button, as shown in Figure 13–7.

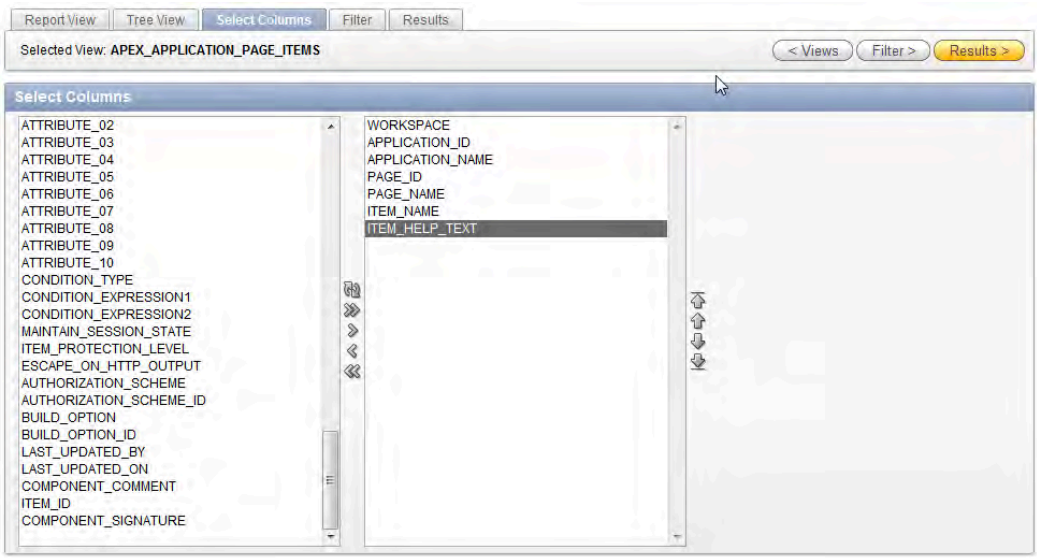


Figure 13–7. Select columns view

- 4. On the filter page, select/enter ITEM_NAME – LIKE – '%PRODUCT_ID%' and click the Results button, as shown in Figure 13–8. It is important to include the single quote characters around your search value just as you would in a SQL query.

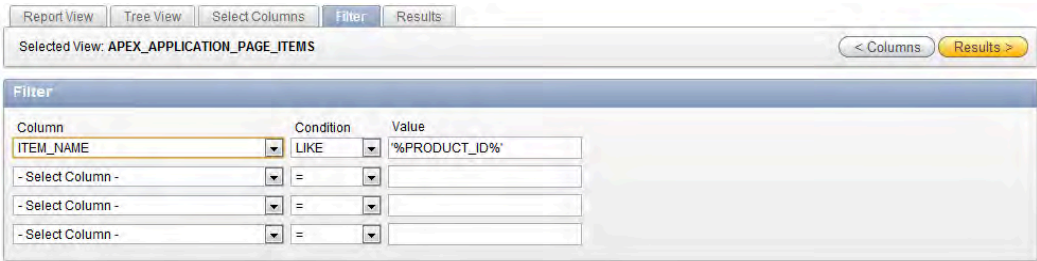


Figure 13–8. Filter view

5. You should now see the list of all page items within the workspace that have `PRODUCT_ID` as part of the item name. Based on these results, you can modify the items that need to be changed. Since the report shows all applications within the workspace, you may want to apply an additional filter for a specific application.

Alternatively, you can view the list of APEX views as a tree by clicking the Tree View tab shown in **Figure 13–9**. Once you click on your desired view, you can continue with the process described above to display the results.

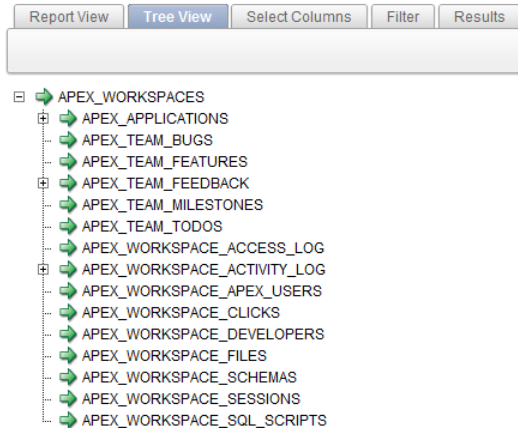


Figure 13–9. Tree view

APEX Dictionary

Since all the data resides in the database, you can reference the APEX views from SQL queries. As you get more familiar with the APEX views, you may prefer to query them using SQL since you can quickly apply predicates and you do not need to use a web browser to run your reports.

APEX provides a view that lists all the APEX views called `APEX_DICTIONARY`. The APEX Dictionary contains all the information that is available in the developer GUI as well as descriptions for each of the columns. The following query lists all the APEX views:

```
SELECT *
  FROM apex_dictionary
 WHERE column_id = 0
```

There is a slight difference when querying the APEX views in SQL compared to the GUI. When using the developer GUI, only applications that reside within the workspace will appear in results. When querying through SQL, only applications whose parsing schema is the same as your connection will appear in results. If you connect as `SYS` or `SYSTEM`, you can view all the applications regardless of the application's parsing schema.

In the previous section, the example described how to look at all the help text for items with `PRODUCT_ID` in its name. You can view the same set of results by running the following query:

```
SELECT workspace,
       application_id,
       application_name,
```

```

    page_id,
    page_name,
    item_name,
    item_help_text
FROM apex_application_page_items
WHERE item_name LIKE '%PRODUCT_ID%'

```

Searching is not the only use for the APEX views. You may need to leverage the APEX views when creating plug-ins or adding advanced features to your application.

Searching within APEX

In the previous section, you examined how to use the APEX views to search for items within your application. In this section, you'll learn alternative ways to search through your application.

APEX Finder

APEX provides a tool within the application that provides some reports that leverage the APEX views on common APEX objects. To access the APEX finder, click on the Find icon (which looks like a flashlight and is located in the top right corner of any page in the Application Builder) as shown in **Figure 13–10**.

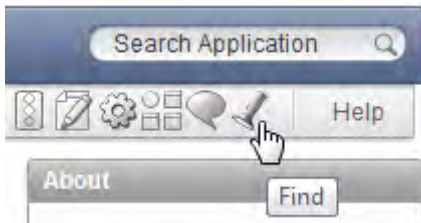


Figure 13–10. Find icon

Clicking on the Find icon will result in a pop-up window that contains interactive reports for the following object types:

- Application and page items
- Pages
- SQL Queries from report regions
- Database tables in the parsing schema
- PL/SQL packages, functions, and procedures in the parsing schema
- Images, including standard images, workspace images, and application images
- A list of debug log entries
- Application items, page items, and collections in the current session
- A list of errors that have occurred in this application at both region and page level

The APEX finder is helpful since it allows you quickly search for something while you're in the application builder and doesn't require you to leave the page. If you need to do more complex searches or filters, I recommend querying the APEX views.

Search Application

The APEX views GUI, queries against the APEX views, and the APEX finder are good tools if you know exactly what you're searching for within an object type. For example, if you want to see if a particular table was referenced in a query, then you can search the APEX_APPLICATION_PAGE_REGIONS view for the table name in the REGION_SOURCE column.

What if you wanted to search an entire application to see whether it references a specific table? Suppose, for example, that you rename the EMP table to EMPLOYEES. How can you easily search through your entire application for any reference to EMP? The answer is to use a new feature added in APEX 4.0 called Search Application. Search Application searches through the entire application.

To search through the entire application, enter your search criteria in the Search Application field (located in the top right corner—see **Figure 13–10**). **Figure 13–11** shows the detailed results of all occurrences of EMP within the application. For each result, a link is provided to the exact location of the result. Using the Search Application feature, you can easily find all references of EMP and replace them with EMPLOYEE.

Home > Application Builder > Application 108 > Search

Search

☐ Case sensitive
☐ Current page only

Applications > 108 - Demo > Pages > 1 - Employee

Attribute	Page Name (Identifies a page within an application)
Value	Employee
	View

Attribute	Page Title (Used by page templates to display the HTML title tag)
Value	Employee
	View

Applications > 108 - Demo > Pages > 1 - Employee > Regions > Employee

Attribute	Region Name (Identifies the Region Name. The display of the region name is controlled by the Region Template substitution string TITLE.)
Value	Employee
	View

Figure 13–11. APEX Search Application results

The problem with the results in **Figure 13–1** is that they contain results for all text that contains EMP. This includes *template*, *employee*, *empno*, etc. Since you only want references to the EMP table with respect to SQL queries and PL/SQL blocks, you can exclude all occurrences of EMP where its previous or next character is alphanumeric.

The APEX search application tool supports regular expressions. To use regular expressions the search criteria must be prefixed with “regexp:”. **Figure 13–12** shows the search application results when a regular expression is used to filter out occurrences of EMP where its previous or next character is alphanumeric.

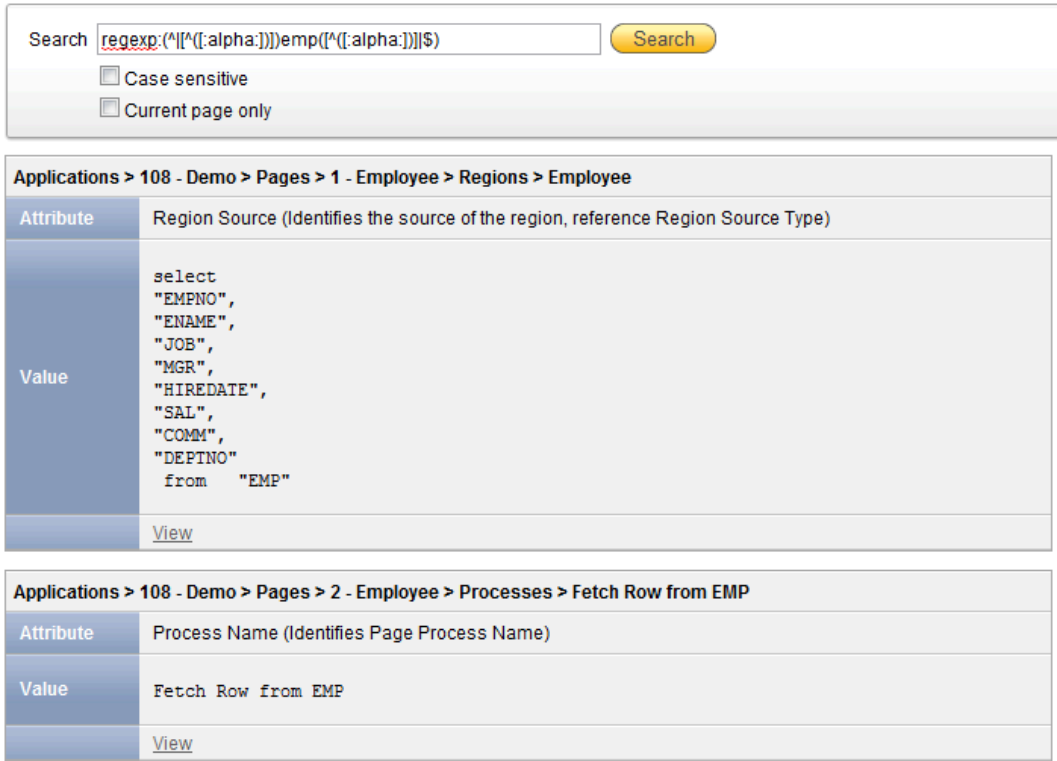


Figure 13–12. APEX search results using regular expressions

Monitoring your APEX Application

APEX can log each page access and login attempt. Logging is an excellent feature to enable because it allows you to monitor your application and provides a way to help reduce errors and improve performance. This section will show you how to enable logging, some uses for the activity log, and how to view all login attempts.

Enabling Logging

By default, logging is enabled when you create an application. To verify that logging is enabled for your application, go to Shared Components and click the Edit Definition link in the Application region on the right side. Under the properties section there will be a logging option, as shown in **Figure 13–13**. Ensure it says Yes and click the Apply Changes button.

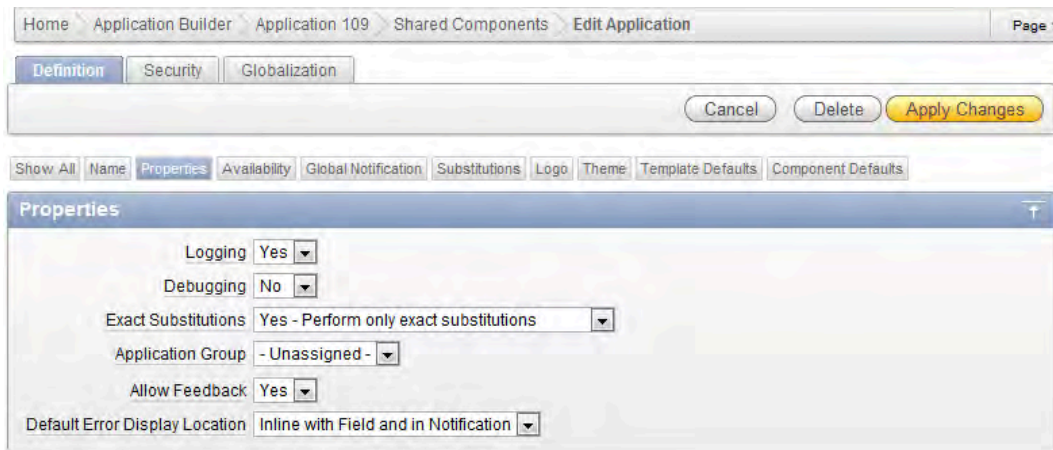


Figure 13–13. Enable logging

For applications that have many page hits, you may want to disable logging as it can slow the application down. Most applications will not have this issue, but it is important to know that the problem might occur.

■ **Note** Logs are stored in two separate tables (views join these tables together to make it appear as one table). Every two weeks one of these log tables is truncated. This means that the log views contain two to four weeks of data. It is recommended that you set up a nightly job to permanently retain this data in your own schemas. An example of storing a local, permanent, log history is shown in the following blog post at www.talkapex.com/2009/05/apex-logs-storing-log-data.html.

Using the Activity Logs

Each time a page is accessed a log entry is stored. It can be referenced from the `APEX_WORKSPACE_ACTIVITY_LOG` view. A good example of how to mine the activity log is to search for errors in your application. No matter how hard you try, unhandled errors occur. Instead of waiting for users to report these errors (assuming that they even report errors), you can take a proactive approach. The following query will identify when an error occurs at the page or region level:

```
SELECT *
  FROM apex_workspace_activity_log
 WHERE error_message IS NOT NULL
```

Once an application has been running for a while you may notice that some pages are accessed more often than others and some pages aren't performing as desired. The following queries identify these two cases:

```
-- Find most accessed pages
SELECT application_id,
```

```

        application_name,
        page_id,
        page_name,
        SUM (page_id) page_hit_count
FROM apex_workspace_activity_log
GROUP BY application_id,
         application_name,
         page_id,
         page_name
ORDER BY SUM (page_id) DESC

-- Find slowest pages
-- Note: This depends on how you calculate slow
SELECT application_id,
       application_name,
       page_id,
       page_name,
       ROUND (AVG (elapsed time), 5) avg_elapsed_time,
       SUM (page_id) page_hit_count,
       MEDIAN (elapsed time) median_elapsed_time
FROM apex_workspace_activity_log
GROUP BY application_id,
         application_name,
         page_id,
         page_name
ORDER BY 5 DESC

```

By identifying the top accessed pages, you can focus your attention on trying to speed them up. Slow pages may require tuning, but if they are accessed infrequently, you may not need to spend a lot of time on them.

Here are some other examples of uses for the activity log:

- **Top browsers:** If you build your application to support Firefox and IE and find that half of your users are using Chrome you may want to invest some time ensuring that your application supports Chrome.
- **The time frame when people are using your application.** This will provide you with an idea for the best time for maintenance and upgrades. You can also derive the peak usage times.
- **Search criteria in interactive reports.** If there is a consistent search pattern, perhaps a better report is required or preset filters.

Login Attempts

The APEX_WORKSPACE_ACCESS_LOG stores all the login attempts to your APEX applications. The access log can be extremely useful when debugging user authentication issues.

An example of utilizing the access log is to monitor invalid login attempts. When a user attempts to log in with invalid credentials, it's not recommended to display the exact reason why their login attempt failed. You do not want to tell the user the exact reason because it could reveal some valuable information, such as if the user exists, etc. It may still be important for your operations team to know why a user was not able to log in, in case they need to resolve the issue. Since all login attempts are stored in the Access Log you can see the exact reason why a user was not able to log in for failed login attempts.

■ **Note** If you create your own authentication process, you should leverage the `APEX_UTIL.SET_AUTHENTICATION_RESULT` and `APEX_UTIL.SET_CUSTOM_AUTH_STATUS` procedures to ensure that you populate the access log with meaningful messages. For more information on these authentication procedures, please read the APEX API documentation.

APEX Advisor

The APEX Advisor is a tool that executes pre-defined checks against your application. These validations can help reduce errors in your application before it is tested or goes into production.

■ **Note** Prior to APEX 4.0, the APEX Advisor was an open source project developed by Patrick Wolf. During the development of APEX 4.0, Patrick joined the APEX team and included the Advisor as a built-in tool. The open source version of the APEX Advisor is available at <http://essentials.oracleapex.info/>

To use the Advisor, go to Utilities ► Advisor. **Figure 13–14** shows all the checks that can be performed. Hovering over each validation will display a brief description of the validation. You have the option to restrict the pages that the Advisor reviews by defining a comma-delimited list of pages to search for in the Check Pages region located at the bottom of the page. Once you select the checks to perform, click the Perform Check button. The results page will provide a list of detailed issues that the Advisor finds plus links to each of the objects.

Checks to Perform

<p>Errors:</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> References with Substitution Syntax <input checked="" type="checkbox"/> References with Column Syntax <input checked="" type="checkbox"/> References with Bind Variable Syntax <input checked="" type="checkbox"/> Declarative References of Application, Page Items, Interactive Report Filters <input checked="" type="checkbox"/> Referenced Page Number Exists <input checked="" type="checkbox"/> Is Valid SQL or PL/SQL Code <input checked="" type="checkbox"/> Fetch, DML, MR* Processes are Valid <input checked="" type="checkbox"/> Unconditional Branch before other Branches <input checked="" type="checkbox"/> Referenced Button in When Button Pressed exists <p>Security:</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Target Page Authorization equals current Component Authorization <input checked="" type="checkbox"/> Inappropriate use of Substitution Syntax <p>Warnings:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Referenced Item is on Current Page <input checked="" type="checkbox"/> Referenced Item is Page Item of Target Page <input checked="" type="checkbox"/> References of Page Item in a String <input checked="" type="checkbox"/> Clear Cache Page Number equals Target or Current Page <input checked="" type="checkbox"/> Length of Item Name 	<p>Performance:</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> V Function used in SQL Statements <p>Usability:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Target Page Authorization is also set for Current Component <input checked="" type="checkbox"/> Associated Item of Validations <p>Quality Assurance:</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Hardcoded Application ID <input checked="" type="checkbox"/> Report has Default Order <input checked="" type="checkbox"/> Page Item has Help Text
--	---

Select All Deselect All

Figure 13–14. APEX Advisor options

The Advisor is an excellent tool to help detect issues before your application is deployed to end users. It is still important to have development standards and a release process to help prevent issues. You should be aware that the Advisor may produce false positives in response to some of the business rules within your organization, so you should analyze each suggestion before fixing it.

Build Options

Build options allow you to conditionally render and execute objects in APEX. Build options are either enabled or disabled for the entire application and can only be changed within the Application Builder. This means that they are not run-time configuration options.

Understanding the Need

Suppose you are working on a custom authentication scheme and you want to verify that the appropriate authentication results and custom status messages are populating in the Activity log for invalid login attempts. Each time you attempt a login you could switch programs and run a query against `APEX_WORKSPACE_ACCESS_LOG`. This process may get cumbersome because you have to toggle between two applications. An alternate solution is to create a report on the Login page to display the most recent login attempts. After each bad login attempt you'll return to the Login page and see an updated report of the Access log report. To build this report on page 101, your Login page, create a report with the following query:

```
SELECT user_name,
```

```

        authentication_method,
        access_date,
        authentication_result,
        custom_status_text
    FROM apex workspace access_log
    WHERE application id = :app_id
    ORDER BY access_date DESC

```

The Login page will now look like Figure 13–15. The report allows you to quickly see the authentication results when testing the login process.

Username	Authentication Method	Access Date	Authentication Result	Custom Status
STEPHANIE	FND_ACCOUNT	11-OCT-10	AUTH_UNKNOWN_USER	-
MARTIN	FND_ACCOUNT	11-OCT-10	AUTH_UNKNOWN_USER	-
STEPHANIE	FND_ACCOUNT	11-OCT-10	AUTH_UNKNOWN_USER	-
GIFFY	FND_ACCOUNT	04-OCT-10	AUTH_UNKNOWN_USER	-

Figure 13–15. Access log

Once you are content that your custom authentication scheme is working, you would normally delete the extra report region since it is only there for debugging purposes. Deleting the region is counterproductive, however, since you may need it in the future when you modify the authentication scheme.

Creating a Build Option

Instead of removing the region, you should tag it as a *Development Only* object so that it will only be available while you are developing your application rather than running it in production. Follow these steps to create a build option to support this requirement:

1. In the Application Builder, go to Shared Components ► Build Options and click the Create button.
2. Fill in each section as shown in **Figure 13–16** and click the Create button.

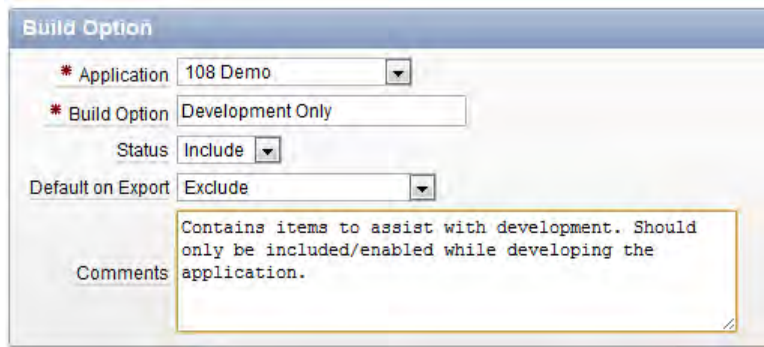


Figure 13-16. Create build option

3. You should now see a Development Only build option, as shown in Figure 13-17.

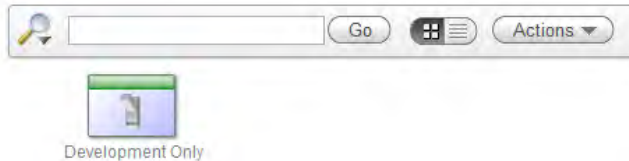


Figure 13-17. Development Only build option

Configuring Build Options

Before I continue with this example, it's important to review some of the options in Figure 13-16. The status values of *include* or *exclude* can be misleading. Build options don't affect what is included in the application, just what is executed or displayed at run time. A better description of the status options would be *enable/disable* or *on/off*.

The Default on Export option sets the default configuration of the build option when the application is exported and then imported. For your example, since you're using the build option to handle Development Only features, it makes sense to always exclude the build option and require developers to explicitly include it.

In some cases there's no clear default option, so the person installing the application must choose the appropriate build option status. You can configure a required choice as part of the application installation script.

Prompting for Build Option Status

To configure the application to prompt for a build option status during installation go to Supporting Objects ► Build Options in the Application Builder. Select Development Only (include) under Prompt for Build Options to prompt for as part of the installation (see Figure 13-18) and click the Apply Changes button.

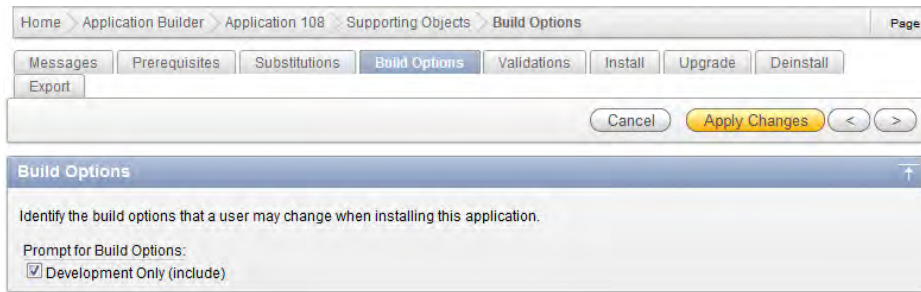


Figure 13-18. Build option installation configuration

When installing the application, the user will have the option to include the build option as shown in **Figure 13-19**. Again, *include* is a misleading term as the build option will be included. If you include it, it will be enabled.

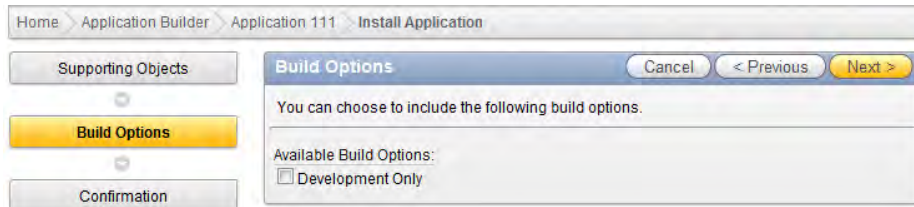


Figure 13-19. Build option prompt

Applying Build Options

Now that you have created and configured the Development Only build option, you need to apply it to the region in question—the Access Log region on page 101. The following steps will apply the Development Only build option to the Access Log region:

1. Edit the Access Log region.
2. Scroll down to the Configuration region.
3. Select Development Only in the Build Option list (see **Figure 13-20**) and click the Apply Changes button.

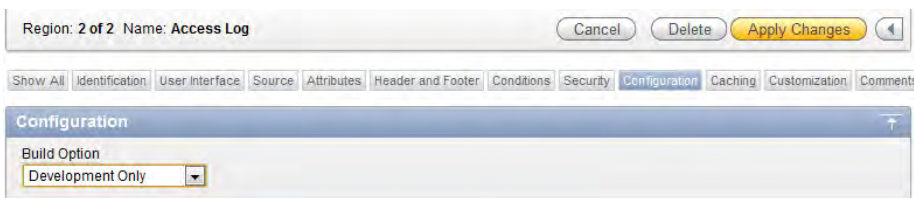


Figure 13-20. Apply build option

■ **Note** When applying build options you can select the opposite result by selecting the “{Not ...}” option. This helps avoid having to create two build options which will always be the reciprocal of each other.

The Access Log region will now only be run when the Development Only build option status is set to include. You can apply build options to other APEX objects using the same process.

Reporting on Build Option Utilization

Build options can be applied to most APEX objects including pages, regions, page items, tabs, etc. It can become difficult to keep track of which objects use which build options in applications. The build option Utilization report enables you to easily view which objects are using a particular build option. This report can be very helpful when trying to get an overview of the impact that a build option has on the application.

To view the build option Utilization report go to Shared Components ► Build Options. Click on the Utilization tab and select the Development Only build option as shown in **Figure 13–21**. The link in the description column will bring you to the specific object that is using the build option.



Figure 13–21. Build option utilization report

Page Specific Utilities

Page specific utilities allow developers to do bulk operations on APEX objects within an application. To access page specific utilities go to Utilities. The Page Specific Utilities region, located on the right, contains all the page specific utilities available (see **Figure 13–22**).

Page Specific Utilities
Cross Page Utilities
Region Utilities
Button Utilities
Item Utilities
Computation Utilities
Validation Utilities
Process Utilities
Dynamic Action Utilities
Branch Utilities

Figure 13–22. Page specific utilities

Each of the utilities provides tools associated to the type of object. I won't cover each utility in this book, but I encourage you to explore the available features.

APEX and Oracle SQL Developer

Oracle SQL Developer is a free database development GUI. For more information about SQL Developer go to <http://technet.oracle.com/sqldeveloper>.

Integration

APEX is integrated with SQL Developer. In SQL Developer, you can see all the APEX applications whose parsing schema is the same as your connecting schema (see **Figure 13–23**).

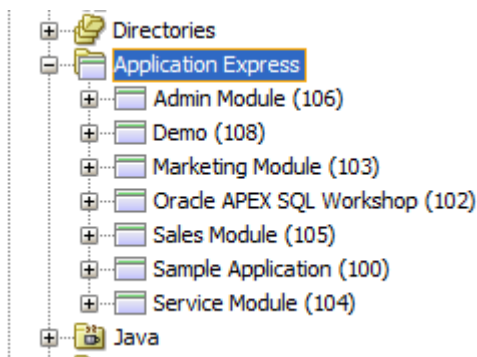


Figure 13–23. APEX in SQL Developer

Within SQL Developer you can view object information and perform basic application level tasks such as importing and exporting an application, changing the application alias, and renaming an application.

Refactoring Support

APEX allows for anonymous blocks of PL/SQL. I strongly recommend that such blocks reference compiled code (packages, functions, and procedures). Storing code in packages helps separate the business logic from the display layer and may have some performance benefits.

In some cases, you may have written large blocks of code directly in the application. Eventually, you'll want to move these large blocks of code into compiled PL/SQL code. SQL Developer provides a tool that automatically generates a PL/SQL package from the anonymous blocks of PL/SQL in APEX. You can then replace your large blocks of code with references to this package. To generate the package, right click on the application in SQL Developer and select Refactor (in bulk), as shown in **Figure 13–24**.

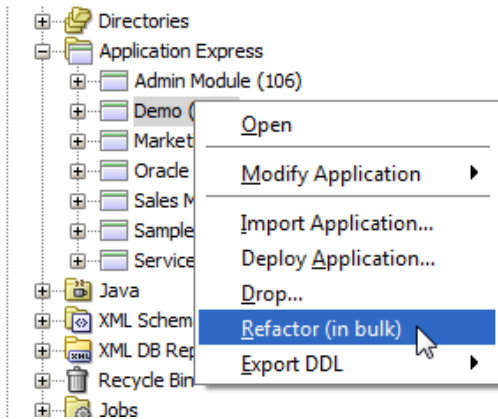


Figure 13–24. Refactor code

SQL Developer will open a new worksheet with the necessary code to compile. The new worksheet also provides notes on what sections of your APEX applications to change and the code to replace it with. Similar to the APEX Advisor, these are recommendations; you should follow your organization's development standards, etc.

Summary

There are many tools within APEX that make your life as a developer easier. Take some time to get to know the tools and utilities presented here and you will undoubtedly speed up your ability to get things done. And while any PL/SQL GUI will help you edit and manage database objects, it should be clear by now that Oracle's SQL-Developer has special hooks to make managing, developing and debugging with APEX far easier.



Managing Workspaces

Once you start developing APEX applications and working in a team environment, you'll need to spend some time managing workspaces. This chapter will cover the various tools and resources available to manage a workspace.

■ **Note** Administering a workspace covers many different areas. This chapter will not cover all areas in detail; some have been mentioned in other chapters and others will not be included due to space constraints. This chapter assumes that the user logging in is a workspace administrator.

Learning About Your Environment

When you log into APEX as a workspace administrator and click on the Administration tab at the top of the page, you will see icons for each of the major sections (see Figure 14–1). This chapter will cover each of these sections and their associated subsections. First, though, I'll show you how to get information about your APEX environment.

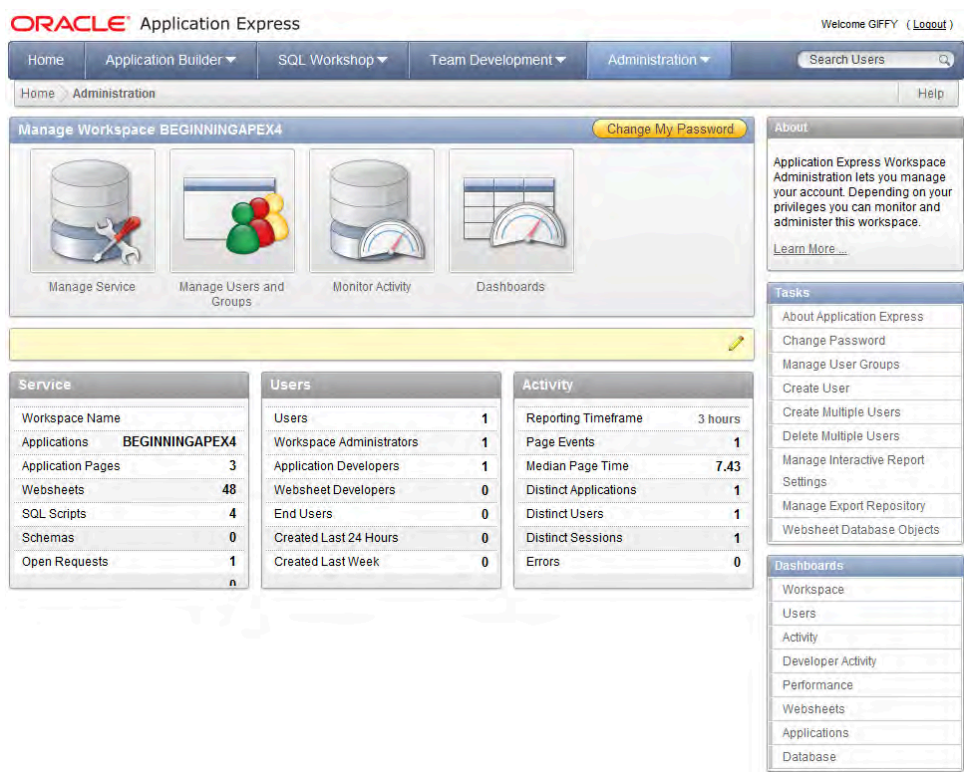


Figure 14–1. Administration home page


Viewing Instance Information

On the Administration home page on the right hand side under the Tasks region, the first a link is called About Application Express. Clicking on the About Application Express link will bring you to a page that shows information about your current APEX instance (see Figure 14–2).

Home Administration About Application Express

Show All About Visit Oracle Online For More Information CGI Environment

About



Product Build:	4.0.1.00.03
Schema Compatibility:	2010.05.13
Last DDL Time:	01/31/2006 11:34:33 PM
Last DDL Time:	08/29/2010 12:27:50 PM
Last DDL Time:	08/29/2010 12:54:41 PM
Host Schema:	ANONYMOUS
Application Owner:	APEX_040000
Workspace ID:	2377712967093271
Workspace Name:	BOOK
Current User:	ADMIN
Language Preference:	en
Current Time (on server):	11/14/2010 08:28:40 AM

Visit Oracle Online For More Information

- Oracle Technology Network
- Oracle Technology Network Discussion Forum

CGI Environment

```

APEX_LISTENER_VERSION
PLSQL_GATEWAY      WebDb
GATEWAY_IVERSION    2
SERVER_SOFTWARE     Oracle Embedded PL/SQL Gateway/10.2.0.1.0
GATEWAY_INTERFACE   CGI/1.1
SERVER_PORT         8080
SERVER_NAME         XDB HTTP Server
REQUEST_METHOD       GET
QUERY_STRING        p=4350:9:2170035190810725::NO:::
PATH_INFO           /f
SCRIPT_NAME         /apex
REMOTE_ADDR         192.168.56.1
SERVER_PROTOCOL     HTTP/1.1
REQUEST_PROTOCOL     tcp
REMOTE_USER         ANONYMOUS
HTTP_USER_AGENT      Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/534.7 (KHTML, like Gecko) Chrome/7.0.517.44 Safari/534.7
HTTP_HOST           fcapex40:8080
HTTP_ACCEPT_ENCODING gzip
HTTP_ACCEPT_LANGUAGE en-US
HTTP_ORACLE_ECID
WEB_AUTHENT_PREFIX
DAD_NAME            apex
DOC_ACCESS_PATH      docs
DOCUMENT_TABLE       www_flow_file_objects$
PATH_ALIAS
REQUEST_CHARSET      AL32UTF8
REQUEST_IANA_CHARSET UTF-8
SCRIPT_PREFIX
  
```

Figure 14–2. About Application Express page

The information on this page can be very useful when developing and debugging APEX applications. In fact, the About page is an excellent location from which to quickly get an overview of your particular APEX instance. Since APEX is stored in the database, all of the values in the figure can also be obtained from a query. For example, you can find the database version from the following query:

```
SELECT *
FROM v$instance
```

The list of CGI variables at the bottom of Figure 14–2 is only available to workspace administrators. If developers need to see a complete list of CGI variables and their values, they can create a PL/SQL region and use `OWA_UTIL.PRINT_CGI_ENV` as the region source.

■ **Note** You can only obtain CGI variables from a web interface since they are web based variables. If developers try to query them in SQL*Plus, they won't get any results. The following article contains more information on CGI variables: http://en.wikipedia.org/wiki/Common_Gateway_Interface

Checking the APEX Version

The About Application Express page displays the current version of APEX. The version can also be obtained by the following query:

```
SELECT *
FROM apex_release
```

Obtaining the APEX version from a query can be important for DBAs when upgrading an application. Also, developers sometimes write code that is dependent on the current version of APEX.

Managing the Service

Click the Manage Service icon to change preferences and other settings affecting the operation of your APEX service. You'll find the Manage Service icon on the Administration home page shown in Figure 14–1.

Figure 14–3 shows the main Manage Service page.

The screenshot displays the 'Manage Service' page. At the top, a breadcrumb trail shows 'Home > Administration > Manage Service'. Below this, a 'Manage Service' header is followed by four icons with labels: 'Make a Service Request' (database with plus), 'Set Workspace Preferences' (clipboard with pencil), 'Edit Announcement' (megaphone), and 'Workspace Utilization' (database with gauge). A yellow bar with a pencil icon is below these. The main content area is divided into two sections: 'Service' and 'Workspace Schemas'. The 'Service' section contains a table with the following data:

Service	
Workspace Name	BEGINNINGAPEX4
Applications	1
Worksheets	2
Schemas	1
SQL Scripts	0
Pages	23
Open Requests	0

The 'Workspace Schemas' section shows 'BEGINNINGAPEX4'. On the right, a sidebar contains 'Manage Services' (with a description of its capabilities), 'Manage Meta Data' (with links like 'Developer Activity and Click Count Logs'), and 'Dashboards' (with links like 'Workspace' and 'Users').

Figure 14–3. Manage Service page

Workspace Preferences

You can enable and disabled the different modules available to APEX developers. To configure each of the modules, click on the Set Workspace Preferences icon that looks like a clipboard (see Figure 14–3). The Set Workspace Preferences page is shown in Figure 14–4.

Home > Administration > Manage Service > Set Workspace Preferences

Cancel Apply Changes

Show All Account Login Control Application Builder SQL Workshop Team Development

Account Login Control

Account Expiration and Locking: ☐ Enable ☒ Disable

Maximum Login Failures Allowed: 10

End User Account Lifetime (days): 999999

Application Builder

Enable Application Builder Yes

SQL Workshop

Enable SQL Workshop Yes

PL/SQL Editing: ☒ Allow PL/SQL program unit editing ☐ Do not allow PL/SQL program unit editing

Team Development

Enable Team Development Yes

Figure 14–4. Set Workspace Preferences page

From the Set Workspace Preferences page, you can enable or disable the following APEX modules:

- Application Builder
- SQL Workshop
- Team Development

If you disable any of these modules, they will be disabled for all users regardless of their privilege.

In production instances you may want to restrict access to the SQL Workshop as part of your corporate policy because the SQL Workshop has access to all the objects in the schema.

The Account Login Control section manages APEX workspace users. Settings in that section do not affect your APEX applications unless you are using the default APEX authentication scheme that references workspace users.

Announcements

Announcements allow workspace administrators to display messages to APEX developers. Announcements will be displayed in key areas throughout the APEX development environment. These messages will not be displayed in your APEX applications.

To create an announcement, click on the Edit Announcement icon that looks like a speaker (see Figure 14–3). Enter a message and click the Apply Changes button, as shown in Figure 14–5. The message will appear in the announcement section, as shown in Figure 14–6.

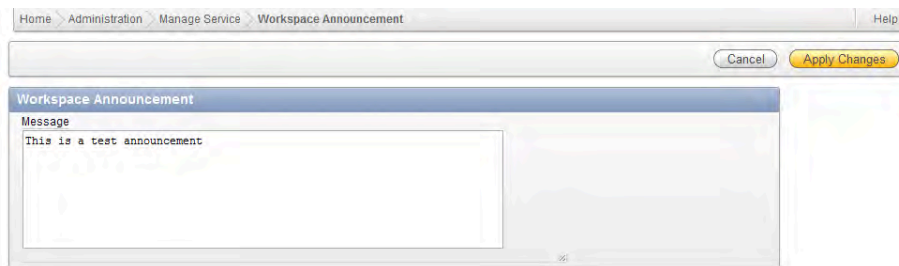


Figure 14–5. Editing a workspace announcement

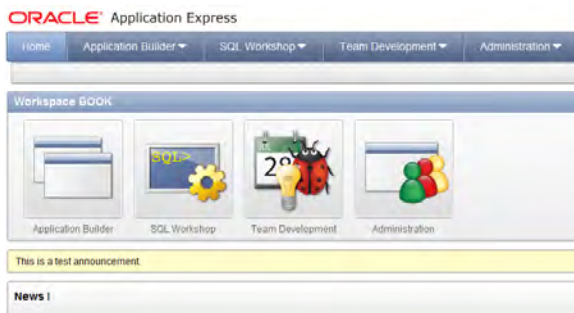


Figure 14–6. A workspace announcement

You'll notice in Figure 14–6 that there is also a News section. The News section is managed as part of Team Development; I'll cover that section in the next chapter. News is slightly different from announcements. Announcements can only be modified by workspace administrators and can only contain one message. News items can be set by users that have access to Team Development and can contain multiple messages.

Managing Meta Data

On the right hand side of the Manage Service page is a region called Manage Meta Data (see Figure 14–7). The following subsections will cover what you can do using the menu choices in the region.

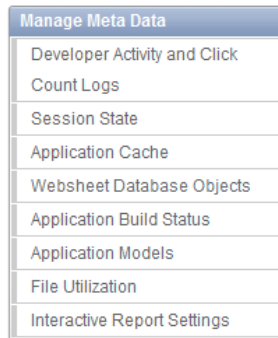


Figure 14–7. Manage Meta Data menu

Developer Activity and Click Count Logs

The Developer Activity and Click Count Logs menu option accesses two types of logs. One logs changes by developers in the workspace. The other tracks user clicks on links to pages outside your APEX application.

Developer Activity Logs

The Application Builder logs all changes made by developers. The logs are referenced in various locations throughout the Application Builder and the Administration section. You can view the logs from the Monitor Activity section, which you can access by clicking the Monitor Activity icon shown in Figure 14–1. You can also obtain the developer activity logs from the following query:

```
SELECT *
FROM apex_developer_activity_log
```

Like the logs mentioned in the previous chapter, the developer activity logs retain up to one month of data. You can purge developer activity logs by clicking on the Purge Developer Log button shown in Figure 14–8.

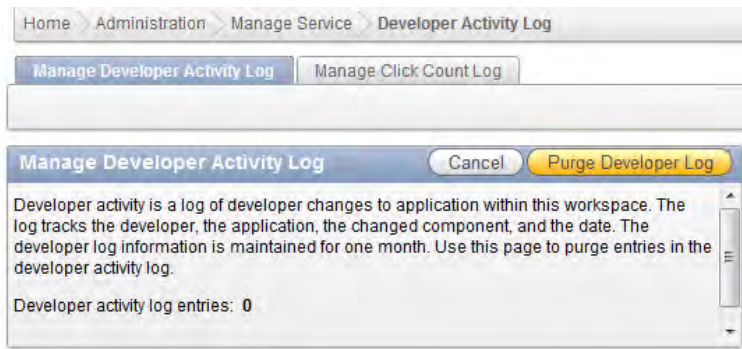


Figure 14–8. Purge Developer Log button

Click Count Logs

When creating a list (Application Builder ► Shared Components ► Lists), you can choose to track clicks on external links. These clicks are logged in the `APEX_WORKSPACE_CLICKS` log. You can also view the click count log in Monitor Activity section (see the icon for Monitor Activity in Figure 14–1).

Similar to the Developer Activity logs, you can purge the click logs by selecting the Manage Click Count Log and clicking on the Purge Click Log button shown in Figure 14–9.

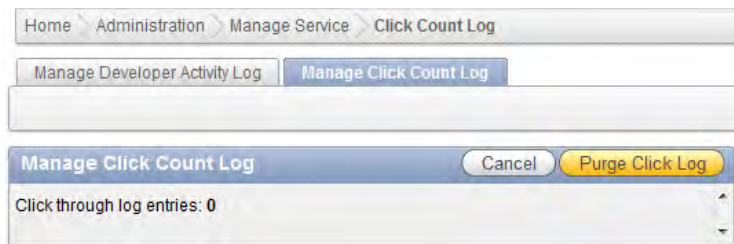


Figure 14–9. Purge Click Log button

Session State

Choosing Session State from the menu in Figure 14–5 brings you to a page that lets you manage session state and preferences. Preferences are slightly different than session state since they are linked to the user and not the current session. This means that changes to a preference will affect all sessions that a user is currently in.

Manage Session State

You can view all the session values, clear these values, and end sessions from the various reports available in the Manage Session State region. A session is automatically created each time a user logs into APEX. Since the APEX builder is an APEX application, sessions are created for both developers and end users.

Clearing session state or terminating a session may be useful in development to simulate different situations. If you are doing this in a production environment with live users, you should be extremely cautious.

Manage Preferences

The Manage Preferences region contains links to view and manage user preferences. Preferences as used to permanently store values for each user. Since they are linked to a user, they are session independent. One of the most common uses for user preferences is to store the sort order for standard reports that APEX automatically manages.

Similar to session states, you should not purge preferences in a production environment unless you are certain about what you're doing.

Application Cache

The Application Cache section provides the tools to purge page and region caches based on different criteria. In APEX you can cache pages and regions for all users or by each individual user. Caching can help improve performance since APEX does not need to regenerate the HTML code for a given region.

Websheet Database Objects

The Websheet Database Objects section allows you to create or delete the tables required for websheets and validate their status. Unlike an APEX application, websheet data is stored in tables that reside in each schema instead of in tables owned by the APEX_040000 schema.

When you first create a workspace, the websheet objects are not created. To create them, go to the Websheet Database Objects section and click the Create Websheet Database Objects link shown in Figure 14–10. Follow the wizard to create the websheet objects. After you have created the websheet objects, you can create a websheet application.

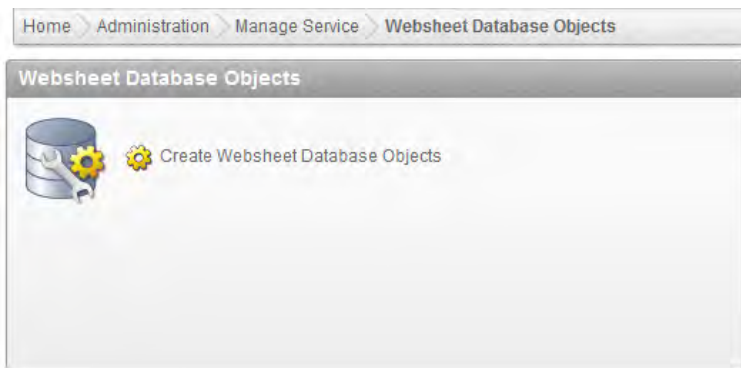


Figure 14–10. Create Websheet Database Objects page

Once the objects are created, the Websheet Database Objects page will look like Figure 14–11. From this screen, you can remove the database objects that relate to websheets or ensure that these objects are valid.

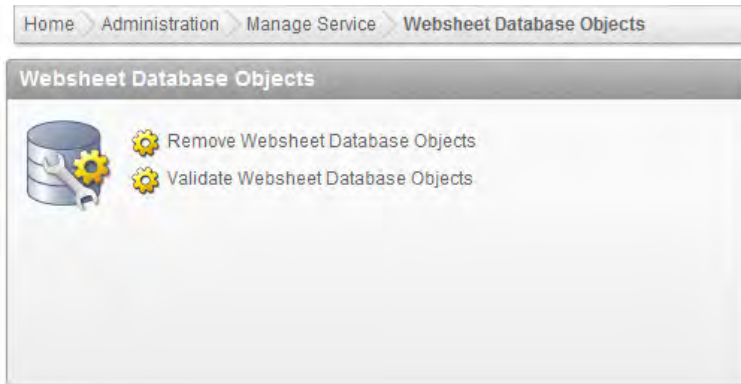


Figure 14–11. Websheet Database Objects page

Application Build Status

From the Application Build Status section, you can quickly manage both the application status and build status for all the applications within the workspace. An application has two main statuses: application status and build status. The application status controls the availability of the application. To get a full explanation of each application status, click the Help link in the top right corner. The build status determines whether an application can be modified by developers or run only. In production environments you may want to set the build status to Run Application Only to prevent any changes.

■ **Note** The application and build statuses can also be set from the application properties page for each application.

Application Models

The Application Models page provides a report of all the saved application models and allows you to delete them. Before reviewing the application models section, it's important to know what application models are.

In the last step of the create application wizard, you are given the option to save the definition as a design model (see Figure 14–12). Design models contain all the initial pages that you created as part of the create application wizard.

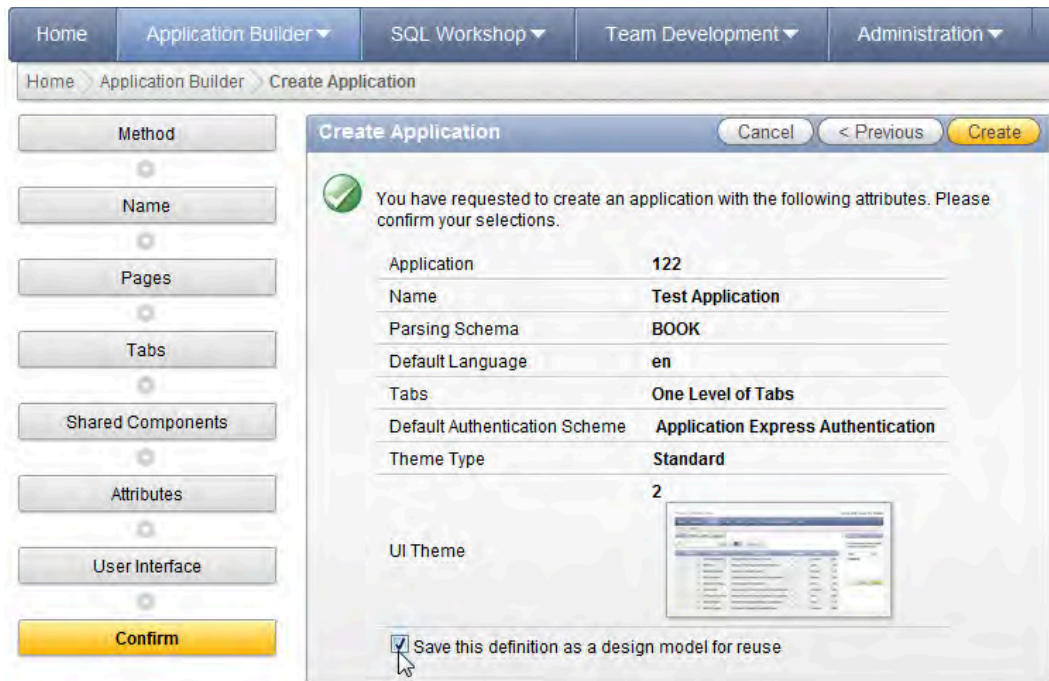


Figure 14–12. The option to save the definition as a design model

The next time you create an application from scratch you'll be given the option to create an application based on existing application design model, as shown in Figure 14–13.

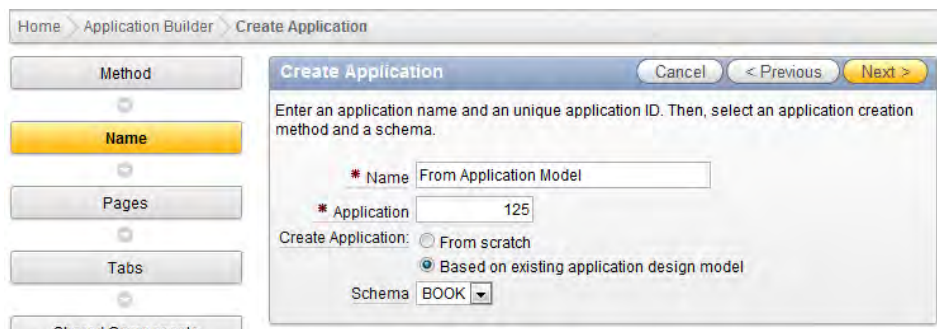


Figure 14–13. Create an application from existing design model

After selecting the “Based on existing application design model” option and clicking the Next button, you can select a saved design model, as shown in Figure 14–14.

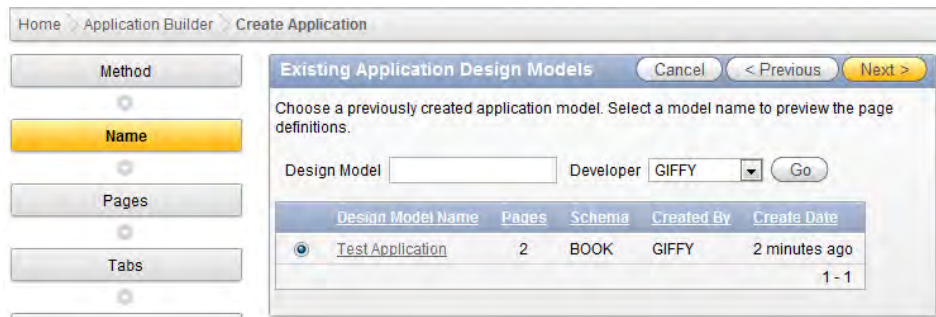


Figure 14–14. Select an application design model

The Application Models page in the Administration section allows workspace administrators to delete any design models. Deleting a design model will have no effect on existing applications.

The only way to modify a design model is to create a new application from scratch, base it on an existing design model, and then save the model at the end of the process.

File Utilization

The File Utilization page provides an overview of all the types of files and their total size in the workspace, as shown in Figure 14–15. There are various locations where files can be stored as part of an APEX application. Over time, these files can take up unnecessary space.

Home > Administration > Manage Service > File Utilization					
<input type="text"/> <input type="button" value="Go"/> <input type="button" value="Actions"/>					
File Type	File Count	Total File Size	Newest File	Oldest File	
Export Repository	3	3MB	12/14/2010	11/26/2010	
Plug-in Files	8	40KB	12/15/2010	11/26/2010	
Cascading Style Sheets (Shared Components)	0	0	-	-	
Images (Shared Components)	0	0	-	-	
Migration Files	0	0	-	-	
Other Workspace Files	0	0	-	-	
Text Files (Shared Components)	0	0	-	-	
Theme Images	0	0	-	-	

1 - 8

Figure 14–15. File utilization information

The export repository tends to consume most of the space compared to other files listed in the File Utilization page. When you import an application into your workspace, the original file is stored in the export repository. The name “export repository” can be misleading since the repository contains the original application files that are imported into the workspace. Once an application has been

successfully imported and installed, you don't need to retain the file in the repository, so it can be removed.

The following steps explain how to clean up the export repository:

1. In the administration module, click on the Manage Export Repository link located in the Tasks region on the right hand side.
2. Select the files that you no longer need and click the Delete Checked button, as shown in Figure 14–16.

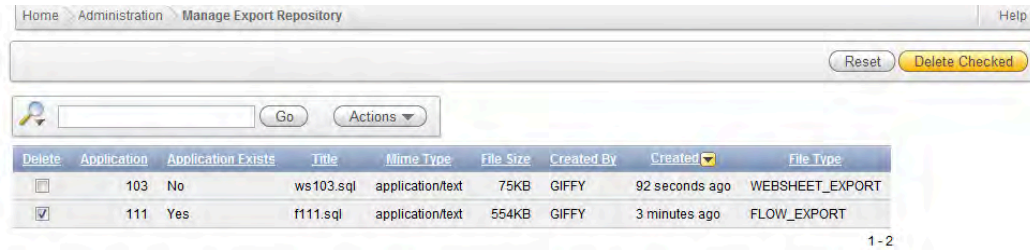


Figure 14–16. Delete Checked button

Interactive Report Settings

The Interactive Reports Settings page allows workspace administrators to manage saved reports and report subscriptions. Only workspace administrators can delete saved reports and report subscriptions. The only way to modify a report configuration is to log in as the user.

Saved Reports

The Saved Reports section allows you to delete certain saved interactive reports. Interactive reports have three different types of saved reports (see Figure 14–17). Primary default is the default report setting and is accessible to all users. Private interactive reports are custom interactive reports that are only visible to the users that created them. The final type, alternative default, is new to APEX 4.0. Alternate default reports are saved by privileged users and are accessible to all users.

On the Saved Reports page, you can delete private (custom) and alternative default reports by selecting the report(s) and clicking the Delete Checked button. Primary default reports can't be deleted.

Application	Page	Region Name	Report Type	Report Name	Report Alias	Owner
102	1	EMP	Alternative Default	Book Demo (alt)	12768	
102	1	EMP	Primary Default	Primary Report	12762	
102	1	EMP	Private	Managers		KAIYA
102	20	Departments	Primary Default	Primary Report	12813	
102	20	Departments	Private	Sales and Ops		JULIAN

Figure 14–17. Managed Saved Interactive Reports page

Subscriptions

Similar to the Saved Reports section, you can delete interactive report subscriptions from the Subscriptions section. Subscriptions allow users to receive e-mailed reports on a schedule that they define. Subscriptions are a new feature in APEX 4.0 and must be explicitly enabled by developers.

On the Manage Subscriptions page, workspace administrators can only delete specific subscriptions or all subscriptions, as shown in Figure 14–18. Just like saved reports, you can't modify any of the subscription attributes.

Application	Page	Region	Subscribed By	Frequency	Email Address	Status	Start Date
102	1	EMP	MALIYA	Daily	maliya@clarifit.com	Sent	21-NOV-2010
102	1	EMP	LEXIYA	Weekly	lexiya@clarifit.com	Submitted	21-NOV-2010

Figure 14–18. Manage Interactive Report Subscriptions page

Managing Users and Groups

This section will cover most of the features available when you click the Manage Users and Groups icon located on the Administration home page shown in Figure 14–1. All references assume that you start on the Manage Users and Groups page shown in Figure 14–19.

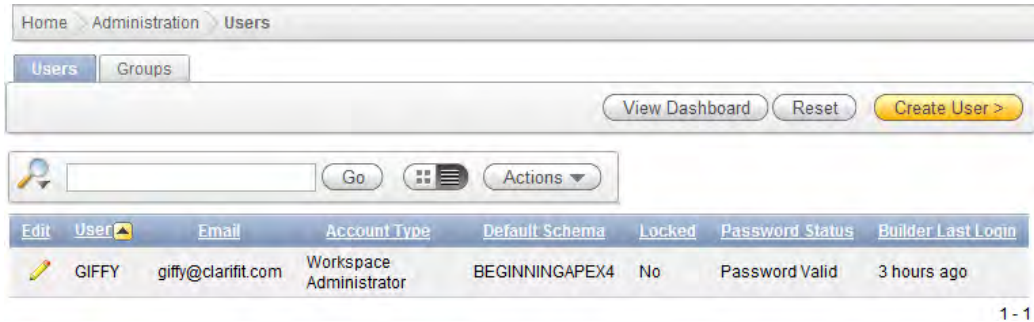


Figure 14–19. Manage Users page

The main page contains a list of all the end users, developers, and workspace administrators. You can modify a user by clicking on the edit link beside their name. End users do not have access to the development environment and will only have access to applications if the authentication scheme is the default Application Express.

Creating One User

To create a single user, click on the Create User button in Figure 14–19. If you are only creating developers, you don't need to worry about entering some of the non-required fields such as first and last name because you will usually know the user based on the username. You can also grant access to specific modules. For example, if you create an account for a project manager, you may only want to give them access to the Team Development module shown in Figure 14–20.

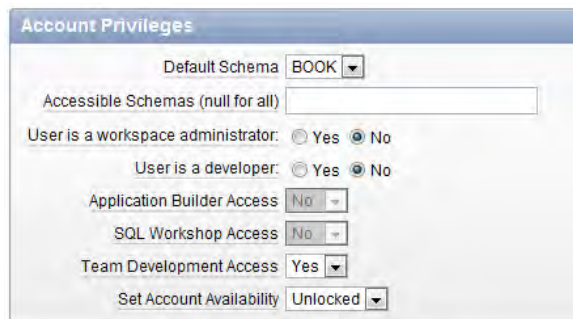


Figure 14–20. Account privileges

Creating Multiple Users

Creating multiple users one at a time using the Create User page can get frustrating. APEX provides an interface to create multiple users. Start by clicking on the Create Multiple Users link under the Tasks region on the Manage Users and Groups main page, as shown in Figure 14–21.

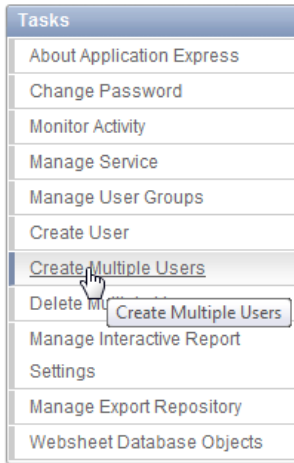


Figure 14–21. Create Multiple Users menu option

On the Create Multiple Users page, you need to provide an e-mail address for each user. The e-mail addresses can be delimited by commas, semicolons, or new lines. APEX generates the usernames from the e-mail addresses that you enter. For example, the e-mail address of `stephanie@clarifit.com` will yield a username of either “stephanie@clarifit.com” or just plain “stephanie”. You can specify whether you wish each e-mail address to translate directly into a username, or whether you want the usernames to exclude the @domain suffixes. Figure 14–22 shows the radio button set for the latter option.

If your usernames do not correspond to your corporate e-mail addresses, you can enter e-mail addresses with invalid domains and select the option to omit the suffixes. For example, enter the username “jonathan@example.com” when you just want a user named jonathan. Then select the option to exclude the @domain portion of the e-mail address.

The password that you enter will be the same for all the users. When they log into the workspace, they will be required to change it.

Home > Administration > Dashboard > Create Multiple Users

Users

* List of Email Addresses

```

stephanie@clarifit.com
eric@clarifit.com
reinhard@clarifit.com
rosemarie@clarifit.com

```

Usernames: ☐ Set username to full email address ☒ Exclude @ domain as part of the username

Account Privileges

Default Schema BOOK ▾
 Accessible Schemas (null for all)
 Users are workspace administrators: ☐ Yes ☒ No
 Users are developers: ☒ Yes ☐ No
 Application Builder Access Yes ▾
 SQL Workshop Access No ▾
 Team Development Access Yes ▾

Password

* Password **** Passwords are case sensitive
 * Confirm Password ****

Figure 14–22. Create Multiple Users page

After you enter all the information on the Create Multiple Users page, click the Next button to go to the confirmation page shown in Figure 14–23. If everything is correct, click the Create Valid Users button.

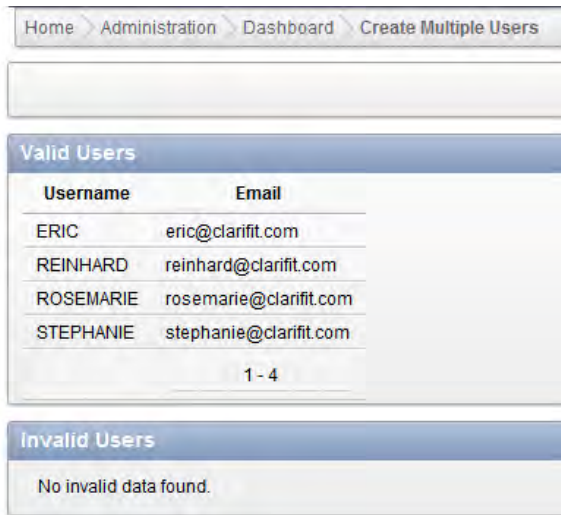


Figure 14–23. Create Multiple Users confirmation

Organizing Users Into Groups

You can associate an APEX user with multiple groups. These groups can be used in authorization schemes to grant or deny access to various parts of an application. Since groups are linked with APEX users, you can only leverage the use of groups when your authentication scheme is set to the default Application Express scheme.

It is important to note that groups are associated with a single workspace. If you have a traditional setup with a development, test, and production environment, you will need to create the same groups in each of the workspaces.

Creating a Group

To create a user group, click on the Manage User Groups link under the Tasks region (shown in Figure 14–21) and click the Create Group button. Enter a unique group name and description, as shown in Figure 14–24.

Figure 14–24. Create a user group

Assigning Users to a Group

Assigning users to a group is not very intuitive. Clicking on the User Group Assignments under the Manage Groups region will only give you a report of all the users and their groups. To assign a user to a group, you need to edit the user (Users ► Edit User) and scroll down to the User Groups region shown in Figure 14–25.

Figure 14–25. Assign user to group

Viewing Usage Reports and Dashboards

The Monitor Activity and Dashboard sections from Figure 14–1 provide you with many detailed and summary reports about your APEX workspace. The reports contain information about both developer activity and end user activity. You are encouraged to explore each of these sections to see all the available reports.

Summary

APEX provides many tools to help you manage your workspace plus reports to get stats about your workspace. Some tools can affect end users and should be used cautiously. However, don't let the need for caution deter you from using tools that can make your job easier. Learn the tools well. Be confident in your knowledge. Take a moment to think before you act. These are the keys to success.



Team Development

Team Development is an excellent tool for managing APEX 4.0 software development. Team Development is embedded within the APEX development environment, allowing you to manage your APEX projects with tools that you and your team use every day like Interactive Reports. This eliminates the need for an external project management tool. Team Development's simple, extensible, and flexible architecture is ideal for managing projects of any size. Its simplicity lends itself to small projects where you want to minimize your project management overhead. Its extensibility scales well, allowing you to manage large projects by using its rich set of attributes to construct a sophisticated work breakdown structure (WBS). Its flexibility allows you to adapt it to your organization's software development culture.

APEX and Team Development are both well suited to work with agile software development methodologies. APEX's Rapid Application Development (RAD) architecture allows the team to rapidly deliver regularly scheduled releases to the business testers and end users. Team Development's feedback mechanism efficiently channels end user issues back to the development team so that appropriate changes can be included in the following releases and easily documented. Team Development makes APEX, which is already an efficient development platform, even more efficient.

The primary purpose of this chapter is to highlight how Team Development works. A secondary purpose is to illustrate how agile software development practices can be used with Team Development to deliver quality software while meeting your cost and schedule constraints. By the end of the chapter, you should have a good understanding of what is in Team Development and some ideas about how you will integrate it into your team's development culture.

Team Development was added to APEX in version 4.0. The APEX team itself used Team Development while it was developing APEX 4.0. This design philosophy of using the tool to build the tool is a key reason why Team Development and APEX itself are practical and easy to use tools.

Team Development Overview

Team Development consists of features, milestones, to-dos, bugs, and feedback. These modules are complemented by a small number of utilities called Team Actions that are used to manage the Team Development environment. The entity relationship diagram (ERD) illustrates the relationships between the main entities (see Figure 15-1).

The Team Development entities are contained within a single workspace. When you are managing several applications simultaneously within a workspace, you and your teams can easily filter Team Development's data by using the tool's Interactive Reports. That way, each stakeholder sees only the work that is of immediate interest for the tasks at hand.

Features describe the big picture. At a high level, they describe and track progress for the major pieces of functionality that your client has requested. Attributes like owner, start date, due date, summary description, priority, status etc. are easily tracked. In addition, you can optionally track the status of individual components like the user interface, testing, documentation, globalization, security, and accessibility. The self-join that is associated with features allows you to further sub-divide a feature

into child features as development progresses and the design becomes more refined. The high-level nature of features makes them an ideal source for management status reports.

Milestones are the high-level scheduling component of Team Development. They can be used to track major and minor releases of an application. Do not confuse milestones with the due dates that are associated with features and to-dos. For example, a milestone could be the scheduled date for a product release, but a feature that is included in the release might be due well in advance of the milestone date. If you use an agile software development strategy, milestones are well suited to define the “time boxes” or “sprints.”

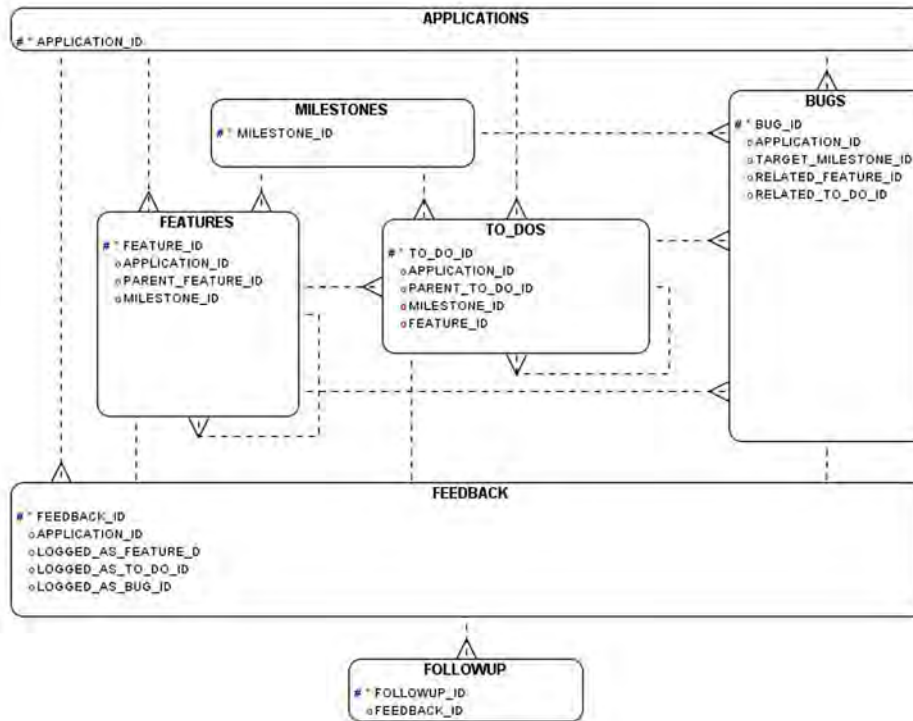


Figure 15–1. A simplified entity relationship diagram of Team Development

To-dos are used to assign detailed work to individuals or teams. Due dates, time estimates, etc. are tracked. Like features, to-dos have a self-join capability that allows you to subdivide them as more information becomes known during the development process. This is handy when a single to-do requires effort from more than one team.

When a defect is found, it’s reported as a bug. Bugs have their own lifecycle that includes a rich set of attributes such as description, resolution, release, assignee, context, and date information. When you find a bug that turns out to be a design flaw, Team Development allows you to link the bug to multiple features, milestones, and to-dos as the bug fix might require an extended time that spans a number of product releases.

Feedback is one of the main sweet spots of Team Development. The feedback mechanism can be installed in an application in a matter of minutes and gives the entire team an efficient and elegant

communication channel for comments, suggestions, issues, defect reports, and responses. Once installed in an application, the feedback mechanism is promoted with the application from the development environment to the test environment and on to the production environment. Feedback data and the related responses are copied back and forth between the various environments via the APEX import/export facility. This keeps the developers, testers, and end users in close touch with each other. Another key feature of the feedback mechanism is that it automatically records all of the critical “under the hood” data that end users know nothing about. The application context, environment variables, and, most importantly, the session state are all captured. This data is invaluable when you are diagnosing an issue.

At first glance, there seems to be a large number of attributes contained in the Team Development entities. So, do you have to enter all of these attributes in order to use Team Development? No. To get the most out of Team Development, you and your team must plan how you are going to use Team Development with a view to effectively managing the software development process with minimum effort. You can pick a small subset of Team Development’s attributes that fit your software development culture and key in only that subset of the data. The Interactive Reports can easily be customized to display only the data that is important to you.

Team Development Interface

The Team Development interface is consistent with the overall APEX development interface. It makes extensive use of dashboards to quickly give you an idea of how your teams are progressing. Drilling into the details to view and update the data is fairly straightforward and intuitive. APEX developers will have no trouble navigating within the tool or customizing their Interactive Reports. Managers, testers, and business analysts will also be able to make use of Team Development after a short training session.

APEX Home Page

The APEX home page (see Figure 15–2) clearly indicates how important Team Development is to the APEX team. The APEX team could have easily put Team Development under a minor link somewhere on the home page. Instead, they have promoted Team Development into a marquee module that is on an equal footing with Application Builder, SQL Workshop, and Administration.

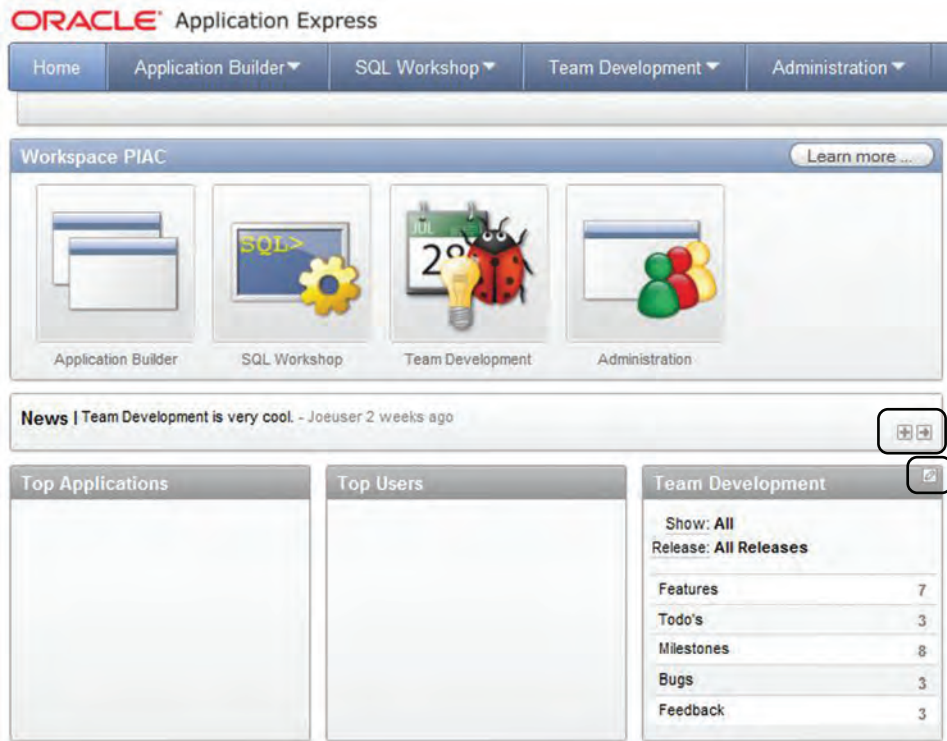


Figure 15–2. APEX home page

The APEX home page highlights the easy and convenient navigation. The News region, which is a feature of Team Development, contains two links (Add and Edit) that enable quick changes to the region without having to drill down into sub pages. The Team Development dashboard region also features an Edit link that controls the Show and Release filters. Look for similar links throughout the Team Development interface; they are very handy, indeed.

Team Development Home Page

The Team Development home page (see Figure 15–3) serves two purposes. One, it contains links to all of the detailed entity regions; and two, it shows you a dashboard for each entity. You'll also see links to Team Actions, which are discussed near the end of the chapter in a section called "Team Actions."

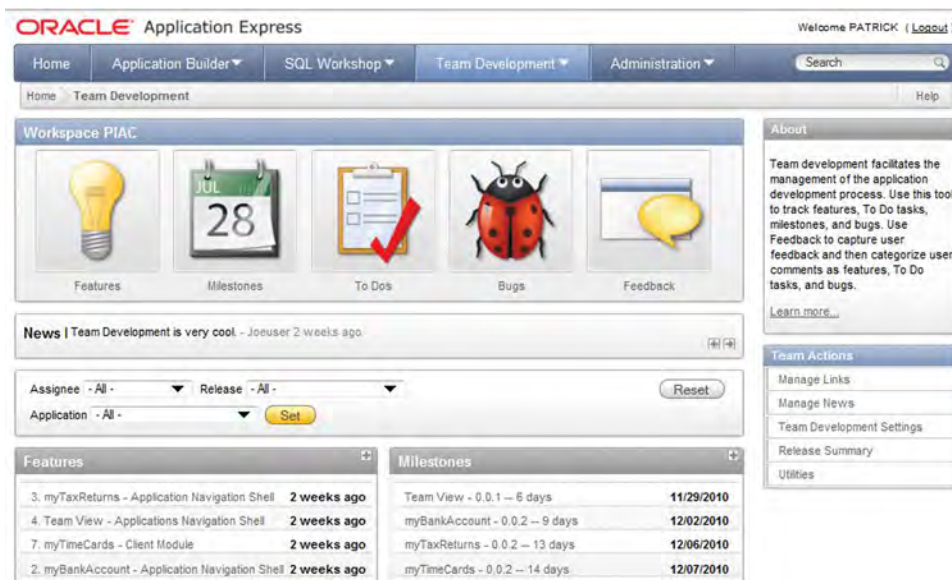


Figure 15–3. Team Development home page

The dashboards can be filtered by assignee, release, and application. These filters make it easy for users to customize the page to their needs. For example, managers can ask “How are we doing?” while a developer can ask “What do I need to do for this release?” This high-level filter region is found on the dashboard pages associated with each Team Development entity. The filter fields are tailored to each entity. The dashboard regions link to their underlying entities. There is an Add icon in each header; when you click on the name, feature, or milestone, you are taken directly to that entity’s Edit page.

Common Design Elements

When you drill into the individual Team Development entities, you will find several common design elements that help you navigate quickly and intuitively between entities. Figure 15–4 highlights these design elements. In the upper right area is a group of icons that are used to switch between the major entities. On the upper left is a set of tabs that is tailored to each entity. Some of the tabs are common to all entities. The Dashboard, Details (feature, milestone, to-do, etc), and Calendar tabs are common to each entity. Others are unique to an entity. When you are on a tab that has an equivalent in the other entities, pressing one of the upper right navigation icons will take you to that equivalent page. For example, when you are looking at the Calendar tab and you press the Milestone icon, you will be taken directly to the milestone’s Calendar tab. When you are on a tab that has no corresponding equivalent in the other entities, the upper right navigation icons are not displayed.

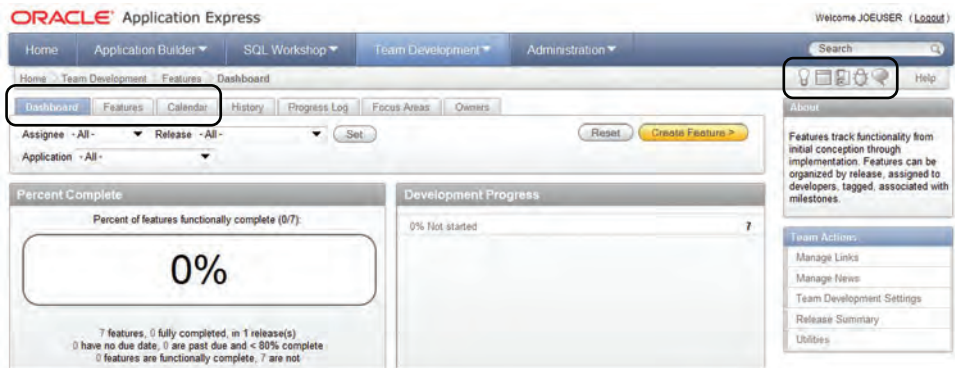


Figure 15–4. GUI designs that are common to all entities

All of the dashboard pages contain a filter region that is tailored to the entity. The filter select lists contain entries for only the entities that are tracked by Team Development. For example, if there is an application that doesn't have any features associated with it, that application will not appear in the select list.

Many of the individual dashboard regions contain links that take you to the entity's Detail page and automatically set the filters on the detail page's interactive report so that you see only the entity records that are related to the dashboard item you selected. This navigation strategy, once you get used to it, is convenient.

Drilldown Functionality

The Calendar tabs (see Figure 15–5) display links to individual entity records based on the entity record's due date. Clicking on the Calendar link takes you directly to the Edit page for the selected entity record. Pressing the Cancel button on the Edit page returns you to the calendar.

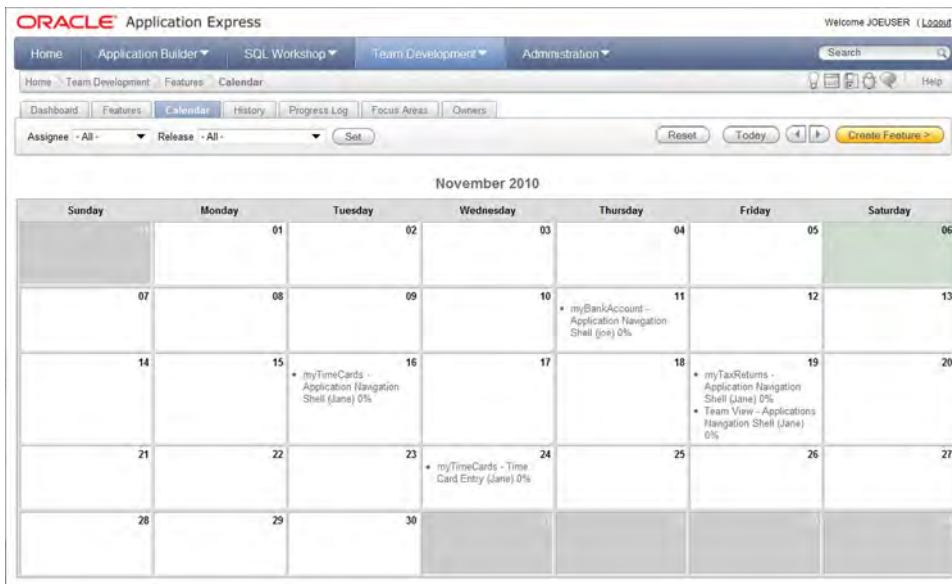


Figure 15–5. Calendar with links to an Edit page

Some of the Entity tabs display their data graphically (see Figure 15–6). Clicking on a graphic pie slice automatically takes you to the entity’s Detail page and sets the Interactive Report to show only the data that was selected on the graph.

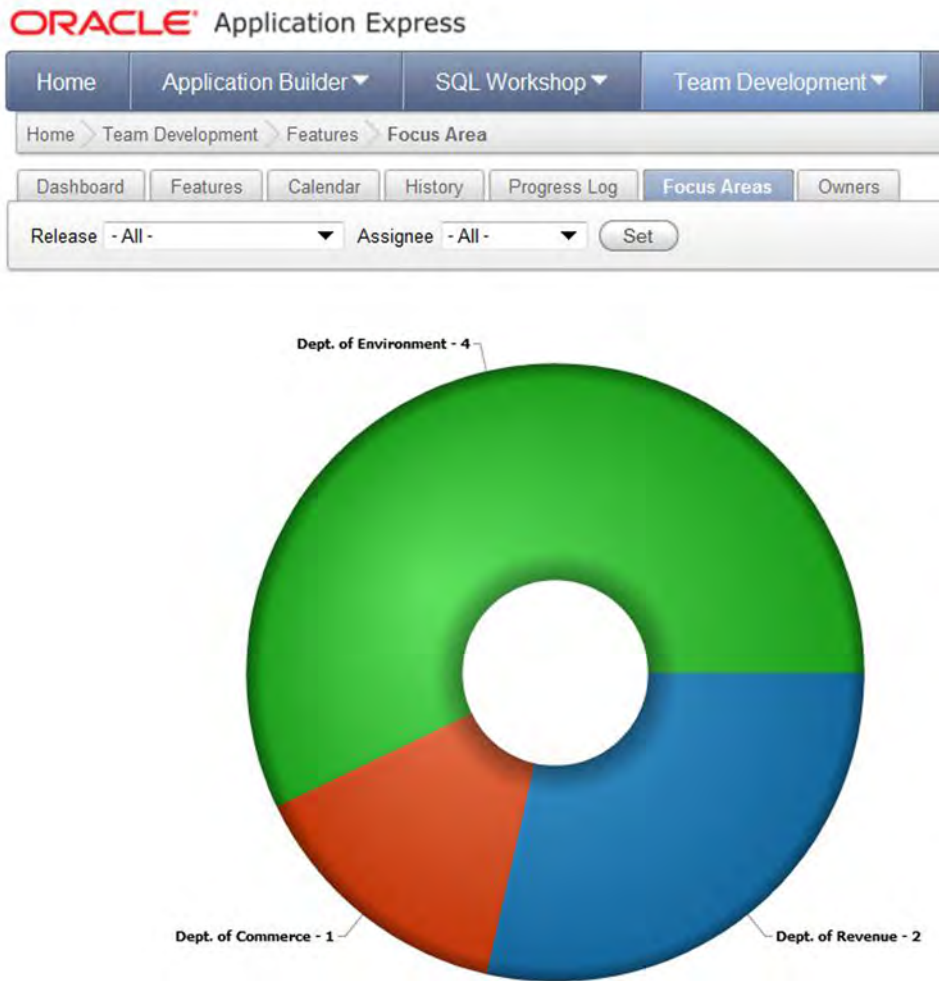


Figure 15-6. Graphic data summary

Tagging

A powerful organizational attribute called tags is associated with all of the entities. A tag is a free form text input that enables you to group records by a key word. This is handy because some record groupings do not fit into the neat and tidy relational data model. For example, you might want to be able to find all of the features that contain a shuttle item across all applications. By adding the tag “shuttle” to these features, you can use the Interactive Report search field to find all records that contain the tag of “shuttle.” All of the dashboard pages contain a region that displays all of the tag strings that are defined in their respective entity. This enables you to quickly find and correct typos so that the free form tags can be kept accurate.

GETTING PAST THE INITIAL DISCOMFORT

At first, Team Development's organization, layout, and navigation might appear a bit strange and non-intuitive. We certainly struggled with it at first. Our initial perception was probably due to our previous experience with project management tools that utilize pages with a work breakdown structure (WBS) on the left and a Gantt chart on the right.

Our initial discomfort with Team Development was similar to our first experiences with APEX itself. Like many experienced developers, we had come from development environments in which screen widgets are listed on the left and dragged onto a screen, and in which the X-Y coordinates are set together with a widget's height and width, all at pixel level precision.

When we first used APEX, most of us took a while to get used to how APEX built pages. After a few tries, we all have learned to love the product. We believe the same will be true with Team Development. After a bit of experience, you will find that the Interactive Reports are, in fact, a practical alternative to Gantt charts, especially in an agile software development environment in which the time boxes are short and the lists in the Interactive Reports are correspondingly small.

Features

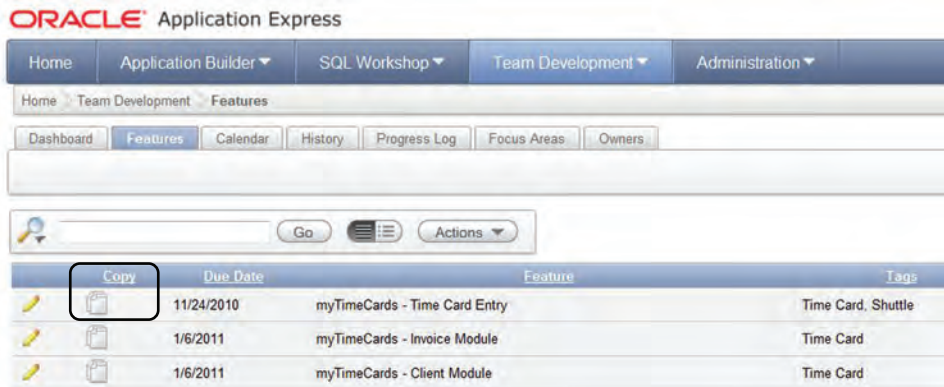
Features describe an application from a high-level perspective. They are used at the beginning of a development project to define the scope well enough so that budgetary estimates can be made of cost, schedule, and resource requirements. Once the project is approved and work begins, child features are added to describe, control, and track progress in more detail. Features are the heart of management status reports.

The features Detail page contains a number of tabs. The Dashboard, Calendar, Focus areas, and Owners tabs are not discussed here as they or their equivalents are common to all entities and their characteristics are mentioned above. The features, history, and progress logs are highlighted below.

Features Tab

The Features tab (see Figure 15–7) contains an Interactive Report that, in turn, links to the Edit page for individual feature records. Since the data grid is an Interactive Report, each user can tailor it to their needs. You can choose your columns from approximately fifty attributes, sort to your taste, and even group related records together. Interactive Reports are discussed in detail in Chapter 6.

The Copy link is useful when you are entering a number of related features. For example, if you are entering ten or twenty features that belong to a single application, you might expect that many of the data attributes will be repeated on every record. The Copy link creates a new feature as an exact copy of the existing one in the Interactive Report. Only the name is changed. Once the new feature is created, you can then quickly edit it and change the attributes that are unique to that record.



ORACLE Application Express

Home Application Builder SQL Workshop Team Development Administration

Home Team Development Features

Dashboard Features Calendar History Progress Log Focus Areas Owners

Go Actions

	Copy	Due Date	Feature	Tags
		11/24/2010	myTimeCards - Time Card Entry	Time Card, Shuttle
		1/6/2011	myTimeCards - Invoice Module	Time Card
		1/6/2011	myTimeCards - Client Module	Time Card

Figure 15–7. Features interactive report

The Edit link takes you to the features Edit page (see Figure 15–8). You can easily explore the individual attributes on your own as their definitions are generally self-evident and documented by clicking an attribute’s label. However, two areas on Figure 15–8 are highlighted. Those deserve more explanation.

- **User interface, testing, documentation, globalization, security, and accessibility regions:** These regions are used to track sub-life cycles that are associated with a feature. For example, if a separate team is responsible for documentation, their progress can be tracked separately from the main feature (see Figure 15–9). If the documentation effort is not being tracked, then the documentation region can be excluded from the features Edit page by turning it off in the Team Development settings area.
- **Configurable Lists of Values (LOV):** Some LOV, like feature status, are controlled by the APEX team. Others, like the feature owner, are controlled by the development team. Instead of maintaining the developer configurable LOV in a separate maintenance area, these lists are maintained in place. For example, to add a new feature owner, you enter the appropriate name in the New Owner item, press the Apply Changes button, and that name is added to the LOV. Unfortunately, currently there is no easy and safe way to edit these configurable LOV; hopefully this ability will come in a future APEX release.

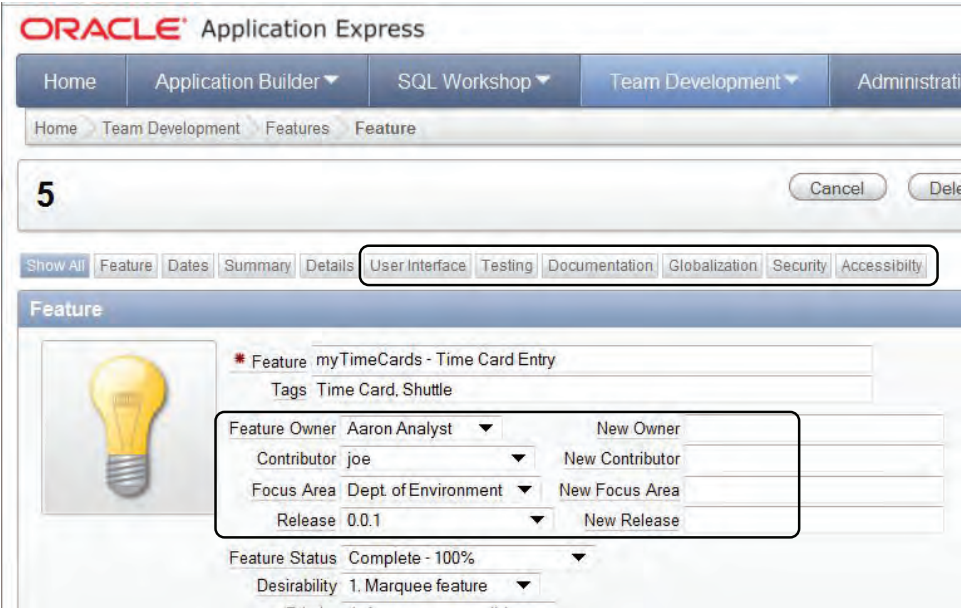


Figure 15–8. Feature Edit page

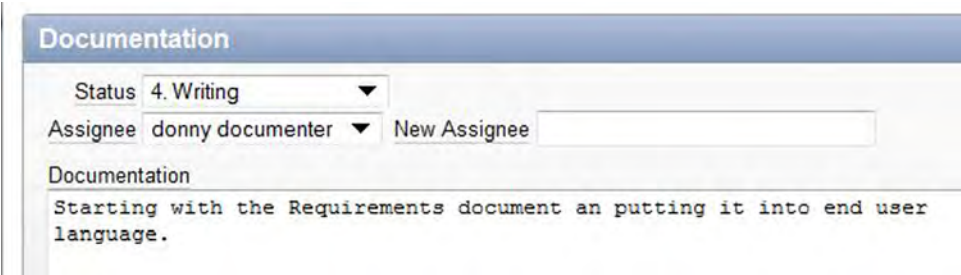


Figure 15–9. Feature documentation region

History Tab

The History tab (see Figure 15–10) presents a detailed audit trail of all changes made to all of the features. The tab presents an Interactive Report that contains the feature, column name, old value, new value, the user who made a change, and when the change was made. You can view the report on the History tab for quality assurance purposes or as a source of evidence if contractual disputes arise.

Feature	Column Name	Old Value	New Value	User	Changed
Team View - Applications Navigation Shell	Milestone Number	-	5070731422638142848	joeuser	5 weeks ago
Team View - Applications Navigation Shell	Feature Owner	Jane	tim tester	patrick	4 weeks ago
Team View - Applications Navigation Shell	Application Number	29443	31534	joeuser	5 weeks ago
Team View - Applications Navigation Shell	Due Date	20101119	20101205	patrick	2 weeks ago

Figure 15–10. Features History tab

Progress Log Tab

The progress log in Figure 15–11 is, in effect, a diary. Stakeholders can, on the features Edit page, enter free form text into the progress log at any time. The progress log acts as a communication channel for the development team and can be used to record new ideas, reminders, a flash of brilliance, questions, and anything else that needs to be remembered.

Progress Note	Feature Owner	Feature Name	Created	Created By	Release
Jim, let's scrap the tabs and go with a navigation list on the left hand side of the pages.	tim tester	Team View - Applications Navigation Shell	4 weeks ago	patrick	Team View - 0.0.1
Follow up to the scrum, the accounting department already has a page that we can use as a starting point.	Jane	myTimeCards - Invoice Module	6 weeks ago	joeuser	myTimeCards - 0.0.2
I strongly recommend BI Publisher for printing the invoices.	Jane	myTimeCards - Invoice Module	6 weeks ago	joeuser	myTimeCards - 0.0.2

Figure 15–11. Features Progress Log tab

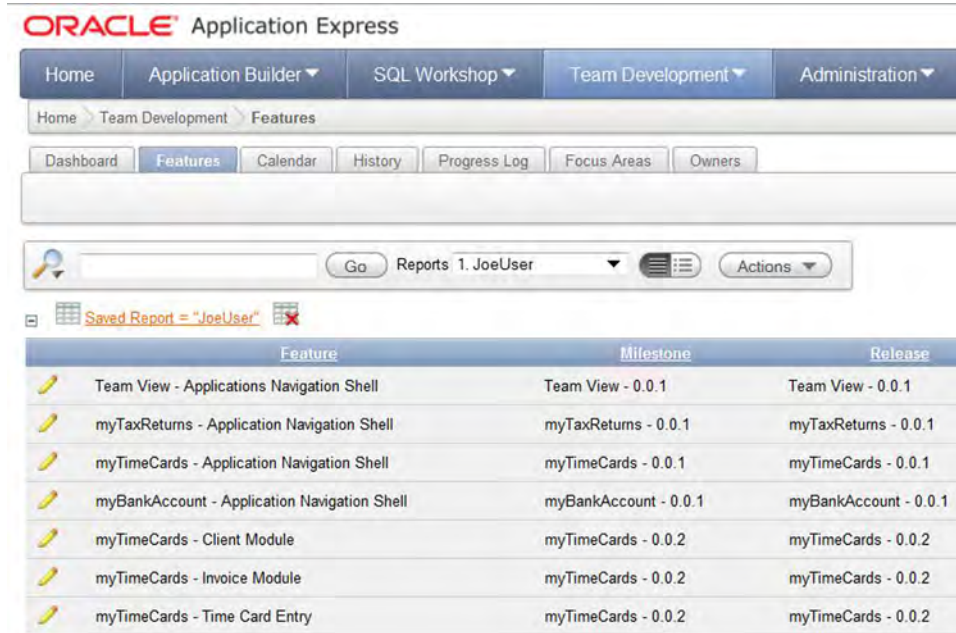
Milestones

Milestones are used to define and track event dates for both scheduled and one-time happenings. On the surface, this appears to be a simple concept; however, it can be a confusing area because milestone functionality overlaps with releases and can be confused with the due dates associated with features, to-dos, and bugs. Fortunately, the confusion can be mitigated with a bit of planning and organization.

One strategy that can be used to organize milestones and releases is to create one of each for the same release event. The milestone contains the metadata that is associated with the release: the date, a type, owner, description, and tags. The release is defined as a configurable LOV that is shared by all of the Team Development entities but it contains no descriptive metadata. Also, releases are one of the

handy high-level filters associated with features, to-dos, and bugs. The need for both metadata and the high level filters is the reason that a milestone and a release must be used to describe the same event. Happily, the overhead in doing this is low.

Another suggestion for workspaces that contain multiple applications is to prefix features, milestones, releases, to-dos, and feedback with the name or code of their associated application (see Figure 15–12). In many cases, this is not strictly necessary but it can make some of the dashboards more readable as they often contain only an entity's name without any supporting data.



The screenshot shows the Oracle Application Express interface. At the top, there's a navigation bar with tabs: Home, Application Builder, SQL Workshop, Team Development (selected), and Administration. Below this is a breadcrumb trail: Home > Team Development > Features. A secondary navigation bar contains: Dashboard, Features (selected), Calendar, History, Progress Log, Focus Areas, and Owners. Below the navigation is a search bar with a magnifying glass icon, a 'Go' button, a dropdown menu showing 'Reports: 1. JoeUser', a list icon, and an 'Actions' dropdown. Below the search bar, there's a message: 'Saved Report = "JoeUser"'. The main content area displays a table with three columns: Feature, Milestone, and Release. The table contains eight rows of data, each with a yellow pencil icon in the first column.

Feature	Milestone	Release
Team View - Applications Navigation Shell	Team View - 0.0.1	Team View - 0.0.1
myTaxReturns - Application Navigation Shell	myTaxReturns - 0.0.1	myTaxReturns - 0.0.1
myTimeCards - Application Navigation Shell	myTimeCards - 0.0.1	myTimeCards - 0.0.1
myBankAccount - Application Navigation Shell	myBankAccount - 0.0.1	myBankAccount - 0.0.1
myTimeCards - Client Module	myTimeCards - 0.0.2	myTimeCards - 0.0.2
myTimeCards - Invoice Module	myTimeCards - 0.0.2	myTimeCards - 0.0.2
myTimeCards - Time Card Entry	myTimeCards - 0.0.2	myTimeCards - 0.0.2

Figure 15–12. Suggested naming pattern

Milestones, of course, are used to define and track other events in the software development lifecycle. Important meetings, tool software upgrades, and requirement deliveries are just a few examples.

The milestones interface contains a number of tabs. The Dashboard and Calendar tabs are common to all entities and are discussed above.

Milestones Tab

The Milestones tab in Figure 15–13 contains an Interactive Report that contains a list of milestones. The high-level filters let you select all or only future events as well as individual releases. The Edit link in the Interactive Report takes you to the milestone's Edit page, which contains intuitive items that are documented under their labels.

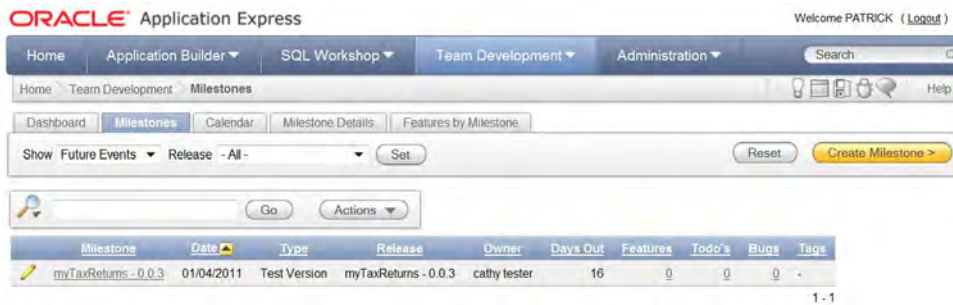


Figure 15–13. Milestones tab

Milestone Details Tab

The Milestone Details tab (see Figure 15–14) is a dashboard that summarizes the relationships between a single milestone and the other Team Development entities. Knowing the number and status of the related features, to-dos, and bugs is a good management tool that is useful for controlling a sprint or time-box that leads up to a release.

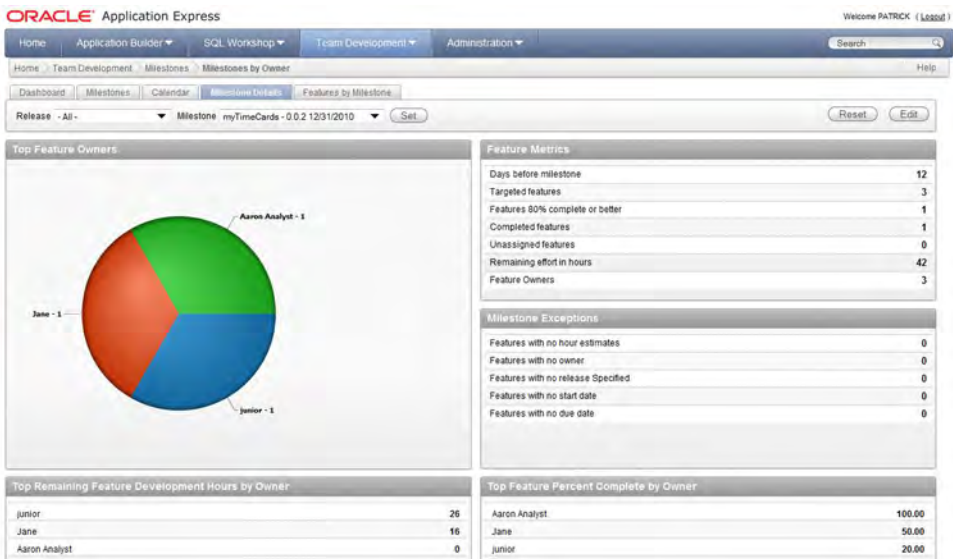


Figure 15–14. Milestone Details tab

Features by Milestone Tab

The Features by Milestone tab (see Figure 15–15) contains an Interactive Report that, in detail, illustrates the relationship between milestones and the features that are associated with the milestone. This is mainly a management report.

Milestone	Milestone Date	Feature Name	Feature Owner	Status	Feature Due	Feature Release	Hours	Tests
myTimeCards - 0.0.2	12/31/2010	myTimeCards - Time Card Entry	Aaron Analyst	Complete (100%)	11/24/2010	myTimeCards - 0.0.2	16	Time Card, Shuttle
Team View - 0.0.1	11/29/2010	Team View - Applications Navigation Shell	tim tester	Under consideration (10%)	12/05/2010	Team View - 0.0.1	4	-
myBankAccount - 0.0.1	11/10/2010	myBankAccount - Application Navigation Shell	joe	Not started (0%)	12/05/2010	myBankAccount - 0.0.1	4	-
myTaxReturns - 0.0.1	11/22/2010	myTaxReturns - Application Navigation Shell	jane	Not started (0%)	12/05/2010	myTaxReturns - 0.0.1	4	-
myTimeCards - 0.0.1	11/12/2010	myTimeCards - Application Navigation Shell	jane	Not started (0%)	12/05/2010	myTimeCards - 0.0.1	4	-
myTimeCards - 0.0.2	12/31/2010	myTimeCards - Client Module	junior	Approved (20%)	01/06/2011	myTimeCards - 0.0.2	32	Time Card
myTimeCards - 0.0.2	12/31/2010	myTimeCards - Invoice Module	Jane	Work Progressing (50%)	01/06/2011	myTimeCards - 0.0.2	32	Time Card

Figure 15–15. Features by Milestone tab

To Do Items

To-do items are the workhorses of Team Development. Team Development uses the term *to-do* as a noun. A to-do is a specific task that is assigned to an individual or team along with a due date. In simple terms, a to-do describes the **what**, **who**, and **when** of the task at hand.

To-dos can be subdivided into many child and grandchild tasks. This process of decomposition is done to the point of diminishing returns. A rule of thumb is that the number of sub-task levels should be appropriate for effectively controlling the tasks at hand. What is appropriate is, of course, a function of your team's culture. Also, in an agile software development shop, the sub-task levels will tend to remain shallow due to the daily face-to-face communication between team members.

There are four tabs in the to-do module: Dashboard, To-dos, Calendar, and Progress log. You will find working with these tabs to be easy and intuitive.

There is a handy navigation feature that links the development environment directly to the to-do module. When a to-do has its context set to both an application and a specific page number within that application, a count of to-dos appears on the page's Development screen (see Figure 15–16). In the upper right corner there are counts of to-dos, bugs, feedback, and comments. When you click on the to-do count, the link takes you directly to the To-do Report page that has the Interactive Report filters set to list only the open to-dos that are associated with the development page (see Figure 15–17). The developer can then quickly select one of the to-dos, see what needs to be done, and then quickly return to the Development page to complete the to-do task. This efficient navigation strategy also applies to bugs and feedback.

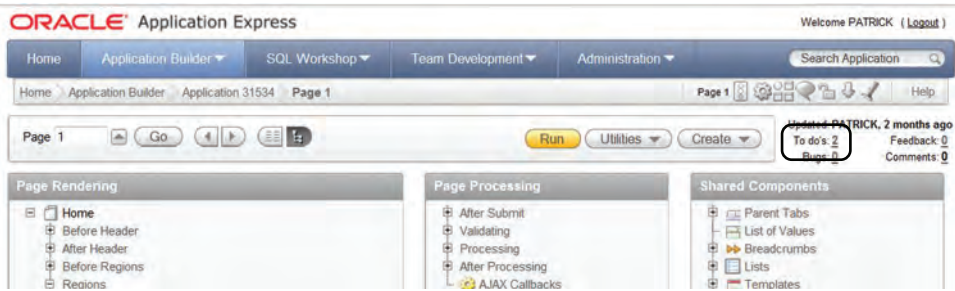


Figure 15–16. Development page linked directly to To-do module

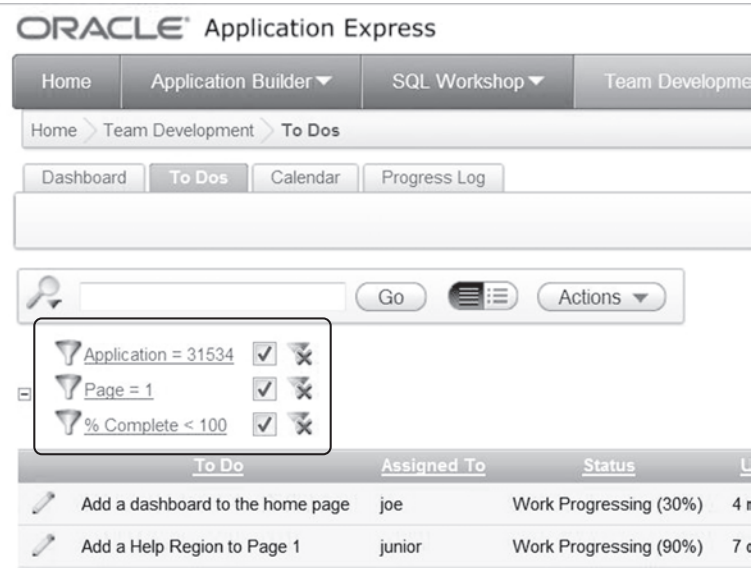


Figure 15–17. To-do page filtered by an application's development count link

Bugs

In the software development world, bugs are a fact of life. We all hope that bugs are found and fixed in the unit and system testing cycles before a product is released to the end users. Happily, this is generally true with the caveat that a small number of bugs will slip through into the production environment. Team Development's bug module is a practical environment that is used to track easy and hard bugs in the development, test, and production environments.

Easy bugs are usually caused by a coding error or a programmer's misunderstanding of a requirement. The symptoms are obvious, the root cause is simple to find, and the fix is almost trivial; for example, changing a plus sign to a minus sign or reorganizing an IF statement in the code. Easy bugs are usually found early in the product's development cycle and are, in general, caught by the programmers

or testers. However, easy bugs must be recorded and reported so that improvements can be made to the coding process; a large number of easy bugs can turn out to be surprisingly expensive.

Hard bugs are, well, hard. They can be caused by design flaws anywhere in the system, subtle interactions between software systems, subtle interactions between software and hardware, and awkward interactions between users and the GUI. Hard bugs can have their own lifecycle and the fix might be spread over several product releases. Team Development's Bug and To-do modules can be used together to track a hard bug's resolution, even when it spans several product releases.

The Bug module highlights how you can take advantage of Team Development's simplicity, extensibility, and flexibility. Simplicity makes tracking easy bugs almost trivial. A bug is found and recorded in Team Development's Bug module where everything about the bug is recorded including the **what**, **who**, and **when** data. Easy bugs can stand on their own or you can associate them with a to-do depending on how you organize your Team Development environment.

Extensibility and flexibility come into play when you deal with hard bugs that may take an extended time to fix and possibly require effort from several developers or teams. Team Development can handle this situation by linking a hard bug to a to-do (see Figure 15–18). The linked to-do is then set up as the parent of several child to-dos that are used to track the tasks that are required for the fix.

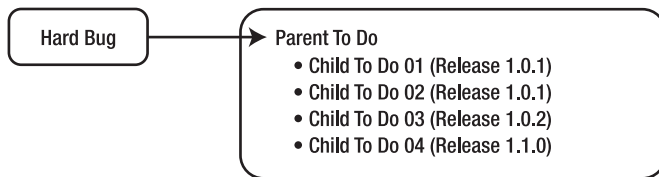


Figure 15–18. Linking a hard bug to a parent to-do

Feedback

Team Development's Feedback module is an APEX "sweet spot." Even if your team chooses not to use Team Development for managing its software development efforts, you should at least consider using the Feedback module. The Feedback module provides a cost-effective channel through which end users, test team members, and even the developers themselves can send suggestions, comments, and bug reports directly to business analysts and developers. Responses to the feedback can optionally be communicated back to the persons who triggered the feedback. And feedback can be taken from all three typical environments: production, test, and development.

■ **Note** The reason that the Feedback module is so cost effective is that the cost-to-benefit ratio almost approaches zero because it takes only a few minutes to set up feedback in an APEX application and it gives you a huge benefit.

Configuring Feedback

Configuring the Feedback mechanism is started by creating a new page in your application. When you press the Create button on your application's developer tool bar (see Figure 15–19) you are taken to the Create Page wizard. This wizard contains a Feedback Page option (see Figure 15–20). Select this option and press the Next button.



Figure 15–19. Application with no feedback

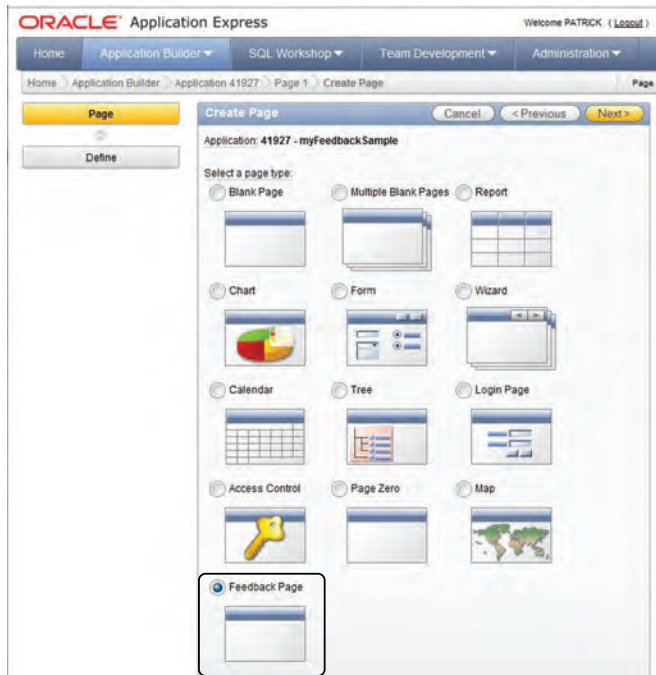


Figure 15–20. Feedback Page option in the Create Page wizard

The next page in the wizard sets up the details for the Feedback page (see Figure 15–21). You enter the page number and page name and then select the page template, form region template, and the label template. You can also declaratively create a Feedback link on the application's global navigation bar. The defaults, in most cases, work well.

However, there is one attribute that requires a bit more explanation because its benefit is not immediately obvious. The extra attributes field allows you to add up to ten custom fields to your feedback page. These are sometimes called flex fields. These fields can be used to prompt the end users for additional information when they submit feedback. For example, a public APEX website requires no login; therefore, the user's identity can't be captured automatically. The **extra attributes** can be used to capture the user's name and e-mail so that the feedback response can be sent to them.

The screenshot shows the Oracle Application Express interface. The top navigation bar includes links for Home, Application Builder, SQL Workshop, Team Development, and Administration. The breadcrumb trail indicates the current location: Home > Application Builder > Application 41927 > Create Page. On the left, a 'Page' tab is active, and a 'Define' button is visible. The main area is titled 'Create Page' and contains the following fields and sections:

- Page Number:** 102
- Page Name:** Feedback
- Popup Page Template:** Popup
- Form Region Template:** 13. Form Region
- Label Template:** Optional Label with Help
- Extra Attributes:** 3 (highlighted with a red box)

Below these fields are two sections:

- Navigation Bar:** Includes a 'Create Navigation Bar Entry' checkbox (checked 'Yes') and an 'Entry Label' field containing 'Feedback'.
- Application Feedback Setting:** Includes an 'Enable Feedback' checkbox (checked 'Yes').

Figure 15–21. Feedback page set up in the Create Page wizard

Press the Create button and then run the application. You will now see the Feedback link in the navigation bar (see Figure 15–22). If you can't use the default Feedback link in your design, you can move it to anywhere you like in your application. The link uses the standard APEX “f?p” syntax.



Figure 15–22. Application with the Feedback link

Polishing the Feedback Page

When you click on the Feedback link, the Feedback Popup page is displayed (see Figure (15–23)). As you can see, the page needs a bit more work to polish it. Since this is a standard APEX page, the polishing can be completed in one or two minutes by pressing the Edit Page button to get into the page's design area.

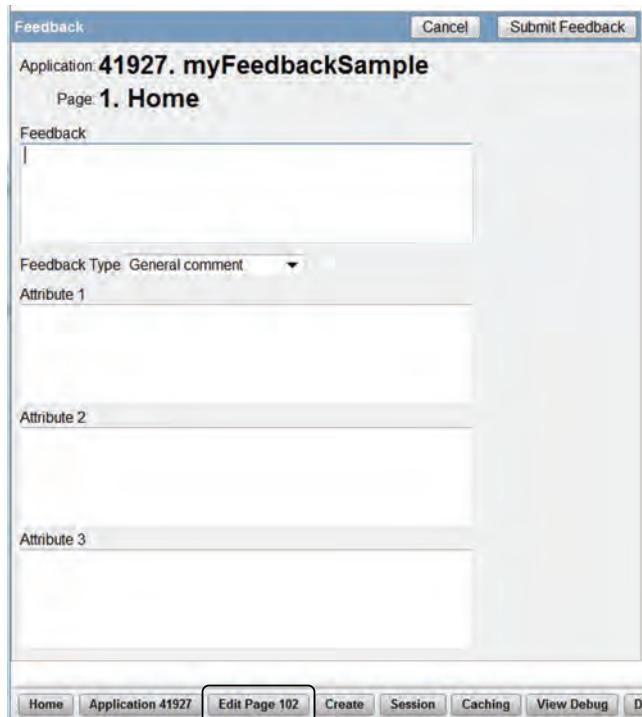


Figure 15–23. Feedback page prior to some polishing

In this example, Attribute1, Attribute2, and Attribute3 will be changed to Name, Department, and Email. APEX automatically added these items to the Feedback page (see Figure 15–24) when you selected **three** in the extra attribute field in the Create Page wizard.

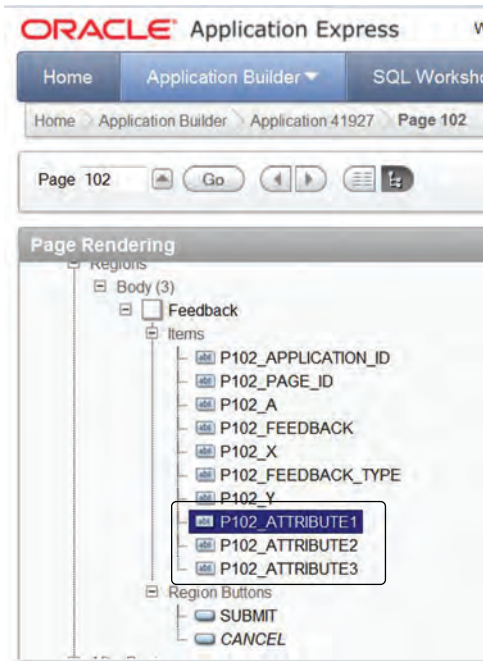


Figure 15–24. Extra attributes on the Feedback page

Configuring the extra attributes is a two step process. First, you rename the items so they describe the data, change the item types from Textarea to an item type that is appropriate for the data, and rename the label to something that is meaningful to the end users (see Figure 15–25).



Figure 15–25. Extra attributes are reconfigured

Second, in the page processing region (see Figure 15–26), you must edit the **submit feedback** process so that the API calling parameters match the renamed and relabelled extra attributes (see Figure 15–27).

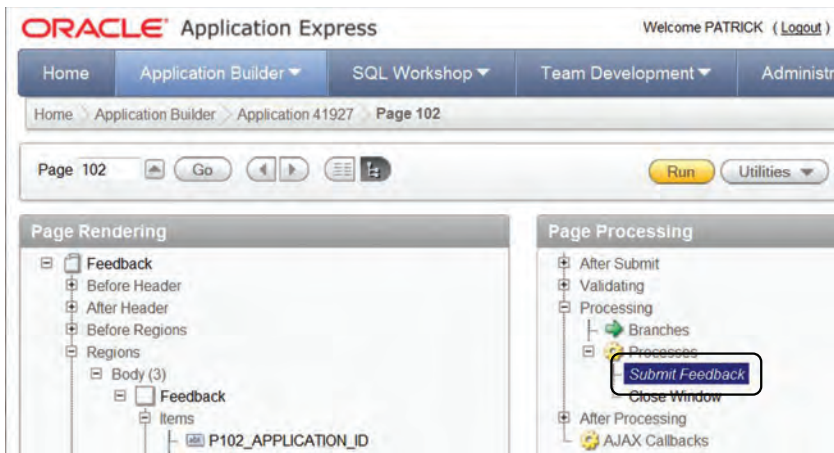


Figure 15–26. Feedback page processing

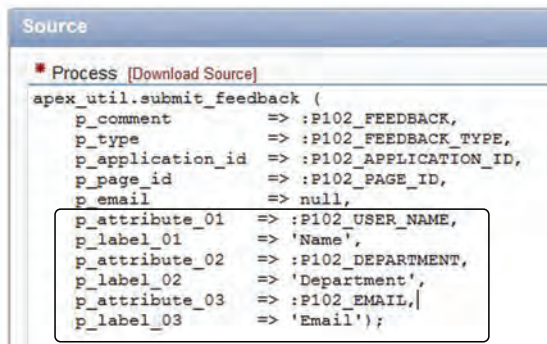


Figure 15–27. Feedback page processing changes in the submit feedback process source

After you complete the changes and you re-run the application, you can see the finished Feedback page (see Figure 15–28). Enter some feedback into the page. Click the Submit Feedback button. Then go on to the next section and learn how to review the feedback that you and others have entered.

Figure 15–28. Finished Feedback page with three extra attributes

Viewing Feedback

You can review all feedback for an application from the Team Development page. Navigate to that page. Then click the Feedback tab. You should see results similar to those in Figure 15–29.

When you drill into the individual feedback record, you'll find a wealth of data and information that will make your life as a developer much easier. The feedback region contains a read-only description of each feedback record. The disposition region is where you track feedback status, tags, developer comments, and public response. In addition, there are three buttons: Log as Bug, Log as To Do, and Log as Feature. These buttons create a new Team Development entity and copy data from the feedback

record to the new entity, which saves time and ensures accuracy. The follow up region is like a diary; it's a list of remarks that are added over time as the feedback is processed. This is handy when several people must review the feedback before action is taken. The extra attributes that were added to the Feedback page are displayed in the additional attributes region. The remaining regions display read-only data that describes the application context, the run-time environment (browser type and version), and the entire session state. If a bug is being reported, the developer has all of the information required to reproduce the bug which is a valuable benefit.

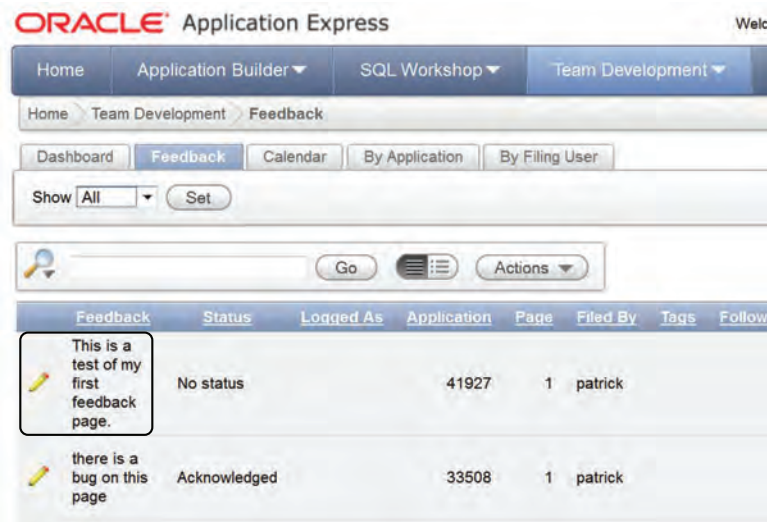


Figure 15–29. Feedback entry

Responses to Feedback

The feedback mechanism contains a response table. In principle, responses should be sent to the users who initiated the feedback. However, there is no easy and declarative mechanism that enables you to send responses to your users. If you want to send responses to your users, you first need to address a number of design issues. For example, do you want to broadcast responses to all your users or send individual responses to individual users? Do you want to use e-mail or create reports? Do you want to send responses to a team? Do you want to route the responses based on a feedback classification scheme? After you design your response strategy, you can then build a tool that fulfills the requirement by writing some PL/SQL code and accessing the APEX views that expose all of the Team Development data. The details of building this tool are beyond the scope of this book.

Communication Between Workspaces

Team Development is a property of a workspace. Many professional shops maintain multiple workspaces for production, testing, and development environments. This means that if an end user enters a feedback record, that record resides in the production workspace and the developers can't see it. This situation is easily remedied by using APEX's existing export/import functionality. APEX version 4.0 has added feedback to the list of entities that can be imported and exported to and from a workspace.

This makes it possible to export the production feedback and import it to the testing environment where the business analysts can evaluate it. If required, the feedback can then be exported from the testing environment and imported to the development environment where the developers can evaluate and then log it as a bug, to-do, or feature.

Team Actions

Team Actions (see Figure 15–30) are a miscellaneous set of utilities that help you manage the Team Development environment. The links to the individual Team Actions are found on the right hand side of the Team Development home page.

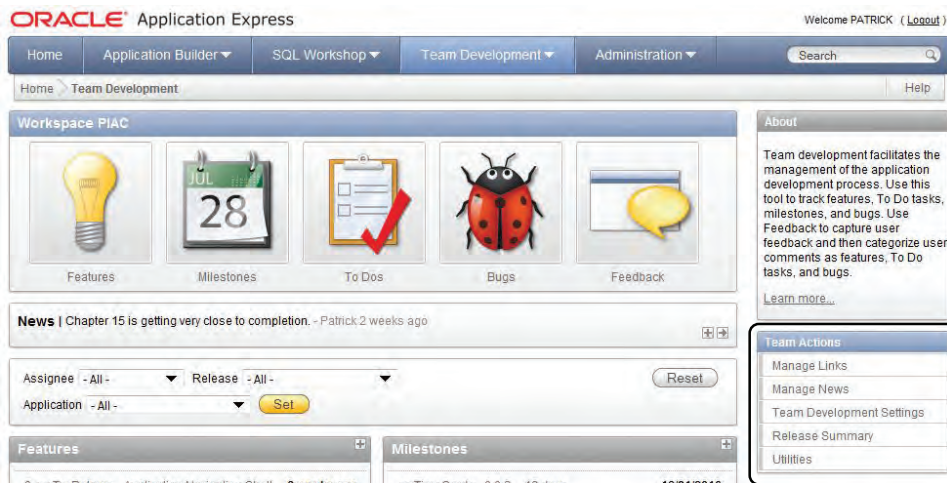


Figure 15–30. Team Actions

Manage Links

The Manage Links feature is a simple list of links to documentation, other systems, and web pages that you can easily define as any valid URL (see Figure 15.31). The links are displayed in a region near the bottom of the Team Development home page. The links let developers quickly navigate to a set of links that have been approved by the Development team. The entire team then shares common documentation, such as SQL or PL/SQL references, that can help the team to adhere to common development styles and standards. This, in turn, helps to encourage consistency which greatly improves the software development process.

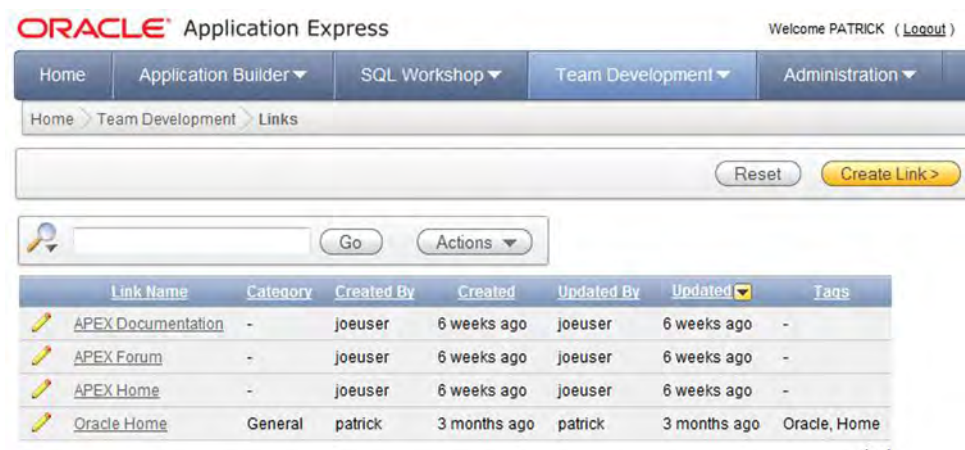


Figure 15–31. Manage Links feature

Manage News

On both the APEX home page and the Team Development home page there is a one-line region called News. One-line news items can be added (see Figure 15–32) by clicking on the Manage News link or by clicking on the plus sign icon in the News region itself. When there are no news items, the region is blank. When there is one news item, it is displayed. When there is more than one news item, they take turns being displayed; the News region refreshes every five seconds or so.

The News region is handy for broadcasting development news when the team is not co-located; the successful or failed promotion of a release to the test environment is an example. Teams that are co-located and that have a daily stand up meeting every morning probably will not use this region.

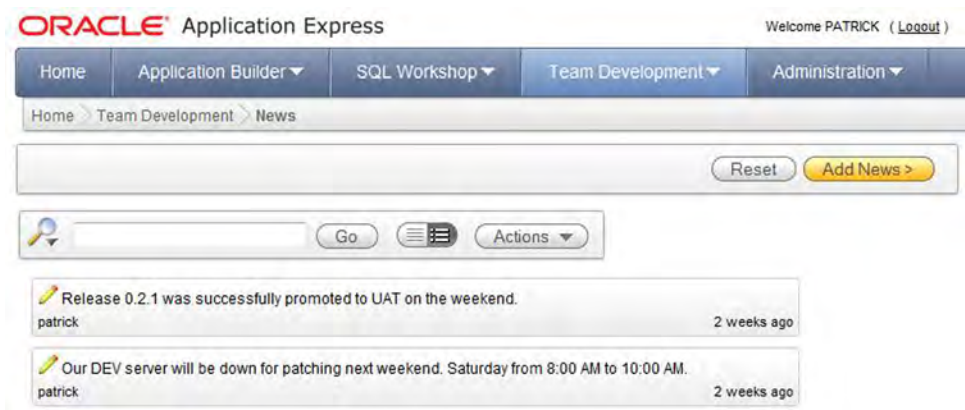
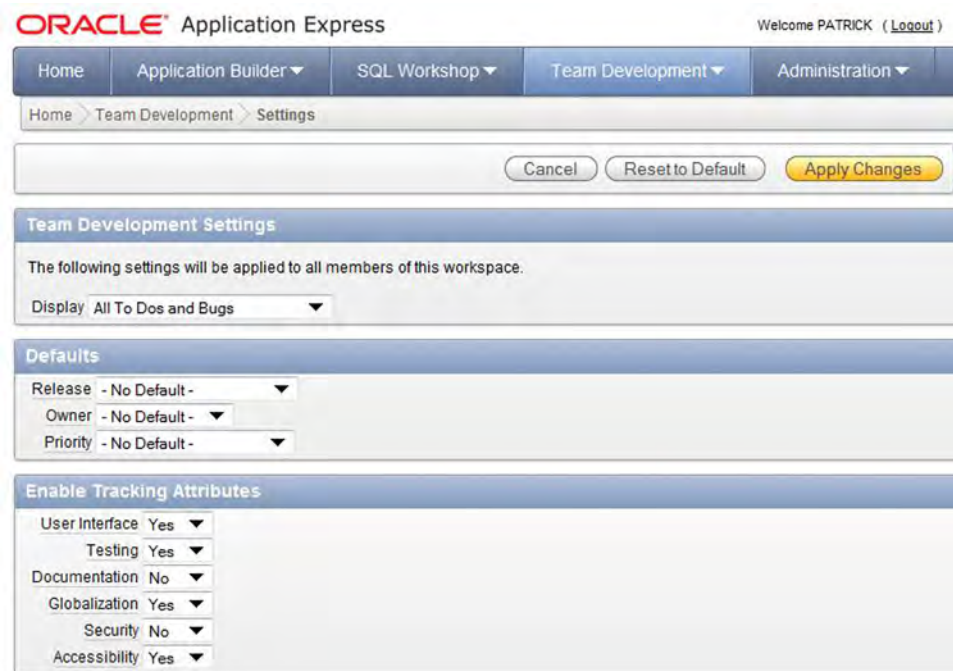


Figure 15–32. Managing the News section

Team Development Settings

A small number of defaults that are global to the workspace are configured on the Team Development Settings page (see Figure 15–33). The **Enable Tracking Attributes** region is used to turn on/off the feature regions that track the detailed work related to the user interface, testing, documentation, globalization, security, and accessibility. When you set these attributes to No, the corresponding region on the feature detail page is not displayed.



ORACLE® Application Express Welcome PATRICK ([Logout](#))

Home Application Builder ▼ SQL Workshop ▼ Team Development ▼ Administration ▼

Home > Team Development > Settings

Cancel Reset to Default Apply Changes

Team Development Settings

The following settings will be applied to all members of this workspace.

Display All To Dos and Bugs ▼

Defaults

Release - No Default - ▼
 Owner - No Default - ▼
 Priority - No Default - ▼

Enable Tracking Attributes

User Interface Yes ▼
 Testing Yes ▼
 Documentation No ▼
 Globalization Yes ▼
 Security No ▼
 Accessibility Yes ▼

Figure 15–33. Settings page

Release Summary

The Release Summary is a management report: it's organized by release name and can be filtered by developer and release name. Figure 15–34 shows a small part of this comprehensive report.

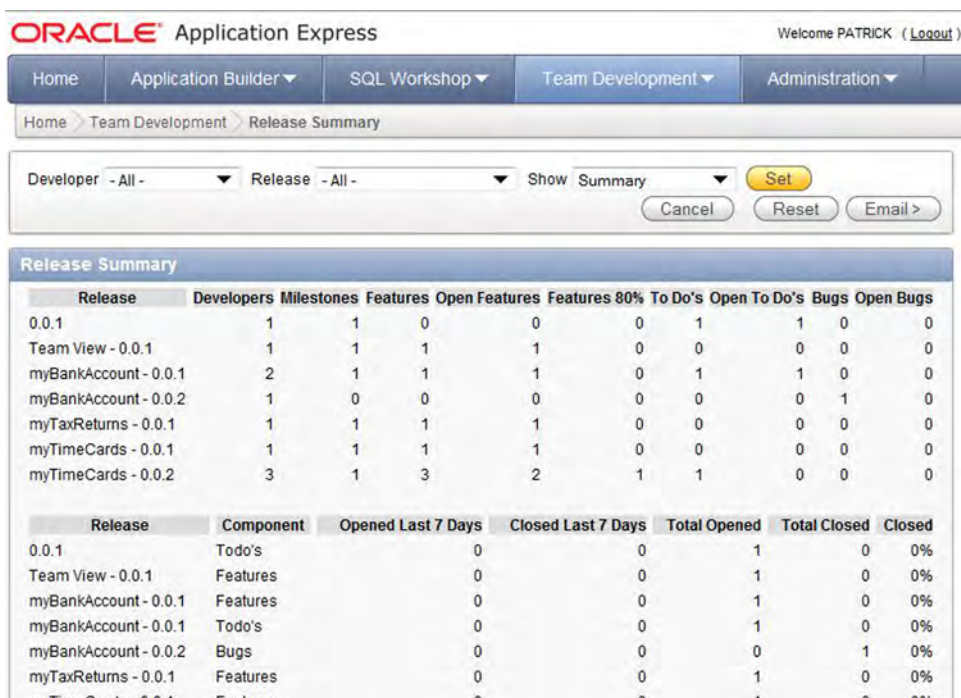


Figure 15–34. Release summary

Utilities

The Team Development utilities (see Figure 15–35) perform bulk updates to the Team Development data. When schedules slip, the **Push past due features** and the **Push past due bugs** are used to change the due dates by a fixed number of days. When a developer moves on or is sick, you can reassign some or all of her work to another developer by using **Update Assignees**. The **Purge Data** utility is used to drop data. The Team Development entities, features, milestones, etc. can be purged separately if desired. Before doing this, having a clean backup is well advised.

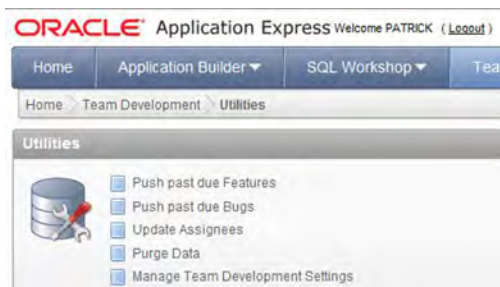
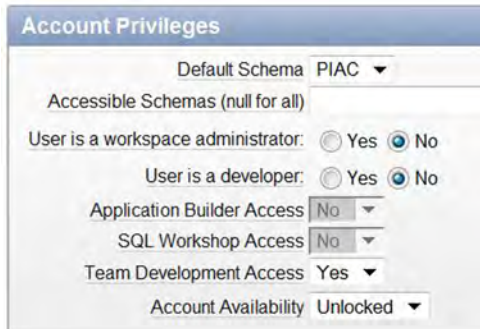


Figure 15–35. Utilities

User Roles for Team Development

Access to Team Development is useful for stakeholders both inside and outside the development team. The team lead and senior developers should have access to Team Development. Access by junior developers depends on the team's culture and trust level. Interested stakeholders who are outside the development team could be the Project Manager, test team, and the business analysts.

Access to APEX development environment is controlled in the APEX Administration area under the **Manage Users and Groups** menu. This area maintains the list of APEX users enabling them to access the workspace and by assigning their account privileges (see Figure 15–36). An outside stakeholder is set up without administrator and developer privileges; however, the Team Development Access field is set to **Yes**. When they log in to the APEX development environment, they see only the Team Development area and nothing else. There is only one problem at this time with this scenario: the outside stakeholders cannot see the applications in the Application dropdown lists. Hopefully this will be fixed in an upcoming patch.



Account Privileges	
Default Schema	PIAC ▼
Accessible Schemas (null for all)	
User is a workspace administrator:	<input type="radio"/> Yes <input checked="" type="radio"/> No
User is a developer:	<input type="radio"/> Yes <input checked="" type="radio"/> No
Application Builder Access	No ▼
SQL Workshop Access	No ▼
Team Development Access	Yes ▼
Account Availability	Unlocked ▼

Figure 15–36. Account Privileges for a Project Manager, Tester, or Business Analyst

Summary

Team Development is a software development tool that has been tailored to work in the APEX development environment. The five main entities (features, milestones, to-dos, bugs, and feedback) work together in a framework that is simple, extensible, and flexible. Teams that embrace agile software development methodologies will find APEX's Team Development tool to be a comfortable fit with their culture.

Index



■ Special Characters

&P200_SEARCH Highlight Words element, 135
&PX_ITEM_NAME substitution string syntax, 72
#COLUMN_HEADER# must have a value error message, 219
#LABEL# substitution variable, 212
#NAVIGATION_BAR# substitution variable, 75
#OWNER# replacement variable, 10
(&P1_ITEM_NAME.) substitution string syntax, 230
:APP_PAGE_ID substitution variable, 90
:APP_USER substitution variable, 231

■ A

About Application Express link, 356
About Database report, 29
About Database utility, 29
access control, 260–265
Access Control link option, 331
Access Control list, 262, 292–294
access controls, for websheets, 331
Access field, Team Development, 403
access log, for monitoring applications, 346–347
Access Log region, 351–352
accessibility region, 384
Account Login Control section, 360
Actions drop-down menu, 307
Actions menu, 167, 170–171, 189
Active Registered Users title, 330
activity log, for monitoring applications, 345–346

Add Child option, 26
Add File link option, 326
Add icon, Team Development home page, 379
Add link, APEX home page, 378
Add New Section link option, 326–327, 330
Add Object button, 316
Add option, Manage drop-down menu, 310
Add Parent option, 26
Add/Remove Columns, 122
Add Row option, Manage drop-down menu, 312
Admin page group, 338
Admin tab, 259
administration, 12, 30
Administration area, APEX, 403
Administration drop-down menu, 290, 292
Administration home page, 356
administration module, 367
Administration section, 361
Administration tab, 331, 355
Administrator role, 290
Aggregate action, 177–178
Allow Public Access drop-down menu, 294
Allow SQL Tags and Reports feature, 301
Allow SQL Tags and Reports option, 300–301
ALTER TABLE tickets ADD (publi_vflag VARCHAR2(1)) Tickets table, 232
alternate default reports, 323–324
Alternate option, 323
Analysis report's Interactive Report properties, 189
Annotation section, 311
annotations, in websheets, 311

- announcements, in Manage Service page, 360–361
- AnyChart engine, 204
- APEX, 1–5
 - defined, 1–2
 - forms, 99–101, 139–149
 - help, 149–151
 - item layout, 140–146
 - Master-Detail cleanup, 147–149
 - getting started, 4–5
 - access to instances, 4
 - SQL Developer, 5
 - web browsers, 5
 - history of, 2–4
 - ancient, 2
 - future of, 3–4
 - recent, 2–3
 - items, 129–130
 - bind variables, 130
 - built-in, 130
 - page versus application, 129
 - searchable reports, 132–139
 - and system designs, 34–36
 - business logic vs. user interface logic, 36
 - placement of database objects, 36
 - primary keys, 35
 - table definition and UI Defaults, 35
 - URL syntax, 131–132
- APEX Advisor, 347–348
- APEX API
 - APEX_UTIL.PUBLIC_CHECK_AUTHORIZATION, 265
- APEX Application Creation wizard, 83
- APEX Credentials authentication scheme, 254
- APEX Development Interface, 16
- APEX engine, 126
- APEX finder, 342–343
- APEX home page, in Team Development, 377–378
- APEX Instance settings, 15
- APEX Issue Tracker, 73, 230
- APEX modules, 12–30
 - Administration and Team Development, 30
 - Application Builder, 15–17
 - Application home page, 17
 - home page, 15–17
 - Home Page, 13–15
 - SQL Workshop, 17–29
 - Object Browser, 18–19
 - Query Builder, 24–28
 - SQL Commands Interface, 19–20
 - SQL Scripts Interface, 20–23
 - Utilities, 28–29
- APEX page item types, 129
- APEX schema, 339
- APEX session, 126–127
- APEX users, 8, 126
- APEX version, checking, 358
- APEX_040000 schema, 333, 363
- APEX_040000.WWV_FLOW_STEP_ITEMS table, 339
- APEX_APPLICATION_PAGE_ITEMS view, 340
- APEX_APPLICATION_PAGE_REGIONS view, 343
- APEX_DICTIONARY view, 341
- apex.oracle.com, 8
- APEX_UTIL.SET_AUTHENTICATION_RESULT procedure, 347
- APEX_UTIL.SET_CUSTOM_AUTH_STATUS procedure, 347
- apex_util.set_session_state procedure, 127
- APEX_WORKSPACE_ACCESS_LOG view, 346, 348
- APEX_WORKSPACE_ACTIVITY_LOG view, 345
- APEX_WORKSPACE_CLICKS log, 362
- API calling parameters, 396
- APP_ALIAS built-in item, 130
- APP_ID built-in item, 130
- APP_ID element, 131
- Application Build Status section, 364
- Application Builder, 15–17
- Application Builder home page, 14, 275
- Application button, 273
- Application Cache section, 363
- Application Content tab, 285
- Application Creation wizard, 70
- Application Express Account authentication scheme, 293–294
- Application Express Account option, 288
- Application Express option, 67
- Application Express scheme, 372
- Application Express workspace, 293
- application groups, 337
- Application Groups option, Tasks region, 337
- Application Groups page, 337
- application items, page items versus, 129
- Application Migration wizard, 17
- Application Models page, 364–366

- Application Properties page, 289, 294, 299
- Application region, 344
- Application tab, 273
- Application Utilities icon, 150
- applications, 8–11, 61–97
 - Application wizard, 61–71
 - database applications, 64–71
 - sample applications, 62
 - websheet applications, 63
 - public pages, 74
 - Static HTML regions, 72–73
- Apply Changes button, 209, 294, 317, 324, 345, 350–351, 360, 384
- ApplyMRD process, 162
- ApplyMRU process, 162
- APP_PAGE_ID built-in item, 130
- APP_PAGE_ID element, 131
- APP_SESSION built-in item, 130
- APP_SESSION element, 131
- APP_SESSION_ID, 127
- APP_USER built-in item, 130
- Arguments Must Have Checksum setting, 271
- As Default Report Settings, 186
- Ascending sort icon, 323
- Assigned Checked button, 337
- Assigned To attribute, 34
- Assigned To column, 175
- attribute field, Create Page wizard, 395
- authentication
 - custom schemes, 255–259
 - methods of, 68–69
 - overview, 254–255
- Authentication Schemes option, 255
- Authentication section, 294
- Authentication tab, Websheet Properties page, 289
- authentica_t_user function, 257
- authorization, 260–265
- Authorization region, Application Properties page, 294
- authorization scheme, 263, 266
- Autocommit, 20

B

- Based on existing application design model
 - option, 365
- BI Publisher, Oracle, 183
- Binary Large Object (BLOB), 151

- bind variables, 130
- BLOB (Binary Large Object), 151
- breadcrumbs, 79–84, 111, 118, 157, 163
- Breadcrumbs region, 80, 82, 252
- Break formatting, 182
- Browse button, 275
- Bug module, Team Development, 390
- bugs
 - defined, 376
 - easy, 390
 - hard, 391
 - in Team Development, 390–391
 - tracking, 391
- Build Option list, 351
- Build options, 348–352
 - applying, 351–352
 - configuring, 350
 - creating, 349–350
 - prompting for status of, 350–351
 - reasons for, 348–349
 - reporting on utilization of, 352
- business logic, vs. user interface logic, 36

C

- Calendar link, 380
- Calendar Region Name, 194
- Calendar tab, 379–380, 389
- calendars, 192–198
- calling parameters, API, 396
- Cancel button, 380
- CHARACTER_SET column, 151
- Chart action, 178–179
- Chart option, 327
- Chart page, 204
- Chart section type, 282
- chart sections, in websheets, 311
- charts, 198–206
- Check All option, 26
- Check Pages region, 347
- Checksum error message, 271
- Checksum Required - Session Level setting, 271
- CLEAN_SCHEMA script, 20
- Clear Cache element, 131
- click count logs, in Manage Meta Data region, 362
- Close Popup Window process type, 226
- Closed Date must be After Creation Date
 - validation name, 215

- Closed On attribute, 34
- Closed On Date must be Later than the Created Date message, 216
- CLOSED ticket status, 214
- ColSpan value, 144, 146
- Column Attributes page, 137
- Column Display Attributes section, 154
- Column Groups option, Manage drop-down menu, 310
- Column Link region, 196
- Column Name, 27
- Column Not Null option, 218
- Column option, 327
- Column Properties option, 310, 321
- Column Selector, 27
- columns
 - heading menu, 168–169
 - searching by, 169–171
 - Select column action, 172
- Columns and Function/Operators regions, 173
- Columns option, Manage drop-down menu, 310
- Columns Selected, 122
- comma-delimited data, 184
- Commands window, SQL, 28
- computations, 220–225
 - execution, 220–221
 - types of, 221
- Compute action, 176–177
- CONCRETE NOUNS, 33
- conditional security, 259–260
- conditions, 207
- Conditions region, 24
- Conditions Tab, 27–28
- Configuration region, 351
- confirmation dialog, 70–71
- Confirmation page, 303
- CONTACT_US procedure, 111, 114–115
- content of websheets, navigating, 283–284
- Contributor role, 290, 331
- Control Break action, 175
- Control Break option, 169
- Control Summary panel, 169, 175
- Control Summary region, 169–170
- Convert to SQL Based calendar option, 268
- Convert to SQL Query option, 134, 153
- Copy link, 383
- Copy option, Manage drop-down menu, 310
- Copy Region, 81

- Create a Ticket, 2 P2_SUBJECT (Subject), 107
- Create a Ticket for the Create Button Label, 104
- Create and Create Another button, 87, 331
- Create Breadcrumb Entry button, 252
- Create button, 40, 44, 62, 303, 337–338, 349–350, 393
- Create Computation option, 223
- Create Dynamic Action option, 239
- Create Entry button, 292, 317, 331
- Create Group button, 372
- Create List of Values, 96
- Create Multiple Users link, 370
- Create Multiple Users page, 370–371
- Create Opposite False Action check box, 240
- Create Page button, 327
- Create Page Item option, 132, 146, 221, 232
- Create Page wizard, 80, 392
- Create Region button, 236
- Create Report button, 301
- Create Report page, 302
- Create Report wizard, 162
- Create Section wizard, 303
- Create Tabular Form wizard, 161
- Create to create a new List of Values option, 95
- Create User button, 369
- Create Valid Users button, 371
- Create Validation option, 107, 254
- Create Websheet Database Objects link, 363
- Created By attribute, 34
- Created By field, 145
- Created On attribute, 34
- Created On field, 144–145
- CREATED_ON column, 53, 121, 162, 194
- CSV file, 122, 184
- CSV format, 184
- Current Open Issues title, 236
- Custom authentication scheme, 255, 293–294
- Custom Authentication Scheme name, 256
- Custom option, 289

■ D

- Dashboard page, 312
- dashboard region, Team Development, 378
- Dashboard section, 373
- Dashboard tab, 379, 389
- data, 267–270
 - loading with Data Workshop utility, 45–50
 - user maintenance, 249–254

- Data drop-down menu, 306
- data grid, LINK_TYPE, 288
- data grids, 318–322
 - applying constraints, 321–322
 - in websheets, 306–310
- Data Manipulation process type, 226
- Data option, 325
- Data Report page, 301
- Data section type, 282
- data sections, in websheets, 299–310
- Data Type Indicator, 27
- Data Workshop utility, loading data with, 45–50
- Database Account option, 68
- database administrator (DBA), 301
- Database Application link, 273
- database applications
 - custom, 65–71
 - authentication methods, 68–69
 - copying shared components, 67
 - final confirmation dialog, 70–71
 - laying out pages, 66
 - naming, 65
 - tabbing, 67
 - themes, 69
 - from spreadsheets, 64
- Database Column source type, 234
- Database Monitor utility, 29
- Database Schema Users authentication
 - scheme, 255
- Database session communication, 126
- databases
 - connections, sharing, 126–127
 - design of, extrapolating to, 33–34
 - objects, placement of, 36
- datagrid, LINK_TYPE, 288
- Date Format setting, 69
- DBA (database administrator), 301
- DEBUG element, 131
- Default Report option, 323
- default reports, alternate, 323–324
- Define Column Mapping region, 50
- Delete Checked button, 367
- Delete Column option, 320
- Delete Columns option, Manage drop-down
 - menu, 310
- Delete Data Grid option, Manage drop-down
 - menu, 310
- Delete icon, 169
- Delete Rows option, Manage drop-down menu,
 - 310
- Demo Application, 16
- DEMO_ORDERS table, 25–26
- deploying applications, 273–279
 - exporting, 273–274
 - importing, 275–277
 - supporting objects for, 277
- Description attribute, 34
- Detail page, 380–381, 383
- Details attribute, 34
- Details tab, 379
- Developer Activity and Click Count Logs menu
 - option, 361
- developer activity logs, in Manage Meta Data
 - region, 361
- Developer Data Modeler, SQL, 37
- development
 - APEX modules, 12–30
 - Administration and Team Development, 30
 - Application Builder, 15–17
 - Home Page, 13–15
 - SQL Workshop, 17–29
 - workspaces, 7–12
 - APEX users, 8
 - applications and schemas, 10–11
 - applications, pages, regions, and items, 8–9
- Development Interface, APEX, 16
- Development Only build option, 350–352
- Development page, To-do module, 389
- Development screen, To-do module, 389
- dictionary, 341–342
- Disable E-Mail Body dynamic action, 239, 241
- Display Point, 92
- do items in, to Team Development, 389
- documentation region, 384
- Download action, 183–185
- download and save option, 274
- Download attributes region, 183
- Drag & Drop Layout option, 141
- Drag & Drop Layout tool, 140
- drilldown functionality, in Team Development,
 - 380–382
- dynamic actions, 238–243
- dynamic LOVs, 95–97
- Dynamic option, 97
- Dynamic SQL, 232–238

■ E

- easy bugs, 390
- Easy Calendar wizard, 197
- Easy/Declarative calendar, 192
- Edit Access Control List button, 317
- Edit Authentication button, 289, 294
- Edit Definition link, Application region, 344
- Edit detail on separate page, 117
- Edit icon, 21, 121
- Edit link option, 286, 295, 298, 306, 324, 327, 378
- Edit page, 273, 285–286, 295–296, 298, 380, 383–384, 387
- Edit Page button, 394
- Edit Properties icon, 315
- Edit Report Attributes option, 119, 122–123, 135, 147, 158, 189–190
- Edit SQL Access and Suggested Objects button, 299
- Edit Table Defaults button, 58
- Ellison, Larry, 2
- Embed Tag column, 303
- EMP table, 343
- Enable Tracking Attributes region, 401
- enabling logging, 344–345
- End Users, 8, 32
- Entity Relationship Diagram (ERD), 37, 375
- Entries by List report, 85
- Entry Details page, 292
- ERD (Entity Relationship Diagram), 37, 375
- execution points
 - for computations, 220–221
 - for processes, 225
- Existing Table, 47
- Existing Trigger, 103, 117, 156, 251
- Export Application button, 274
- Export button, 273
- Export option, 351
- exporting applications, 273–274
- extrapolating to database designs, 33–34

■ F

- False Actions dynamic action, 238
- Features by Milestone tab, in Team Development, 388
- features, in Team Development, 383–386
 - Features tab, 383–385
 - History tab, 385–386
 - Progress Log tab, 386

- Features tab, in Team Development, 383–385
- Feedback link, 393
- Feedback module, in Team Development, 391–399
 - configuring, 392–394
 - creating feedback page, 394–397
 - responses to feedback, 398
 - using between workspaces, 398–399
 - viewing feedback, 397–398
- Feedback page, 393–394
- Feedback Page option, Create Page wizard, 392
- File Utilization page, 366–367
- FILENAME column, 151
- FILE_NAME column, 152
- Files annotation, 311
- Fill option, Manage drop-down menu, 310
- Filter action, 172–173
- Filter button, 340
- filter fields, 379
- Filter option, 323
- filter page, 340
- Filters action, 176
- Find icon, 342
- Finder drop-down menu, 170
- Finder drop down, Search Bar, 166
- Firefox Web Developer Toolbar add-in, 109
- First row contains column names, 48, 50
- Flashback action, 180
- Flashback check box, 189
- f_login function, 316
- Form on a Procedure, 100
- Form on a SQL Query, 100
- Form on a Table or View, 100
- Form on a Table or View with Report, 100
- Form on a Web Service, 100
- Form page, 307
- Form Pagination process type, 226
- Format Action option, 170
- forms
 - APEX, 99–101, 139–149
 - help, 149–151
 - items, 129–130, 140–146
 - Master-Detail cleanup, 147–149
 - searchable reports, 132–139
 - URL syntax, 131–132
 - BLOB columns, 151
 - calendars, 192–198
 - charts, 198–206

- Forms on a Procedure, 111–115
 - creating, 111–113
 - modifying, 113–114
- Forms on a Table, 101–111
- interactive reports, 162–192
 - Aggregate action, 177–178
 - Chart action, 178–179
 - column heading menu, 168–169
 - Compute action, 176–177
 - Control Break action, 175
 - Download action, 183–185
 - Filter action, 172–173
 - Flashback action, 180
 - Group By action, 179
 - Help action, 181
 - Highlight action, 176
 - modifying, 185–191
 - Reset action, 181
 - restricting functionality, 167–168
 - running, 165–167
 - Save Report action, 180
 - searching by column, 169–171
 - Select column action, 172
 - Sort interface, 174
 - Subscription action, 182–183
- Master-Detail reports and forms, 115–124
 - creating, 115–119
 - modifying, 119–124
- session state, 125–128
 - setting and retrieving, 127
 - sharing database connections, 126–127
 - viewing, 127–128
- tabular, 155–162
- From Scratch input method, 321
- From Scratch option, 64, 75
- From Spreadsheet option, 64
- Function Returning Boolean, 215

■ G

- Gantt chart, 383
- Generate DDL wizard, 29
- Get Next or Previous Primary Key Value
 - process, 269
- get_status ('OPEN'), 160
- globalization region, 384
- Go button, 307
- Goals section title, 327
- Grid Edit of all Item Help Text option, 150

- Grizzlies Soccer application name, 314
- Grizzlies_Schedule.csv file, 319
- Group By action, 179
- groups
 - assigning users to, 373
 - creating, 372
- GUI designs, 380

■ H

- hard bugs, 391
- help, 149–151
 - Help Text, 150–151
 - for websheets, 285–286
- Help action, 181
- help desk application
 - dynamic action in, 238
 - process in, 226–230
- Help link, 15, 285, 364
- Help page tabs, 297
- Help section, 297
- helpdesk_spreadsheet.xls file, 47
- HEX notation, 176
- Hichwa, Mike, 2
- Hide and Show Region, 149
- Highlight action, 176
- History links, 312
- History option, Manage drop-down menu, 310
- History tab, in Team Development, 385–386
- Home Page List, 85, 90
- home pages
 - APEX modules, 13–15
 - Application, 17
 - Application Builder, 15–17
- Home parent page, 327
- Home region, 73
- HTML DB, 281
- HTML format, 184
- http.p output, 231
- HTTP.P procedure, 339
- http://apex.oracle.com, 4
- http://otn.oracle.com/apex, 4

■ I, J

- I_dont_know OR 'x' = 'x password, 130
- image, LINK_TYPE, 288
- Import button, 275
- import/export facility, APEX, 377

- Important News text section, 286
- Important News title, 324
- importing applications, 275–277
- Inline in Notification error display location, 215
- Install option, 276
- Instance Administrators, 8
- instance information, viewing, 356–358
- Instance settings, APEX, 15
- instances, access to, 4
- integration, of Oracle SQL Developer, 353–354
- interactive edit section, 309
- Interactive Report (IR), 16
- Interactive Report option, 163, 340
- Interactive Reports Settings page
 - Saved Reports section, 367
 - Subscriptions section, 368
- interface, of Team Development, 377–383
 - APEX home page, 377–378
 - common design elements in, 379–380
 - drilldown functionality of, 380–382
 - tagging in, 382–383
 - Team Development home page, 378–379
- IR (Interactive Report), 16
- is null condition, 240
- item layout, 140–146
- Item Level Validation option, 107, 210
- Item Names element, 131
- Item Not NULL option, 212
- Item Utilities option, 150
- Item Value computation type, 221
- Item Values element, 131
- ITEM_HELP_TEXT column, 340
- ITEM_NAME column, 340
- items, 8–9

■ K

- Kallman, Joel, 2
- Kyte, Tom, 2

■ L

- Language setting, 69
- Last Updated process, 228
- LAST_UPDATED column, 151, 226–227
- LDAP authentication scheme, 254, 294
- LDAP (Lightweight Directory Access Protocol), 289
- LDAP option, 289, 293

- Lightweight Directory Access Protocol (LDAP), 289
- Link Column attributes, 190
- Link to Custom Target, 190
- LINK_NAME keyword, 288
- Links annotation, 311
- LINK_TARGET keyword, 288
- LINK_TYPE keyword, 287–288
- List of Values Definition, 261
- List of Values option, Manage drop-down menu, 310
- lists
 - overview, 84–88
 - regions, 88–92
- Lists maintenance screen, 85
- Lists of Values. *See* LOVs
- Load Data button, 49
- loading
 - data, with Data Workshop utility, 45–50
 - SQL Scripts, 53–55
- LOADPSP program, 1–2
- Lock Checked button, 335
- Log as Bug button, 397
- Log as Feature button, 397
- Log as To Do button, 397
- logging, enabling, 344–345
- logic, business vs. user interface, 36
- Login link, 295
- Login page, 349–350
- Lookup Table, 52
- lookup tables, creating, 50–52
- LOVs (Lists of Values), 93–97, 261
 - configurable, 384, 386
 - dynamic, 95–97
 - static, 94–95

■ M

- Manage button, 319
- Manage Click Count Log, 362
- Manage drop-down menu, 308, 310
- Manage Export Repository link, 367
- Manage Groups region, 373
- Manage Links feature, in Team Development, 399–400
- Manage menu option, 309
- Manage Meta Data region, 361–368
 - Application Build Status section, 364
 - Application Cache section, 363

- Application Models page, 364–366
- click count logs in, 362
- developer activity logs in, 361
- File Utilization page, 366–367
- Interactive Reports Settings page
 - Saved Reports section, 367
 - Subscriptions section, 368
- Session State page
 - managing preferences, 363
 - managing session state, 362–363
- Websheet Database Objects section, 363
- Manage Multiple Tickets button, 264
- Manage Multiple Tickets option, 157–158, 217
- Manage Multiple Tickets report, 268
- Manage News link, in Team Development, 400
- Manage Script Results page, 22–23
- Manage Service icon, 358
- Manage Service page, 358–361
 - announcements in, 360–361
 - workspace preferences in, 359–360
- Manage Subscriptions page, 368
- Manage Table Dictionary button, 57
- Manage Tickets form, 125
- Manage Unassigned link, Application Groups page, 337
- Manage User Groups link, Tasks region, 372
- Manage Users and Groups menu, APEX, 403
- Manage Users and Groups page, 369–370
- Manage Users page, 369–373
 - assigning users to group, 373
 - creating groups, 372
 - creating multiple users, 370–371
 - creating single user, 369
- markup syntax, for websheets, 287–288
- Master-Detail Form, 100, 115, 119, 138–139
- Master-Detail reports and forms, 115–124, 138–139
 - cleanup, 147–149
 - creating, 115–119
 - modifying, 119–124
- Maximum Levels input, 298
- Milestone Details tab, in Team Development, 388
- milestones, in Team Development, 386–388
 - Features by Milestone tab, 388
 - Milestone Details tab, 388
 - Milestones tab, 387–388
- Milestones tab, in Team Development, 387–388

- MIME_TYPE column, 151–152
- MIME_TYPE Settings region, 152
- modules, 12–30
 - Administration and Team Development, 30
 - Application Builder, 15–17
 - Application home page, 17
 - home page, 15–17
 - Home Page, 13–15
 - SQL Workshop, 17–29
 - Object Browser, 18–19
 - Query Builder, 24–28
 - SQL Commands Interface, 19–20
 - SQL Scripts Interface, 20–23
 - Utilities, 28–29
- Monitor Activity icon, 361
- Monitor Activity page, 312
- Monitor Activity section, 361–362, 373
- monitoring applications
 - access log for, 346–347
 - activity log for, 345–346
 - enabling logging, 344–345
- month_calendar process, 198

N

- Name column, 306
- naming patterns, 387
- navigation
 - breadcrumb entries, 79–84
 - list regions, 88–92
 - lists, 84–88
 - LOVs, 93–97
 - configurable, 384, 386
 - dynamic, 95–97
 - static, 94–95
 - navigation bar entries, 75–78
 - Page Zero, 78–79
 - template positions, 92–93
 - user maintenance, 245–249
 - websheets, 283–285
 - content, 283–284
 - structure, 285
- navigation bar
 - conditions, 77
 - entries, 75–78
- Navigation section type, 282, 328, 330
- navigation sections, in websheets, 298–299
- Navigation to URL option, 76
- New Owner item, 384
- New Page link, 326

New Section links, 303, 325
 News region, 15, 378
 News section, 361
 Next button, Create Page wizard, 392
 No Authentication option, 68
 No Tabs option, 67
 None authentication scheme, 255
 Notes annotation, 312

■ 0

Object Browser
 creating objects with, 39–45
 overview, 18–19
 Object Reports, 29
 objects
 creating with Object Browser, 39–45
 database, placement of, 36
 One Level Tabs option, 67
 Open Door authentication scheme, 255
 optimistic locking, 334
 Oracle BI Publisher, 183
 Oracle PL/SQL engine, 91
 Oracle Single Sign On authentication scheme, 254
 Oracle SQL Developer
 integration of, 353–354
 refactoring support in, 354

■ P

(P1_ITEM_NAME) bind variable syntax, 132, 230
 P2_CREATED_BY item, 106, 108
 P2_STATUS_ID item, 106
 P2_SUBJECT is NOT NULL item, 107–108
 P2_SUBJECT label template, 106
 P3_BODY item, 114–115, 241
 P3_FROM item, 113–115, 239
 P200_GO Button name, 133
 P200_SEARCH filter, 134
 P200_SEARCH item, 132–133, 137
 P200_STATUS_ID field, 201
 P200_STATUS_ID item, 201–202, 204
 P210_ASSIGNED_TO item, 142–143
 P210_CLOSED_ON item, 141, 211
 P210_CREATED_BY item, 142–143
 P210_CREATED_ON item, 140, 142
 P210_DAYS_OPEN item, 222

P210_DESCR item, 145, 208
 P210_PUBLIC_FLAG item, 233
 P210_STOP_START_TABLE item, 146
 P210_SUBJECT item, 144
 P210_TICKET_ID item, 190, 222
 P210_TICKET_STATUS_ID, 202
 P220_ATTACHEMENT item, 152
 P220_CREATED_BY item, 148
 P220_CREATED_ON item, 147
 P220_DETAILS item, 147
 P220_TICKET_DETAILS_ID computation, 138
 P220vTICKET_ID item, 147
 P610_PASSWORD element, 254
 P610_PASSWORD is not null validation name, 254
 P610_PASSWORD item type, 253
 Page control on this page option, 134, 209, 215
 Page Creation wizard, 83
 page groups, 338
 Page Groups page, 338
 Page in this Application target, 76
 page items, versus application items, 129
 Page Level Validation option, 215
 page, LINK_TYPE, 288
 page locks, 333–336
 administration of, 336
 optimistic locking, 334
 pessimistic locking, 334–335
 unlocking page, 335
 Page Locks option, 336
 Page Locks report, 336
 Page Navigation type, 330
 Page Processing column, 192
 Page Processing section, 197
 Page Render region, 335
 Page Rendering region, 191, 197, 204
 page sections, 324–330
 Page Sections link, 296
 page specific utilities, 352–353
 Page Specific Utilities region, 354
 Page tab, 324
 Page Zero option, 78–79
 pages
 laying out, 66
 overview, 8–9
 public, 74
 Parent Tabs node, 263
 parse as schema, 10–11

Password attribute, 33
 Perform Check button, 347
 pessimistic locking, 334–335
 PL/SQL (anonymous block), 230
 PL/SQL code, 127
 PL/SQL engine, 91
 PL/SQL Expression, 140, 147, 160, 221
 PL/SQL Function Body, 90, 106, 213, 221, 265
 PL/SQL option, 227
 PL/SQL procedure, 114
 PL/SQL process type, 226
 PL/SQL regions, 230–232
 PL/SQL Server Pages (PSPs), 2
 PL/SQL Web Toolkit, 2
 Player of the Week title, 326
 Populated from a new sequence radio button, 41
 popupurl, LINK_TYPE, 288
 p_password varchar2(255) parameter, 257
 preferences
 for user sessions, in Session State page, 363
 workspace, in Manage Service page, 359–360
 Primary Key option, 41
 primary keys, 35
 Printer Friendly element, 131
 problem solving, 31–38
 identifying system requirements, 31–34
 broken systems, 32
 extrapolating to database designs, 33–34
 system designs and APEX, 34–36
 business logic vs. user interface logic, 36
 placement of database objects, 36
 primary keys, 35
 table definition and UI Defaults, 35
 translating theory to practice, 37–38
 processes, 225–230
 execution points, 225
 in help desk application, 226–230
 types of, 226
 programmatic elements, 207–243
 computations, 220–225
 execution, 220–221
 types of, 221
 conditions, 207
 dynamic actions, 238–243
 in help desk application, 238
 parts of, 238

Dynamic SQL, 232–238
 PL/SQL regions, 230–232
 processes, 225–230
 execution points, 225
 in help desk application, 226–230
 types of, 226
 validations, 207–220
 item level, 209–214
 page level, 214–217
 required values, 208–209
 tabular form, 217–220
 Progress Log tab, in Team Development, 386
 Project Marvel, 281
 Properties option, Manage drop-down menu, 310
 PSPs (PL/SQL Server Pages), 2
 Public Flag label name, 233
 PUBLIC_FLAG column, 234–235
 Purge Click Log button, 362
 Purge Data utility, 402
 Purge Developer Log button, 361
 p_username varchar(255) parameter, 257
 Push past due bugs, 402
 Push past due features, 402

■ Q

queries, writing for charts, 199
 Query Builder, 19, 24–28
 Query Definition sub-tab, 122
 Quick Menu, 53
 q'X string X' basic syntax, 238

■ R

RAD (Rapid Application Development), 1, 375
 read-only items, 265–266
 Reader role, 290
 Recycle Bin utility, 29
 Redirect to Page in this Application setting, 160
 Refactor option, 354
 refactoring support, in Oracle SQL Developer, 354
 Region Buttons node, 160
 Region Definition, 205
 Region on this page option, 149, 235
 Region Template Position #CREATE# setting, 158, 160
 regions, 8–9
 REGION_SOURCE column, 343

- Release filter, Team Development dashboard region, 378
- Release summary, in Team Development, 401–402
- Remaining Games report, 324
- Remaining Practices report, 324
- Remove option, 26
- Replace option, Manage drop-down menu, 310
- Report Attributes tab, 137, 153
- Report Data page, 303
- Report feature, 299
- report, LINK_TYPE, 288
- Report Region Title, 58
- Report Select List, Search Bar, 166
- reporting, on utilization of Build options, 352
- reports
 - alternate default, 323–324
 - APEX, 99–101, 139–149
 - help, 149–151
 - items, 129–130, 140–146
 - Master-Detail cleanup, 147–149
 - searchable reports, 132–139
 - URL syntax, 131–132
 - BLOB columns, 151
 - calendars, 192–198
 - charts, 198–206
 - creating, 199–204
 - writing queries for, 199
 - Forms on a Procedure, 111–115
 - creating, 111–113
 - modifying, 113–114
 - Forms on a Table, 101–111
 - interactive, 162–192
 - Aggregate action, 177–178
 - Chart action, 178–179
 - column heading menu, 168–169
 - Compute action, 176–177
 - Control Break action, 175
 - Download action, 183–185
 - Filter action, 172–173
 - Flashback action, 180
 - Group By action, 179
 - Help action, 181
 - Highlight action, 176
 - modifying, 185–191
 - Reset action, 181
 - restricting functionality, 167–168
 - running, 165–167

- Save Report action, 180
- searching by column, 169–171
- Select column action, 172
- Sort interface, 174
- Subscription action, 182–183
- Master-Detail reports and forms, 115–124
 - creating, 115–119
 - modifying, 119–124
- session state, 125–128
 - setting and retrieving, 127
 - sharing database connections, 126–127
 - viewing, 127–128
- tabular forms, 155–162
- in websheets, 299–303
- Reports drop-down list, 307
- Reports menu, 180
- Reports Region, 163, 250
- Reports select list, 189
- REQUEST element, 131
- Required with Help, 106, 233
- Reset action, 181
- Reset Pagination type, 134, 226
- Results button, 340
- Results page, 286, 328–329
- Results report, 324
- Results Tab, 27
- RETURN authenticate_user, 256
- return f_login statement, 316
- RETURN get_status('OPEN'), 106
- Row Filter interface, 177
- Rows option, Manage drop-down menu, 310
- Rows Per Page selector, Search Bar, 166
- Run icon, 22
- Run On Demand Process process type, 226
- Run Script wizard, 22
- Run Stored Procedure, 114
- running SQL Scripts, 53–55

■ S

- sample applications, 62
- Save Public Report check box, 189
- Save Report action, 180
- Save Report check box, 189
- Saved Report Settings drop-down list, 303
- Saved Reports section, in Interactive Reports
 - Settings page, 367
- Saved SQL Tab, 27
- Scale XY Center chart animation, 201

- Schedule option, 325
- Schedule page, 328–329
- Schema Comparison utility, 29
- Schema select list, 47
- Schema Selection dialog, 17
- schemas, 10–11
- Scripts Repository, SQL, 21
- Search Application feature, 343–344
- Search Application field, 343
- Search Bar, 169
- Search field, 166, 170, 382
- Search function, 170
- searching in APEX, 342–344
 - with APEX finder, 342–343
 - Search Application feature, 343–344
- section, LINK_TYPE, 288
- Section Navigation navigation type, 328
- sections, in websheets, 295–311
 - chart sections, 311
 - data grids, 306–310
 - data sections, 299–310
 - navigation sections, 298–299
 - reports, 299–303
 - text sections, 295–297
- security, 245–271
 - authentication
 - custom schemes, 255–259
 - overview, 254–255
 - authorization, access control and, 260–265
 - conditional, 259–260
 - data, 267–270
 - read-only items, 265–266
 - session state protection, 270
 - user maintenance
 - data entry, 249–254
 - navigation, 245–249
- security region, 384
- Seed Item Help Text option, 150
- Select a Tech - Null display value, 159
- Select a User - Null display value, 160
- Select column action, 172
- Select Column for Join option, 26
- Select Columns menu, 185
- Select Columns option, 323
- Select Columns screen, 340
- select count query, 330
- select * from DOUG.emp, 10
- select * from emp, 10
- select * from #OWNER#.emp, 10
- SELECT * FROM tickets_v, 164
- select statement, 330
- Send Email process type, 226
- Sequence input, 298
- Series element, 204
- session state, 125–128
 - protection, 270
 - setting and retrieving, 127
 - sharing database connections, 126–127
 - viewing, 127–128
- Session State page
 - managing preferences, 363
 - managing session state, 362–363
- Session State process type, 226
- Session State Protection option, 270
- Set Column Values option, Manage drop-down
 - menu, 310
- Set Workspace Preferences icon, 359
- Set Workspace Preferences page, 359–360
- settings area, Team Development, 384
- shared components, copying, 67
- Shared Components section, 75, 83–84, 137, 192
- Show checkbox option, 120, 158
- Show filter, Team Development dashboard
 - region, 378
- Show/Hide Columns option, 26
- shuttle tag, 382
- Single Row View, 190
- Single Sign On authentication scheme, Oracle, 254, 294
- Single Sign-On option, 289, 293
- Sort box, 120
- Sort checkbox, 123
- Sort interface, 174
- Source - Region Source section, 73
- Specify the True Action option, 240
- Spreadsheet Data link, 49
- Spreadsheet Data option, 47
- SQL Calendar, 192
- SQL Commands Interface, 19–20
- SQL Developer Data Modeler, 37
- SQL Developer, Oracle, 353–354
 - integration of, 353–354
 - overview, 5
 - refactoring support in, 354
- SQL Expression computation type, 221
- sql, LINK_TYPE, 288

- SQL Query prompts, 66
- SQL Query Report Source, 302
- SQL Query (Return colon separated value)
 - computation type, 221
- SQL Query (return single value) computation
 - type, 221, 223
- SQL Scripts icon, 53
- SQL Scripts Interface, 20–23
- SQL Scripts, loading and running, 53–55
- SQL Scripts Repository, 21
- SQL statement, 130, 224
- SQL Syntax link, 52
- SQL Tab, 27
- SQL Tags, 330
- SQL Workshop, 17–29, 39–59
 - creating lookup tables, 50–52
 - creating objects with Object Browser, 39–45
 - loading and running SQL Scripts, 53–55
 - loading data with Data Workshop utility, 45–50
 - Object Browser, 18–19
 - Query Builder, 24–28
 - SQL Commands Interface, 19–20
 - SQL Scripts Interface, 20–23
 - UI defaults, 56–59
 - utilities, 28–29
- SQL Workshop module, 360
- SQL Workshop tab, 47, 53, 56
- sqlvalue
 - LINK_TYPE, 288
 - SQL tag, 330
- Standard Dynamic Actions wizard, 243
- Starting Page input, 298
- Static Assignment computation type, 221
- STATIC data option, 94
- Static HTML regions, 72–73
- static LOVs, 94–95
- STATIC, Y,N List of Values query, 234
- STATIC2 data option, 94
- Status attribute, 34
- Status chart region, 205
- STATUS column, 122
- Status column, 175, 186
- status, of Build options, 350–351
- STATUS_ID column, 109, 122, 162
- STATUS_ID maps, 106
- STATUS_LOOKUP table, 52, 122
- Stop Table, 146

- structure, of worksheets
 - navigating, 285
 - overview, 281–282
- Subject attribute, 34
- Subject column, 186
- SUBJECT column, 196
- Subject is Not Null validation name, 218
- Submit Feedback button, 397
- Subscription action, 182–183
- Subscription check box, 189
- Subscriptions section, in Interactive Reports
 - Settings page, 368
- Summary Page, 100
- supporting objects, 277
- Supporting Objects Scripting feature, 277
- Synchronize< button, 57
- SYSDATE reference point, 323
- system designs, and APEX, 34–36
 - business logic vs. user interface logic, 36
 - placement of database objects, 36
 - primary keys, 35
 - table definition and UI Defaults, 35
- system requirements, 31–34
 - broken systems, 32
 - extrapolating to database designs, 33–34

■ T

- tabbing, 67
- Table Actions, 26
- table definition, 35
- Table Dictionary, 56
- Table option, 40
- Table Report Source, 302
- tables
 - defining UI Defaults for, 56–59
 - lookup, creating, 50–52
- TableView Owner, 156
- Tabular Forms, 100, 155–162, 217
- T_ADMIN (Users), 251, 260
- tagging in Team Development, 382–383
- Tags annotation, 311
- Tasks region, 297, 337, 356, 372
- Team Actions links, 378
- Team Actions utilities, in Team Development, 399–402
 - Manage Links feature, 399–400
 - Manage News link, 400
 - Release summary, 401–402

- Team Development Settings page, 401
- Utilities page, 402
- Team Development, 375–403
 - and administration, 30
 - bugs in, 390–391
 - to do items in, 389
 - features in, 383–386
 - Features tab, 383–385
 - History tab, 385–386
 - Progress Log tab, 386
 - Feedback module in, 391–399
 - configuring, 392–394
 - creating feedback page, 394–397
 - responses to feedback, 398
 - using between workspaces, 398–399
 - viewing feedback, 397–398
 - interface of, 377–383
 - APEX home page, 377–378
 - common design elements in, 379–380
 - drilldown functionality of, 380–382
 - tagging in, 382–383
 - Team Development home page, 378–379
 - milestones in, 386–388
 - Features by Milestone tab, 388
 - Milestone Details tab, 388
 - Milestones tab, 387–388
 - overview, 375–377
 - Team Actions utilities, 399–402
 - Manage Links feature, 399–400
 - Manage News link, 400
 - Release summary, 401–402
 - Team Development Settings page, 401
 - Utilities page, 402
 - user roles for, 403
- Team Development Settings page, in Team Development, 401
- technicians, 32
- Template Based Button, 160
- template positions, 92–93
- testing region, 384
- Text section type, 281
- text sections, in websheets, 295–297
- themes, 69
- Ticket Details form, 125
- Ticket Details ID attribute, 34
- Ticket Details region, 153
- Ticket Details report attributes, 147
- Ticket ID attribute, 34

- Ticket Id column, 186
- Ticket Items section, 221
- TICKET noun, 33
- ticket record, 267
- TICKET_ACTIVITY_V view, 267
- TICKET_DETAILS process, 139
- TICKET_DETAILS table, 44, 115, 119, 139, 149, 152
- TICKET_DETAILS_ID, 44, 138
- TICKET_ID column, 122, 147
- TICKET_ID item, 43, 45, 103, 109–110, 147
- TICKET_ID (NUMBER) option, 41
- TICKET_ID Unique Column, 164
- Tickets by Status name, 188
- TICKETS columns, 138
- TICKETS link, 57
- Tickets region, 122–123, 229
- Tickets report, 124
- TICKETS row, 110
- TICKETS tab, 47
- TICKETS table, 43, 47, 110, 115, 119, 138, 149, 162, 226, 267
- TICKET_SECURE_V view, 267
- TICKETS_ID value, 138
- TICKETS_SECURE_V table, 270
- TICKETS.STATUS_ID column, 123
- TICKET_V view, 267
- Title input, 298
- To-do module, 389
- To-do Report page, 389
- To-dos tab, 389
- Toggle Checkboxes option, Manage drop-down menu, 310
- tool bar, 189
- Tracker PL/SQL region, 231
- tracking bugs, 391
- transactional mode, 20
- True Actions dynamic action, 238
- TS1 (Home, Tickets, Analysis), 193, 200
- TUSERS object, 316
- Two Level Tabs option, 67

■ U

- UI (User Interface) Defaults, 35, 56–59
- #SUBJECT# Column Format, 194
- UNDO_RETENTION parameter, 180

- Uniform Resource Locator (URL) syntax, 131–132
- Unlock Checked button, 335
- unlocking page, 335
- Upcoming Games and Practices title, 325
- Updateable Columns, 156
- url, LINK_TYPE, 288
- URL (Uniform Resource Locator) syntax, 131–132
- Use Tab to T_HOME, 103, 112
- Use Tab to T_TICKETS, 157
- user authentication of websheets, 288–289
- user authorization of websheets, 290–295
- User Groups region, 373
- User ID attribute, 33
- user interface logic, vs. business logic, 36
- user interface region, 384
- User Interface (UI) Defaults, 35, 56, 59
- User is Authenticated (not public) setting, 78, 259
- User is the Public User (user has not authenticated), 77
- User Language Preference setting, 69
- user maintenance
 - data entry, 249–254
 - navigation, 245–249
- User Name attribute, 33
- USER noun, 33
- user roles, for Team Development, 403
- USER_ID, 250–251
- USER_NAME, 37, 250
- users
 - APEX, 8
 - assigning to groups, 373
 - creating multiple, 370–371
 - creating single, 369
- Users and Groups icon, 369
- USERS table, 255, 259, 279
- Utilities, 28–29
- Utilities page, in Team Development, 402
- Utilization tab, 352

■ V

- Validating node, 107
- Validation option, 210, 310
- validations, 207–220
 - item level, 209–214
 - page level, 214–217

- required values, 208–209
- tabular form, 217–220
- Value of Item in Expression 1 is NOT NULL
 - condition type, 222
- value required validation, 208–209
- VARCHAR column types, 51
- Vertical Unordered List with Bullets, 85
- View Lock icon, 336
- View Results icon, 23
- views, 339–341
- (v('P1_ITEM_NAME')) v function, 230

■ W

- WBS (work breakdown structure), 375, 383
- web browsers, 5
- Web Developer Firefox add-on, 144
- Web Services process type, 226
- Web Toolkit, PL/SQL, 2
- WebDB, 2
- Websheet application type, 314
- Websheet Applications, 16, 63
- Websheet Database Objects section, 363
- Websheet Properties page, 289
- websheets, 281–331
 - access controls, 331
 - adding content to, 317–330
 - alternate default reports, 323–324
 - data grids, 318, 321–322
 - page sections, 324–330
 - players data, 322
 - SQL Tags, 330
 - administrating, 312
 - annotations in, 311
 - creating and configuring, 314–317
 - help for, 285–286
 - markup syntax for, 287–288
 - navigating, 283–285
 - content, 283–284
 - structure, 285
 - sections in, 295–311
 - chart sections, 311
 - data grids, 306–310
 - data sections, 299–310
 - navigation sections, 298–299
 - reports, 299–303
 - text sections, 295–297
 - setup, 313–314
 - structure of, 281–282

- user authentication of, 288–289
 - user authorization of, 290–295
- where clauses, 232
- Wolf, Patrick, 347
- work breakdown structure (WBS), 375, 383
- Workshop Utilities section, SQL, 28
- Workspace Administrators, 8
- Workspace Home Page, 13
- workspaces, 7–12, 355–374
 - APEX users, 8
 - applications and schemas, 10–11
 - applications, pages, regions, and items, 8–9
 - checking APEX version, 358
 - Dashboard section, 373
 - feedback in Team Development between, 398–399
 - Manage Meta Data region, 361–368
 - Application Build Status section, 364
 - Application Cache section, 363
 - Application Models page, 364–366
 - click count logs in, 362
 - developer activity logs in, 361

- File Utilization page, 366–367
 - Interactive Reports Settings page, 367–368
 - Session State page, 362–363
 - Websheet Database Objects section, 363
- Manage Service page, 358–361
 - announcements in, 360–361
 - workspace preferences in, 359–360
- Manage Users page, 369–373
 - assigning users to group, 373
 - creating groups, 372
 - creating multiple users, 370–371
 - creating single user, 369
- Monitor Activity section, 373
- preferences, in Manage Service page, 359–360
- viewing instance information, 356–358

■ X, Y, Z

- XML Data, 46
- X-Y coordinates, 383

Beginning Oracle Application Express 4



Doug Gault
Karen Cannell
Patrick Cimolini
Martin D'Souza
Timothy St. Hilaire

Apress®

Beginning Oracle Application Express 4

Copyright © 2011 by Doug Gault, Karen Cannell, Patrick Cimolini, Martin D'Souza, Timothy St. Hilaire

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): 978-1-4302-3147-9

ISBN-13 (electronic): 978-1-4302-3148-6

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

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: Jonathan Gennick

Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Frank Pohlmann, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Coordinating Editor: Anita Castro

Copy Editor: Mary Behr

Compositor: MacPS, LLC.

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/info/bulksales.

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.

To my better half, David, for staying sane throughout all of the insanity and for supporting me no matter what harebrained scheme I was trying to hatch.

—Doug Gault

To Gary and Kali, who tolerate my brain on APEX.

—Karen Cannell

I would like to dedicate this book to all those who have worked in computer technology and taken the time to share their experiences and knowledge. Only by teaching each other do we truly grow.

—Tim St. Hilaire

Contents

Contents at a Glance	iv
About the Authors	xv
Acknowledgments	xvii
■ Chapter 1: An Introduction to APEX 4	1
What is APEX?	1
A Brief History of APEX.....	2
Ancient History	2
More Recent History.....	2
APEX 4 and the Future	3
What You Need to Get Started.....	4
Access to an APEX Instance.....	4
Web Browser.....	5
SQL Developer.....	5
Summary.....	5
■ Chapter 2: A Developer's Overview	7
The Anatomy of a Workspace	7
APEX Users.....	8
Applications, Pages, Regions, and Items	8
Workspaces, Applications, and Schemas	10
A Final Word on Workspaces	11
A Tour of the APEX Modules.....	12
The Home Page.....	13
Application Builder.....	15

SQL Workshop.....	17
Administration and Team Development.....	30
Summary.....	30
Chapter 3: Identifying the Problem & Designing the Solution.....	31
Identifying System Requirements	31
Never a Clean Slate.....	31
A Broken System.....	32
How Do You Fix Things?.....	32
System Design with APEX in Mind	34
Table Definition and User Interface Defaults	35
APEX and Primary Keys.....	35
Business Logic vs. User Interface Logic	36
Placement of Database Objects	36
Translating Theory to Practice	37
Summary.....	38
Chapter 4: SQL Workshop.....	39
Creating Objects with the Object Browser	39
Loading Data with the Data Workshop Utility.....	45
Creating a Lookup Table	50
Loading and Running SQL Scripts.....	53
User Interface Defaults	56
Defining UI Defaults for Your Tables	56
Summary.....	59
Chapter 5: Application and Navigation.....	61
The Create Application Wizard	61
Sample Applications	62
Worksheet Applications.....	63
Database Applications from Spreadsheets	64
Database Applications from Scratch.....	65

Static HTML Regions	72
Public Pages	74
Navigation Bar Entries	75
Page Zero	78
Breadcrumb Region	79
Breadcrumb Entries	83
Lists.	84
List Regions	88
Template Positions	92
Lists of Values	93
Static List of Values	94
Dynamic List of Values	95
Summary.	97
■ Chapter 6: Forms and Reports – The Basics	99
APEX Forms.	99
Form on a Table	101
Creating a Form on a Table	101
Modifying a Form on a Table.....	105
Looking Behind the Scenes.....	109
Form on a Procedure	111
Creating a Form on a Procedure	111
Modifying a Form on a Procedure.	113
Looking Behind the Scenes.....	114
Master-Detail Report and Form	115
Creating a Master-Detail Report and Form	115
Modify a Master-Detail Report.	119
Session State	125
Understanding Session State	126
Sharing Database Connections	126
Setting and Retrieving Session State.	127

Viewing Session State.....	127
APEX Items.....	129
Page versus Application items.....	129
The Importance of Bind Variables.....	130
Built In Items.....	130
APEX URL Syntax.....	131
Searchable APEX Reports.....	132
Creating a Searchable APEX Report.....	132
Looking Behind the Scenes – APEX Report.....	136
Looking Behind the Scenes – APEX Master-Detail Forms.....	138
More on APEX Forms.....	139
Item Layout.....	140
Master-Detail Cleanup.....	147
APEX Help.....	149
Adding a Help Text Region.....	149
Seeding Help Text.....	150
Declarative Blobs.....	151
Summary.....	154
■ Chapter 7: Forms and Reports – Advanced.....	155
Tabular Forms.....	155
Creating a Tabular Form.....	155
Modifying a Tabular Form.....	158
Looking Behind the Scenes.....	161
Interactive Reports.....	162
Creating an Interactive Report.....	163
Running an Interactive Report.....	165
Restricting Functionality by Report.....	167
Restricting Functionality by Column.....	168
Using the Column Heading Menu.....	168
Searching by Column.....	169
Selecting Columns.....	172

Filtering.....	172
Sorting.....	174
Adding Breaks.....	175
Highlighting.....	176
Computing Columns.....	176
Adding Aggregates.....	177
Adding Charts.....	178
Grouping.....	179
Using Flashback.....	180
Saving an Interactive Report.....	180
Resetting an Interactive Report	181
Getting Help	181
Adding a Subscription.....	182
Downloading	183
Modifying an Interactive Report.....	185
Looking Behind the Scenes.....	191
Calendars	192
Understanding Calendar Types	192
Creating a Calendar	192
Looking Behind the Scenes.....	197
Charts.....	198
Writing Queries for Charts.....	199
Creating a Chart	199
Looking Behind the Scenes.....	204
Summary.....	206
■ Chapter 8: Programmatic Elements	207
Conditions	207
Validations	207
Required Values	208
Item Level Validation	209
Page Level Validation	214
Tabular Form Validation	217

Computations	220
Execution	220
Types.....	221
Creating a Computation	221
Processes.....	225
Execution Points.....	225
Process Types	226
Process in the Help Desk Application	226
PL/SQL Regions.....	230
Dynamic SQL.....	232
Dynamic Actions	238
Major Parts.....	238
Dynamic Actions in the Help Desk Application	238
Summary.....	243
Chapter 9: Security	245
User Maintenance Navigation	245
User Maintenance Data Entry	249
Authentication	254
Custom Authentication Schemes	255
Conditional Security	259
Access Control and Authorization	260
Read-Only Items.....	265
Data Security.....	267
Session State Protection.....	270
Summary.....	271
Chapter 10: Application Deployment.....	273
Exporting	273
Importing.....	275
Supporting Objects.....	277

Summary.....	279
Chapter 11: Understanding Websheets	281
Worksheet Structure	281
Navigation	283
Content Navigation.....	283
Structural Navigation	285
Help	285
Markup Syntax	287
User Authentication.....	288
User Authorization.....	290
Sections	295
Text Sections	295
Navigation Sections	298
Data Sections.....	299
Chart Sections.....	311
Annotations	311
Administration.....	312
Summary.....	312
Chapter 12: A Worksheet Example	313
Setup.....	313
Creating and Configuring a Worksheet Application.....	314
Adding Content to a Worksheet.....	317
Creating Data Grids	318
Adding Players	322
Creating Alternate Default Reports	323
Creating Page Sections.....	324
SQL Tags.....	330
Access Controls	331
Summary.....	331

Chapter 13: Extended Developer Tools.....	333
Page Locks.....	333
APEX Conflicts.....	334
Locking an APEX Page	334
Unlocking a Page	335
Administering Page Locks.....	336
Application and Page Groups	336
Application Groups	337
Page Groups.....	338
APEX Views and the APEX Dictionary.....	338
The APEX Schema	339
APEX Views	339
APEX Dictionary.....	341
Searching within APEX.....	342
APEX Finder.....	342
Search Application	343
Monitoring your APEX Application.....	344
Enabling Logging	344
Using the Activity Logs.....	345
Login Attempts	346
APEX Advisor	347
Build Options	348
Understanding the Need	348
Creating a Build Option	349
Configuring Build Options	350
Prompting for Build Option Status	350
Applying Build Options	351
Reporting on Build Option Utilization	352
Page Specific Utilities	352
APEX and Oracle SQL Developer	353
Integration.....	353
Refactoring Support	354

Summary	354
Chapter 14: Managing Workspaces	355
Learning About Your Environment	355
Viewing Instance Information	356
Checking the APEX Version	358
Managing the Service	358
Workspace Preferences	359
Announcements	360
Managing Meta Data	361
Developer Activity and Click Count Logs	361
Session State	362
Application Cache	363
Worksheet Database Objects	363
Application Build Status	364
Application Models	364
File Utilization	366
Interactive Report Settings	367
Managing Users and Groups	369
Creating One User	369
Creating Multiple Users	370
Organizing Users Into Groups	372
Viewing Usage Reports and Dashboards	373
Summary	374
Chapter 15: Team Development	375
Team Development Overview	375
Team Development Interface	377
APEX Home Page	377
Team Development Home Page	378
Common Design Elements	379
Drilldown Functionality	380
Tagging	382

Features	383
Features Tab	383
History Tab	385
Progress Log Tab	386
Milestones	386
Milestones Tab	387
Milestone Details Tab	388
Features by Milestone Tab	388
To Do Items	389
Bugs	390
Feedback	391
Configuring Feedback	392
Polishing the Feedback Page	394
Viewing Feedback	397
Responses to Feedback	398
Communication Between Workspaces	398
Team Actions	399
Manage Links	399
Manage News	400
Team Development Settings	401
Release Summary	401
Utilities	402
User Roles for Team Development	403
Summary	403
Index:	405

About the Authors



■ **Doug Gault** is a Director and Co-Founder at Sumneva™, a world-class Oracle Application Express (APEX) consulting, training, and solutions firm founded in 2010. He has been working with Oracle since 1988, starting with version 5.1B, SQL*Forms 2.0, and RPT/RPF. He has focused his career on Oracle's development technologies, spending the last decade on web-based technologies and the last five years specifically on APEX.

Prior to co-founding Sumneva, Gault was Vice President of Sumner Technologies, an Oracle APEX consulting, training, and solutions firm. Prior to that, he served as the Product Development Director for Hotsos Enterprises, during which time he was the Lead Architect/Developer and Product Manager for two commercial products written exclusively in APEX. His 21 years of Oracle experience have taken him all over the world to participate in some truly ground-breaking projects.

Gault has presented and participated in roundtable discussions at a number of conferences including Oracle OpenWorld, UKOUG, and ODTUG's APEXposed. He holds an Associate's Degree in Computer Science and an honorary Master's Degree from The School of Hard Knocks, believing there is no replacement for hard-earned experience.



■ **Karen Cannell** is president of TH Technology, a small consulting firm providing Oracle technology services, focusing on Application Express. A Mechanical Engineer by degree (one of them), she has analyzed, designed, developed, converted, upgraded, enhanced and otherwise worked on legacy and commercial database applications for over 25 years, concentrating on Oracle technologies since 1994. She has worked with Application Express since it's Web DB and HTMLDB beginnings, and continues to leverage the Oracle suite of tools to build quality web applications for clients in government, medical, and engineering industries. Karen can be contacted at kcannell@thtechnology.com



■ **Patrick Cimolini, P.Eng., PMP**, is a principal with Patrick International APEX Consulting, a consultancy that specializes in Project Management and Software Development services for Oracle Application Express (APEX) projects. His formal training in engineering, business administration, and project management is complemented by thirty years of experience that has evolved through mainframe, client/server, and web platforms. Patrick has enjoyed attending the Oracle Development Tools User Group (ODTUG) conferences where he has been a regular presenter. His industrial areas of expertise include mining, consulting, and government service. His Oracle experience dates back to the early 1990s.



■ **Martin Giffy D'Souza** is a Co-Founder and Chief Technical Officer at ClariFit Inc., a consulting firm and custom solutions provider that specializes in APEX and PL/SQL development. Martin's experience in the technology industry has been focused on developing database-centric web applications using the Oracle APEX technology stack.

Prior to co-founding ClariFit Inc., Martin held a range of positions within award winning companies. Martin is also the author of the highly recognized blog, www.TalkApex.com, which boasts a multitude of posts on a wide array of APEX-focused topics. He has also presented at numerous international conferences such as ODTUG, APEXposed, and COUG.

Martin holds a Computer Engineering degree from Queen's University in Kingston, Ontario, Canada.



■ **Tim St. Hilaire** has been working with Oracle as a developer, integrator, and architect for over ten years. His most recent focus is on the APEX tool set and integration opportunities available both internally and externally. Leading a team of developers, APEX has been integrated as a strategic tool within his current business. Sharing those experiences with others is both a hobby and a passion.

Acknowledgments

First, my heart-felt thanks to all of my co-authors. If not for them, this book may never have come to be. The opportunity to work with such a talented and distinguished group of individuals has been a pleasure.

I'd also like to thank a few people by name. Kerry Osborne and Gary Goodman for providing me with an immense amount of mentorship and encouragement over the years, even after I worked for them. Cary Millsap for helping to solidify in my mind how to think objectively about technology and use proof to find the truth. And last but not least, my business partner Scott Spendolini for his all-around support before, during, and after the book. Without these people, I wouldn't be where I am today.

—Doug Gault

Many thanks to Doug Gault and Apress for including me in this project. It has been an uplifting and enriching experience. A big thank you goes to April Bending, my wife, who supported and encouraged me to take on the challenge.

—Patrick Cimolini

I would like to thank the other authors of this book: Doug Gault, Karen Cannell, Patrick Cimolini, and Tim St. Hilaire. It has been a great journey writing this book with you and I truly feel that we have achieved our goal as a team.

I would like to thank the entire APEX community for all the help and guidance over the past several years. I've learned a lot from all the excellent articles, blogs, and forum posts. I hope that this book will help other developers who are starting out with APEX.

I'd also like to thank Cameron Mahbubian and Chris Hritzuk for their never-ending support over the years.

Finally I'd like to thank my family for their constant encouragement: my parents for providing me with opportunities to succeed, without which I would not be where I am today, and my partner Stephanie for being understanding and putting up with my many late nights of sitting in front of the computer.

—Martin Giffy D'Souza

I would like to thank my wife for her support and understanding during this project. I have a newfound respect for her work as a writer and the challenges posed by the elusive word.

—Tim St. Hilaire