Professional Expertise Distilled

Force.com Tips and Tricks

A quick reference guide for administrators and developers to get more productive with Force.com

Ankit Arora Abhinav Gupta

[PACKT] PUBLISHING enterprise professional expertise distilled

Force.com Tips and Tricks

A quick reference guide for administrators and developers to get more productive with Force.com

Ankit Arora

Abhinav Gupta

[PACKT] PUBLISHING enterprise

professional expertise distilled

BIRMINGHAM - MUMBAI

Force.com Tips and Tricks

First published: February 2013

Production Reference: 1230113

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84968-474-3

www.packtpub.com

Cover Image by Artie Ng (artherng@yahoo.com.au)

Credits

Authors
Ankit Arora
Abhinav Gupta

Reviewers
Naveen Gabrani
Srikanth Goati
Aruna A. Lambat
Caleb Poitevien
Karanraj Sankaranarayanan
Dianne Siebold

Acquisition Editor
Rukhsana Khambatta

Lead Technical Editor
Dayan Hyames

Technical Editor
Prasad Dalvi

Project Coordinator
Leena Purkait

Proofreaders
Aaron Nash
Maria Gould

Indexer
Monica Ajmera Mehta

Graphics
Valentina D'silva

Production Coordinators
Melwyn D'sa
Nilesh R. Mohite

Cover Work
Melwyn D'sa
Nilesh R. Mohite

About the Authors

Ankit Arora (`@forceguru`) is an avid Force.com-certified professional who has been working on the platform since 2008. Since then, he has been involved in architecting, building, and implementing Force.com solutions for on-premise and AppExchange applications. He has also won many online challenges rolled out by Salesforce such as TwitterTrivia, Hammer of Thor, and CloudTrivia.

Ankit is a **Force.com MVP (Most Valuable Professional**) and leader of Jaipur Salesforce Platform Developer Users Group. He has been contributing to the Salesforce community in various ways and through various channels. He is passionate about Force.com and exhibits this by actively blogging at `forceguru.blogspot.in`. He is acting as moderator on the Salesforce Discussion Boards and shares his knowledge and experience by providing effective and converging solutions to developer queries. He has submitted many cookbook recipes that can be found in the online *Force.com Cookbook*.

Ankit resides in Jaipur, also known as the pink city, located in Rajasthan, India, a city that has been able to maintain its rich heritage from the times of Maharajas, yet picking up the pace to emerge as a strong contender for one of the fastest growing cities in India. Ankit lives with his family and likes to play first person combat games such as *Counter Strike* in his free time. He is an enthusiast sportsman and a national level player in the online *Counter Strike* competition.

Though only my name appears on the cover of this book, a great many people have contributed to its production. In particular, I would like to thank my brother-in-law Ajay Deewan, for being my greatest teacher, mentor, and for helping me throughout the book. His patience and support helped me overcome many crisis situations and finish this book.

Finally I would like to thank my family and friends for supporting me.

Abhinav Gupta is a solution architect and an expert cloud computing consultant. He is a Force.com contributor and an avid blogger. He actively participates on Force.com discussion boards, blogs about cloud computing, Salesforce.com, and open source technologies at `http://www.tgerm.com`, and also contributes to various open source projects.

Abhinav has specialized in both native Force.com app development and B2B/B2C integrations with other platforms/APIs. His area of expertise is not only limited to Force.com; he has also done quality work on other cloud platforms such as developing JEE apps on Heroku, Amazon Web Services (EC2, BeanStalk, and so on), and Google App Engine. He is a **Force.com MVP** with notable achievements that include creating the Code Share project Tolerado, winning third place in the 2010 Salesforce Developer Challenge with his mobile location sharing application, and frequently being mentioned in the Salesforce.com newsletter and blog.

Abhinav lives in India with his wife and three-year-old daughter, and enjoys reading technology books and magazines and playing computer games.

Thanks to my wife for being supportive and compromising her time and weekends to spare me for the book, my little daughter for all the fun and naughtiness she spreads around, my mom and dad for all the affection and care, and last but not least the Packt Publishing team and editors for their patience and cooperation throughout the book writing process.

About the Reviewers

Naveen Gabrani is a Force.com architect and is founder of the Salesforce consulting company Astrea IT Services. Astrea is a leader in providing Salesforce.com services. Astrea has three products on AppExchange, Smart vCard, Astrea Clone, and Object Hierarchy, which were envisioned by Naveen. Naveen has 20 years of experience in the IT industry in various technical and management positions.

Srikanth Goati is a Salesforce-certified professional and co-founder of Salesforce Hyderabad User Group. Currently, he is working as a Salesforce administrator with Y-Axis Solutions Pvt Ltd, Hyderbad, AP, India.

Srikanth is an MCA Graduate from Hyderabad. He has acquired the Salesforce Certifications DEV401 and ADM201. He is one of the top three bloggers in the Salesforce community. He has conducted Salesforce training sessions for more than 350 users in his company.

Overall, he has more than 2 years of experience in Salesforce.com development and administrating. He has experience in the Sales cloud, Service cloud, Apex, Visualforce, Database.com, Site.com, and Customer portal.

Internally in his company, he has written more than 50 training manuals on different modules of Salesforce.com.

I wish to thank my parents Prakash and Nagamani and all family members, friends, colleagues, for all the joy they bring into my life.

Thanks to my Salesforce community friends.

Thanks to the folks at Packt Publishing, the authors of this book, and the many others who have provided help and inspiration along the way.

Aruna A. Lambat is an enthusiastic Technical Leader working on Salesforce.com technology with a profound understanding of software design and development. She is passionate about building better products and providing excellent services leading to healthier customer satisfaction.

She has been working on the Salesforce.com platform since 2008. She entered into IT in 2004 as a student. She completed her Master's Degree in Computer Applications from the state Maharashtra, India. She has been working in the IT industry since 2007. She started her carrier as a Java developer and later shifted her focus to cloud computing, specifically in Salesforce.com. She is a Sun-certificated Java developer, web component developer, and Salesforce-certified developer. She is a regular contributor to the Salesforce developer community. She helped the author to cite the example in the book, *Force.com Developer Certification Handbook (DEV401)*.

Aruna works for HCL Technologies; it is primarily engaged in providing a range of outsourcing services, business process outsourcing, and infrastructure services. Aruna works as a Lead Consultant on Salesforce.com technology-based customer services.

Aruna resides in Pune, the cultural capital of Maharashtra, also known for its educational facilities and relative prosperity. She is from Nagpur, the orange city. Her parents staying in the heart of the orange city. She completed her education in this city and achieved success at different time points in her career with immense support from her parents Mr and Mrs Anandrao Lambat. Aruna loves travelling for nature visits, reading fiction books, playing pool, and roaming with friends in her free time.

Aruna can be contacted at:

- Gmail: `Aruna.Lambat@gmail.com`
- LinkedIn: `Aruna Lambat`
- Twitter: `@arunalambat`
- Facebook: `/aruna.lambat`

My special thanks to Siddhesh Kabe for his help and providing me the opportunity for a little contribution for his book, which in turn provided me with the opportunity to work for further Salesforce books from Packt Publishing.

Karanraj Sankaranarayanan is a certified Salesforce.com developer and works full time at Tiara Consulting Services (I) Pvt Ltd, Chennai, the Indian operations of Tiara Consulting headquartered in California, USA. Karan holds a Bachelors Degree in Engineering from Anna University with a specialization in Computer Science. He is passionate about the Salesforce platform, an active member/contributor of the Salesforce customer community/developer forum, and writes blogs. He is also the leader of the Chennai Salesforce Developer user group based in Chennai, India. He can be reached via Twitter `(@karanrajs)`.

Table of Contents

Preface

Force.com is an interesting platform, which allows us to do many things by using the declarative or point-and-click model; without writing a single line of code. This book takes you beyond any documentation or course, and promises hands-on expertise.

Force.com Tips and Tricks will quickly groom you for various Force.com platform secrets that can normally be learnt only after years of exposure. This book is your key to the authors' vast experience with the platform.

Force.com Tips and Tricks starts with very basic admin tasks and gradually moves to hardcore coding tips and tricks for the multitenant Force.com platform.

You will learn admin concepts and basics where you will gain tips and tricks for key topics such as schema and accurate reporting for an organization. Troubleshooting a problem and code re-use are two important aspects that help in boosting productivity; a complete chapter is dedicated to these tasks. As the Force.com platform is multitenant in nature, it requires a more mature mindset compared to other programming languages; expert tips on developing this skill are covered in detail.

What this book covers

Chapter 1, *Learning to Fly with Force.com*, covers the basics of cloud computing. This chapter discusses the principles and constructs of Force.com, the benefits and building blocks of Force.com, when to choose this platform, and many more topics.

Chapter 2, *Admin Tools*, delves deeper into the Force.com platform. This chapter will discuss topics such as Data Loader, the Import wizard, AppExchange marketplace, and Integrated Development Environment.

Chapter 3, Making Best Use of Salesforce Objects, discusses different field data types in Salesforce, various considerations for defining relationships between objects, key standard objects (for CRM), and so on.

Chapter 4, Understanding Analytics, explains about the Salesforce analytics. Salesforce.com provides a very comprehensive analytics and reporting system, which can be used to organize, view, and analyze your data so as to provide real-time visibility into the business.

Chapter 5, Setting Up Development Environments, discusses various development and test environments, and their usages in different scenarios. This chapter will provide tips on how to choose an appropriate development environment.

Chapter 6, Tools and Destinations that Every Force.com Developer Should Know, introduces various tools such as Schema Explorers, toolkits, and data migrators, and destinations such as Twitter, Cookbook, and the DeveloperForce wiki.

Chapter 7, Writing Better Apex Code, illustrates some key best practices, tips, and tricks to write better code in Apex and maintain a good relationship with the governor, that is, as a good tenant.

Chapter 8, Writing Better Visualforce Code, covers tips and tricks around key Visualforce areas such as differentiating facts about Visualforce architecture, how to re-use the native look and feel in pages, and limiting the view state.

What you need for this book

Here is the list of software that you may require for implementing the examples discussed in this book:

- Stable version of a good A grade browser (latest version would be best) supported by Salesforce, such as Chrome, Firefox, or Internet Explorer
- Salesforce Developer Edition Org—one you can sign up for an account at `http://www.developerforce.com/events/regular/registration.php?d=70130000000EjHb`
- Eclipse for desktop installation or an in-browser Developer Console should be fine
- Salesforce DataLoader and Microsoft Excel for Data Loader related tasks

Who this book is for

Force.com Tips and Tricks is not a bible or a complete reference for the Force.com platform development. The time-saving tips and tricks make this book handy for novices as well as experienced developers. This is basically for Force.com developers, who want to extend their Force.com applications, using Flex, Apex, and Visualforce.

When to adopt cloud computing

For an organization, shifting the delivery model from traditional on-premise development to the cloud is a great strategic step, and there are some key considerations to it. Services offered by the cloud vendors may not be suitable for a particular enterprise as the size of an organization is one of the major deciding factors. An organization may need a service when it is in the initial stages, but may need to drop it as it grows.

Consider the following points when deciding on adopting the cloud delivery model:

- The cloud is built to scale its services on demand. Assess whether your demand is stable or changes widely. If it's more or less stable, you may not need to go for extensive cloud services, otherwise the cloud is for you.
- Is the usage frequency of cloud services high? If yes, you may not need to opt for the cloud's "pay-as-you-go" model.
- If your application is mission critical, and needs very strict SLAs (service-level agreements) and almost full control over the infrastructure, you may need to reconsider going for the cloud.
- Are you a start up? If yes, you may not need to invest upfront heavily in infrastructure. the cloud's "pay-as-you-go" model fits easily here.
- Does your organization have a preferred technology and development platform? If yes, vendor lock-in may be a potential issue as migrating from one cloud service provider to another would be much more painstaking than doing it in-house with on-premise software.
- How do you want show your expenses in the balance sheet? Cloud computing model related expenses are being treated as operational expenses and not capital expenses!

When to adopt Force.com for your project

If you have decided to go with the cloud computing way, you may want to consider the option of using Force.com for your projects. We have listed some key guidelines based on the features that the platform has to offer, to help you decide whether Force.com is the right choice for you or not.

- Is your application data centric with storage and retrieval of structured data? This is the core capability of the platform, and applications that are focused on structured data are best suited for this platform.
- Is your application going to store and work with high data volume? Do you have any data warehousing requirements or complex analytics? Force.com may not be the right choice in this case as it's a simple transactional database, limited to only a few million records per object/table.
- Does your application involve large binary content files, such as audio/ video, and other heavy marketing material content? The data size is costly on Force.com, so you may either consider another platform or work in conjunction with other cloud services such as Amazon S3 servers.
- Is your application built around designing configurable dynamic page layouts, wizards, reports, dashboards? Force.com is for you then.
- Does your application address a complex business problem involving many workflows and approvals? Do you want non-technical people to manage and maintain applications with the point and click operations? Force.com is the right choice here.
- Do you need fine-grained security and sharing settings on your data? Do you want to provide hierarchical data access to your users based on the organizational roles? Custom solutions take a lot of time to build this capability, which is natively offered by the platform through the point and click operations.
- Will your application talk to other applications? Is there any third-party integration needed? Force.com has native robust and extensive support for web services integration, both inbound and outbound.
- Does your application involve e-mails, discussions, and collaborations with Twitter-like functionalities? Features such as e-mail services and chatter are presented as native offerings, thus making Force.com the right choice.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "Within the `build.xml` file, there are named targets that process a series of commands when you run Ant with a target name."

A block of code is set as follows:

```
IF(
AND(Payment_Due_Date__c < TODAY(),
ISPICKVAL(Payment_Status__c, "UNPAID")),
"PAYMENT OVERDUE",
null )
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
public with sharing class AccountExtension {
  public Account[] init() {
    // Apart from matching criteria, only those accounts visible to
    current user will be returned
    return [Select Id, Name from Account Where Name like '%corp'];
 }
}
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "If your object name is not displayed, click on **Show all objects**."

[Warnings or important notes appear in a box like this.]

[Tips and tricks appear like this.]

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on `www.packtpub.com` or e-mail `suggest@packtpub.com`.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.PacktPub.com`. If you purchased this book elsewhere, you can visit `http://www.PacktPub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Learning to Fly with Force.com

The fact that you are reading this book implies that you have already chosen Force.com as the preferred platform for developing your applications and may also have developed a few. This book is not a complete, detailed reference of the Force.com platform; its purpose is to provide tips and tricks in both configuration and code that will help you ease some complex tasks, discuss approaches to work around governor limits, talk about some hacks, and more. This book assumes that you are familiar with the platform. In this chapter we will focus on the basics of cloud computing and briefly go over the following:

- Principles and constructs of Force.com
- Benefits and building blocks of Force.com
- When to choose this platform
- Which edition is right for you?
- A cursory overview of how to manage your Salesforce.com org

What is cloud computing?

If you have been in the IT industry for some time, you probably know what cloud means. For the rest, it is used as a metaphor for the worldwide network or the Internet. Computing normally indicates the use of computer hardware and software. Combining these two terms, we get a simple definition—use of computer resources over the Internet (as a service). In other words, when the computing is delegated to resources available over the Internet, we get what is called cloud computing. As Wikipedia defines it:

Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility (like the electricity grid) over a network (typically the Internet).

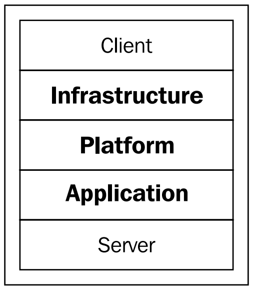
Still confused? A simple example will help clarify it. Say you are managing the IT department of an organization, where you are responsible for purchasing hardware and software (licenses) for your employees and making sure they have the right resources to do their jobs. Whenever there is a new hire, you need to go through all the purchase formalities once again to get your user the necessary resources. Soon this turns out to be a nightmare of managing all your software licenses! Now, what if you could find an alternative where you host an application on the Web, which your users can access through their browsers and interact with it? You are freed from maintaining individual licenses and maintaining high-end hardware at the user machines. Voila, we just discovered cloud computing!

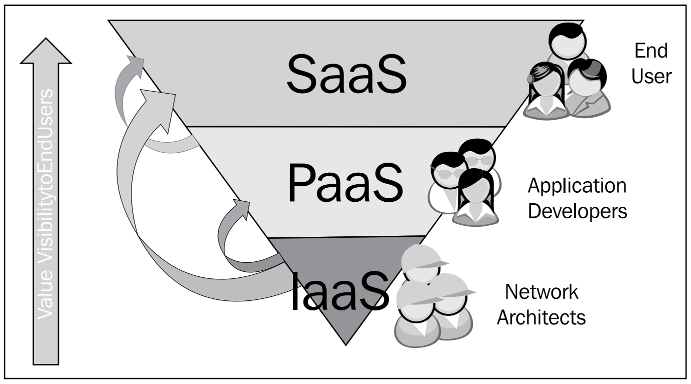
Cloud computing is the logical conclusion drawn from observing the drawbacks of in-house solutions. The trend is now picking up and is quickly replacing the on-premise software application delivery models that are accompanied with high costs of managing data centers, hardware, and software. All users pay for is the quantum of the services that they use. That is why it's sometimes also known as utility-based computing, as the corresponding payment is resource usage based.

Chances are that even before you ever heard of this term, you had been using it unknowingly. Have you ever used hosted e-mail services such as Yahoo, Hotmail, or Gmail where you accessed all of their services through the browser instead of an e-mail client on your computer? Now that is a typical example of cloud computing.

Anything that is offered **as a service** (**aaS**) is usually considered in the realm of cloud computing. Everything in the cloud means no hardware, no software, so no maintenance and that is what the biggest advantage is. Different types of services that are most prominently delivered on the cloud are as follows:

- **Infrastructure as a service** (**IaaS**)
- **Platform as a service** (**PaaS**)
- **Software as a service** (**SaaS**)





Infrastructure as a service (IaaS)

Sometimes referred to hardware as a service, infrastructure as a service offers the IT infrastructure, which includes servers, routers, storages, firewalls, computing resources, and so on, in physical or virtualized forms as a service. Users can subscribe to these services and pay on the basis of need and usage. The key player in this domain is `Amazon.com`, with EC2 and S3 as examples of typical IaaS. **Elastic Cloud Computing** (**EC2**) is a web service that provides resizable computing capacity in the cloud. Computing resources can be scaled up or down within minutes, allowing users to pay for the actual capacity being used. Similarly, S3 is an online storage web service offered by Amazon, which provides 99.999999999 percent durability and 99.99 percent availability of objects over a given year and stores arbitrary objects (computer files) up to 5 terabytes in size!

Platform as a service (PaaS)

PaaS provides the infrastructure for development of software applications. Accessed over the cloud, it sits between IaaS and SaaS where it hides the complexities of dealing with underlying hardware and software. It is an application-centric approach that allows developers to focus more on business applications rather than infrastructure-level issues. Developers no longer have to worry about the server upgrades, scalability, load balancing, service availability, and other infrastructure hassles, as these are delegated to the platform vendors. Paas allows development of custom applications by providing the appropriate building blocks and the necessary infrastructure available as a service.

An excellent example, in this category, is the Force.com platform, which is a game changer in the aaS, specially in the PaaS domain. It exposes a proprietary application development platform, which is woven around a relational database. It stands at a higher level than another key player in this domain, Google App Engine, which supports scalable web application development in Java and Python on the appropriate application server stack, but does not provide equivalent robust proprietary components or the building blocks as Force.com.

Another popular choice (or perhaps not) is Microsoft's application platform called Widows Azure, which can be used to build websites (developed in ASP.NET, PHP, Node.JS), provision virtual machines, and provide cloud services (containers of hosted applications).

A limitation with applications built on these platforms is the quota limits, or the strategy to prohibit the monopolization of the shared resources in the multitenant environment. Some developers see this as a restriction, which allows them to build applications with limited capability, but we reckon this as an opportunity to build highly efficient solutions to work within governor limits, while still maintaining the business process sanctity.

Specificcally for the Force.com platform, some people consider shortage of skilled resources as a possible limitation, but we think the learning curve is steep on this platform and an experienced resource can pick proprietary languages pretty quickly, average ramp up time spanning anywhere from 15 to 30 days!

Software as a service (SaaS)

The opposite end of IaaS is SaaS. Business applications are offered as services over the Internet to users who don't have to go through the complex custom application development and implementation cycles. They also don't invest upfront on the IT infrastructure or maintain their software with regular upgrades. All this is taken care of by the SaaS vendors. These business applications normally provide the customization capability to accommodate specific business needs such as user interfaces, business workflows, and so on. Some good examples in this category are the Salesforce.com CRM system and Google Apps services.

What is Force.com?

Force.com is a natural progression from Salesforce.com, which was started as a sales force automation system offered as a service (SaaS). The need to go beyond the initially offered customizable CRM application and develop custom-based solutions, resulted in a radical shift of cloud delivery model from SaaS to PaaS. The technology that powers Salesforce CRM, whose design fulfills all the prerequisites of being a cloud application, is now available for developing enterprise-level applications.

An independent study of the Force.com platform concluded that compared to the traditional Java-based application development platform, development with the Force.com platform is almost five times faster, with about a 40 percent smaller overall project cost and better quality due to rapid prototyping during the requirement gathering—thanks to the declarative aspect of the Force.com development—and less testing due to proven code re-use.

What empowers Force.com?

Why is Force.com application development so successful? Primarily because of its key architectural features, discussed in the following sections.

Multitenancy

Multitenancy is a concept that is the opposite of single-tenancy. In the Cloud Computing jargon, a customer or an organization is referred to as tenant. The various downsides and cost inefficiencies of single-tenant models are overcame by the multitenant model. A multitenant application caters to multiple organizations, each working in its own isolated virtual environment called org and sharing a single physical instance and version of the application hosted on the Force.com infrastructure. It is isolated because although the infrastructure is shared, every customer's data, customizations, and code remain secure and insulated from other customers.

Multitenant applications run on a single physical instance and version of the application, providing the same robust infrastructure to all their customers. This also means freedom from upfront costs, ongoing upgrades, and maintenance costs. The test methods written by the customers on respective orgs ensure more than 75 percent code coverage and thus help Salesforce.com in regression testing of the Force.com upgrades, releases, and patches. The same is difficult to even visualize with an in-house software application development.

Metadata

What drives the multitenant applications on Force.com? Nothing else but the metadata-driven architecture of the platform! Think about the following:

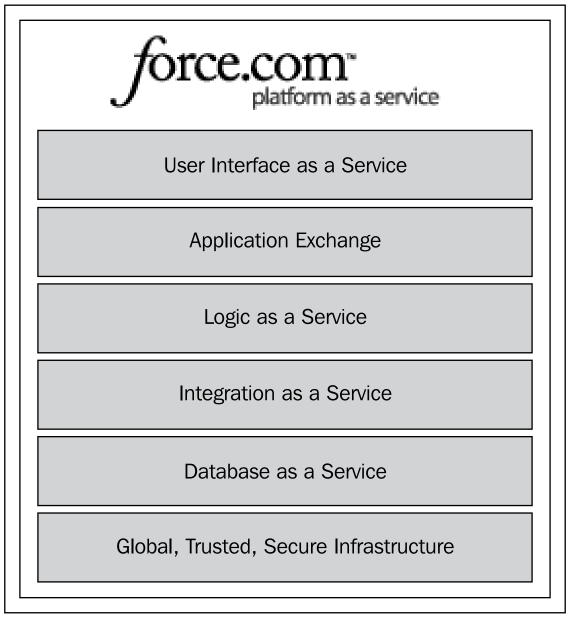
- The platform allows all tenants to coexist at the same time
- Tenants can extend the standard common object model without affecting others
- Tenants' data is kept isolated from others in a shared database
- The platform customizes the interface and business logic without disrupting the services for others
- The platform's codebase can be upgraded to offer new features without affecting the tenants' customizations
- The platform scales up with rising demands and new customers

To meet all the listed challenges, Force.com has been built upon a metadata-driven architecture, where the runtime engine generates application components from the metadata. All customizations to the standard platform for each tenant are stored in the form of metadata, thus keeping the core Force.com application and the client customizations distinctly separate, making it possible to upgrade the core without affecting the metadata. The core Force.com application comprises the application data and the metadata describing the base application, thus forming three layers sitting on top of each other in a common database, with the runtime engine interpreting all these and rendering the final output in the client browser.

As metadata is a virtual representation of the application components and customizations of the standard platform, the statically compiled Force.com application's runtime engine is highly optimized for dynamic metadata access and advanced caching techniques to produce remarkable application response times.

Understanding the Force.com stack

A white paper giving an excellent explanation of the Force.com stack has been published. It describes various layers of technologies and services that make up the platform. We will also cover it here briefly. The application stack is shown in the following diagram:



Infrastructure as a service

Infrastructure is the first layer of the stack on top of which other services function. It acts as the foundation for securely and reliably delivering the cloud applications developed by the customers as well as the core Salesforce CRM applications. It powers more than 200 million transactions per day and more than 1.5 million subscribers. The highly managed data centers provide unparalleled redundancy with near-real-time replication, world class security at physical, network, host, data transmission, and database levels, and excellent design to scale both vertically and horizontally.

Database as a service

The powerful and reliable data persistence layer in the Force.com stack is known as the Force.com database. It sits on top of the infrastructure and provides the majority of the Force.com platform capabilities. The declarative web interface allows user to create objects and fields generating the native application UI around them. Users can also define relationships between objects, create validation rules to ensure data integrity, track history on certain fields, create formula fields to logically derive new data values, create fine-grained security access with the point and click operations, and all of this without writing a single line of code or even worrying about the database backup, tuning, upgrade, and scalability issues!

As compared with the relational database, it is similar in the sense that the object (a data instance) and fields are analogous to tables and columns, and Force.com relationships are similar to the referential integrity constraints in a relation DB. But unlike physically separate tables with dedicated storage, Force.com objects are maintained as a set of metadata interpreted on the fly by the runtime engine and all of the application data is stored in a set of a few large database tables. This data is represented as virtual records based on the interpretation of tenants' customizations stored as metadata.

Integration as a service

Integration as a service utilizes the underlying Force.com database layer and provides the platform's integration capabilities through the open-standards-based web services API. In today's world, most organizations have their applications developed on disparate platforms, which have to work in conjunction to correctly represent and support their internal business processes. Customers' existing applications can connect with Force.com through the SOAP or REST web services to access data and create mashups to combine data from multiple sources. The Force.com platform also allows native applications to integrate with third-party web services through callouts to include information from external systems in organizations' business processes.

These integration capabilities of the platform through API (for example, Bulk API, Chatter API, Metadata API, Apex REST API, Apex SOAP API, Streaming API, and so on) can be used by developers to build custom integration solutions to both produce and consume web services. Accordingly, it's been leveraged by many third parties such as Informatica, Cast Iron, Talend, and so on, to create prepackaged connectors for applications and systems such as Outlook, Lotus Notes, SAP, Oracle Financials, and so on. It also allows clouds such as Facebook, Google, and Amazon to talk to each other and build useful mashups.

The integration ability is the key for developing mobile applications for various device platforms, which solely rely on the web services exposed by the Force.com platform.

Logic as a service

A development platform has to have the capability to create business processes involving complex logic. The Force.com platform oversimplifies this task to automate a company's business processes and requirements. The platform logic features can be utilized by both developers and business analysts to build smart database applications that help increase user productivity, improve data quality, automate manual processes, and adapt quickly to changing requirements.

The platform allows creating the business logic either through a declarative interface in the form of workflow rules, approval processes, required and unique fields, formula fields, validation rules, or in an advanced form by writing triggers and classes in the platform's programming language—Apex—to achieve greater levels of flexibility, which help define any kind of functionality and business requirement that otherwise may not be possible through the point and click operations.

User interface as a service

The user interface of platform applications can be created and customized by either of the two approaches. The Force.com builder application, an interface based on point-and-click/drag-and-drop, allows users to build page layouts that are interpreted from the data model and validation rules with user defined customizations, define custom application components, create application navigation structures through tabs, and define customizable reports and user-specific views.

For more complex pages and tighter control over the presentation layer, a platform allows users to build custom user interfaces through a technology called **Visualforce** (**VF**), which is based on the XML markup tags. The custom VF pages may or may not adopt the standard look and feel based on the stylesheet applied and present data returned from the controller or the logic layer in the structured format.

The Visualforce interfaces are either public, private, or a mix of the two. Private interfaces require users to log in to the system before they can access resources, whereas public interfaces, called sites, can be made available on the Internet to anonymous users.

Development as a service

This a set of features that allow developers to utilize traditional practices for building cloud applications. These features include the following:

- **Force.com Metadata API**: Lets developers push changes directly into the XML files describing the organization's customizations and acts as an alternative to platform's interface to manage applications
- **IDE (Integrated Development Environment)**: A powerful client application built on the Eclipse platform, allowing programmers to code, compile, test, package, and deploy applications
- **A development sandbox**: A separate application environment for development, quality assurance, and training of programmers
- **Code Share**: A service for users around the globe to collaborate on development, testing, and deployment of the cloud applications

Force.com also allows online browser based development providing code assist functionality, repository search, debugging, and so on, thus eliminating the need of a local machine specific IDE.

DaaS expands the Cloud Computing development process to include external tools such as integrated development environments, source control systems, and batch scripts to facilitate developments and deployments.

Force.com AppExchange

This is a cloud marketplace (accessible at `http://appexchange.salesforce.com/`) that helps commercial application vendors to publish their custom development applications as packages and then reach out to potential customers who can install them on their orgs with merely a button click through the web interface, without going through the hassles of software installation and configuration. Here, you may find good apps that provide functionality, that are not available in Salesforce, or which may require some heavy duty custom development if carried out on-premises!

Introduction to governor limits

Any introduction to Force.com is incomplete without a mention of governor limits. By nature, all multitenant architecture based applications such as Force.com have to have a mechanism that does not allow the code to abuse the shared resources so that other tenants in the infrastructure remain unaffected. In the Force.com world, it is the Apex runtime engine that takes care of such malicious code by enforcing runtime limits (called governor limits) in almost all areas of programming on the Force.com platform.

If these governor limits had not been in place, even the simplest code, such as an endless loop, would consume enough resources to disrupt the service to the other users of the system, as they all share the same physical infrastructure. The concept of governor limits is not just limited to Force.com, but extends to all SaaS/PaaS applications, such as Google App Engine, and is critical for making the cloud-based development platform stable.

This concept may prove to be very painful for some people, but there is a key logic to it. The platform enforces the best practices so that the application is practically usable and makes an optimal usage of resources, keeping the code well under governor limits. So the longer you work on Force.com, the more you become familiar with these limits, the more stable your code becomes over time, and the easier it becomes to work around these limits.

In one of the forthcoming chapters, we will discover how to work with these governor limits and not against them, and also talk about ways to work around them, if required.

Salesforce environments

An environment is a set of resources, physical or logical, that let users build, test, deploy, and use applications. In the traditional development model, one would expect to have application servers, web servers, databases, and their costly provisioning and configuration. But in the Force.com paradigm, all that's needed is a computer and an Internet connection to immediately get started to build and test a SaaS application.

An environment, or a virtual or logical instance of the Force.com infrastructure and platform, is also called an **organization** or just **org**, which is provisioned in the cloud on demand. It has the following characteristics:

- Used for development, testing, and/or production
- Contains data and customizations
- Based on the edition containing specific functionality, objects, storage, and limits
- Certain restricted functionalities, such as the multicurrency feature (which is not available by default), can be enabled on demand
- All environments are accessible through a web browser

There are broadly three types of environments available for developing, testing, and deploying applications:

- **Production environments**: The Salesforce.com environments that have active paying users accessing the business critical data.
- **Development environments**: These environments are used strictly for the development and testing applications with data that is not business critical, without affecting production environment. Developer environments are of two types:
    - **Developer Edition**: This is a free, full-featured copy of the Enterprise Edition, with less storage and users. It allows users to create packaged applications suitable for any Salesforce production environment. It can be of two types:
        - **Regular Developer Edition**: This is a regular DE org whose sign up is free and the user can register for any number of DE orgs. This is suitable when you want to develop managed packages for distribution through AppExchange or Trialforce, when you are working with an edition where sandbox is not available, or if you just want to explore the Force.com platform for free.
        - **Partner Developer Edition**: This is a regular DE org but with more storage, features, and licenses. This is suitable when you expect a larger team to work who need a bigger environment to test the application against a larger real-life dataset. Note that this org can only be created with the Salesforce Consulting partners or Force.com ISV.
    - **Sandbox**: This is nearly an identical copy of the production environment available to Enterprise or Unlimited Edition customers, and can contain data and/or customizations. This is suitable when developing applications for production environments only with no plans to distribute applications commercially through AppExchange or Trialforce, or if you want to test the beta-managed packages. Note that sandboxes are completely isolated from your Salesforce production organization, so operations you perform in your sandboxes do not affect your Salesforce production organization, and vice versa. Types of sandboxes are as follows:
        - **Full copy sandbox**: Nearly an identical copy of the production environment, including data and customizations
        - **Configuration-only sandbox**: Contains only configurations and not data from the production environment

    - **Developer sandbox**: Same as Configuration-only sandbox but with less storage

- **Test environments**: These can be either production or developer environments, used speficially for testing application functionality before deploying to production or releasing to customers. These environments are suitable when you want to test applications in production such as environments with more users and storage to run real-life tests.

Summary

This chapter talked about the basic concepts of cloud computing. The key takeaway items from this chapter are the explanations of the different types of cloud-based services such as IaaS, SaaS, and PaaS. We introduced the Force.com platform and its key architectural features that power the platform types, such as multitenant and metadata. We briefly covered the application stack—technology and services layers—that makes up the Force.com platform. We gave an overview of governor limits without going too much detail about their use. We discussed situations where adopting cloud computing may be beneficial. We also discussed the guidelines that help you decide whether your software project should be developed on the Force.com platform or not. Last, but not least, we discussed various environments available to developers and business users and their characteristics and usage.

2
Admin Tools

Now that we have set the ground work, we can delve deeper into the Force.com platform. In this chapter, we will focus on simplifying some administrative tools and utilities. We will cover:

- Data Loader
- Import Wizard
- Other third party tools for Data Integration
- Integrated Development Environment
- AppExchange Market Place
- Some good AppExchange listings

Data Loader

As the name itself indicates, this is a Windows-based client application that is used for inserting, updating, upserting, deleting, and extracting records. It can be accessed by navigating to **Your Name | Setup | Administration Setup | Data Management | Data Loader**. This tool can be used to move data into and out of any object via CSV files. The records do not get duplicated and it is ensured by maintaining a unique ID from the external system called the External ID. During data import, the Data Loader identifies the existing records by using this External ID and prevents creating duplicate records by upserting them.

The org-wide data import for any object can only be carried out by administrators, whereas the **Import My Contacts** operation can be performed by any user in the system but only in those fields, which are accessible through field-level security and their page layouts.

In addition to using this tool in the interactive mode through the user interface, it can also be run as a scheduled batch process from the command line (via the command line interpreter) so that manual intervention is not required. In this case, the tool needs configurations, data sources, mappings, and actions in the form of files to avoid manual intervention.

The tool also supports Bulk API to import large datasets of up to 5 million records, and can also be used for importing documents and attachments into Enterprise, Unlimited, and Developer Editions of Salesforce.com.

When to use Data Loader

Salesforce.com has provided certain guidelines for using Data Loader over the web-based import wizard. Use Data Loader when you want to:

- Import data of up to 5 million records
- Load records for objects not supported by the import wizard
- Schedule regular data load
- Export data for backup purposes
- Delete custom object data that can't be accessed through the web interface

Strategy to import data

One of the most important steps in any project implementation is importing legacy/bulk data for testing the application being developed. The data import process can be divided into the steps discussed in the following sections.

Identifying data sources

There's a lot of data around you when you actually start noticing it. You need to analyze what goes into Salesforce. The following are some guidelines:

- This is the time to clean up source data when unused, old, archived, or insignificant data gets thrown away during identification of data sources
- Clearly document the various data sources and their attributes of records, such as type, count, and so on
- Generate the high-level mapping of source data to target schema
- Identify dependency of source data to define the sequence/order of data import; for example, before importing the Opportunities data, it's logical to import the Accounts and Contact data first

Data preparation

There are differing opinions on data preparation. Some prefer to do it before they do the import. Some prefer doing it after the data has been inserted into Salesforce. Irrespective of the approach you decide to go with, here are a few tips that can help you get your data prepared:

- You may export data to any third-party tools that allow you to delete columns, sort rows, and make global changes
- Ensure the data follows any standardized naming conventions, otherwise you may run the risk of data duplication
- Make the Salesforce schema changes at the field level as per the source data to enable data migration
- Make sure to mark appropriate fields as External ID in Salesforce
- Generate field-level mappings
- Make sure the source data conforms to the Salesforce standards in terms of size, type, and so on
- A data source column should be added as a custom field in Salesforce to the data being imported to always identify where it came from
- Assign appropriate owners to the imported data records
- Get stakeholders' acceptance of the mapping and the migration strategy document you prepared
- There may be triggers and/or workflows that may get executed when the data is loaded—analyze whether you need to disable them when the data is loaded

Testing the data import

You may want to test it out first before you go for final data import. Here are a few tips:

- Select a small sample of data, usually containing the most significant records for the stakeholders so that the most valued data gets immediately reviewed and approved
- In addition, try using the page layouts to further validate data import
- Do not try it on the production environment directly; try on a full copy sandbox instead, if available

Analyzing the test import

Once the data has reached Salesforce, make sure you crosscheck it with the test file that was imported to ensure correctness and integrity. Here are a few tips to analyze your test data import results:

- Build a report that lets you look at the record data collectively
- Pick up a few samples of the imported records and compare them with the original data; make sure that the relationship between objects is set up properly and whether there are fields with large text data, and then validate that the data is completely transferred
- Develop a custom view for the relevant tab's homepage with the most important columns that can be viewed by the customers to verify the imported data
- Get your imported data verified by the stakeholders as it's their feedback which is of utmost importance
- Incorporate feedback and modify the data and/or the import process in terms of mapping, schema, and so on, to rectify the data import

Final data import

After the test data import analysis, you are ready to take off. Get ready to commence the final data import. Here are a few tips:

- Plan for the final data import as it cannot be expected to start abruptly
- Set the right expectations for the end users for the down time
- Large volume data might mean running the import process during non-working hours
- Always plan for errors and start the process well before the committed timelines

Validating the import

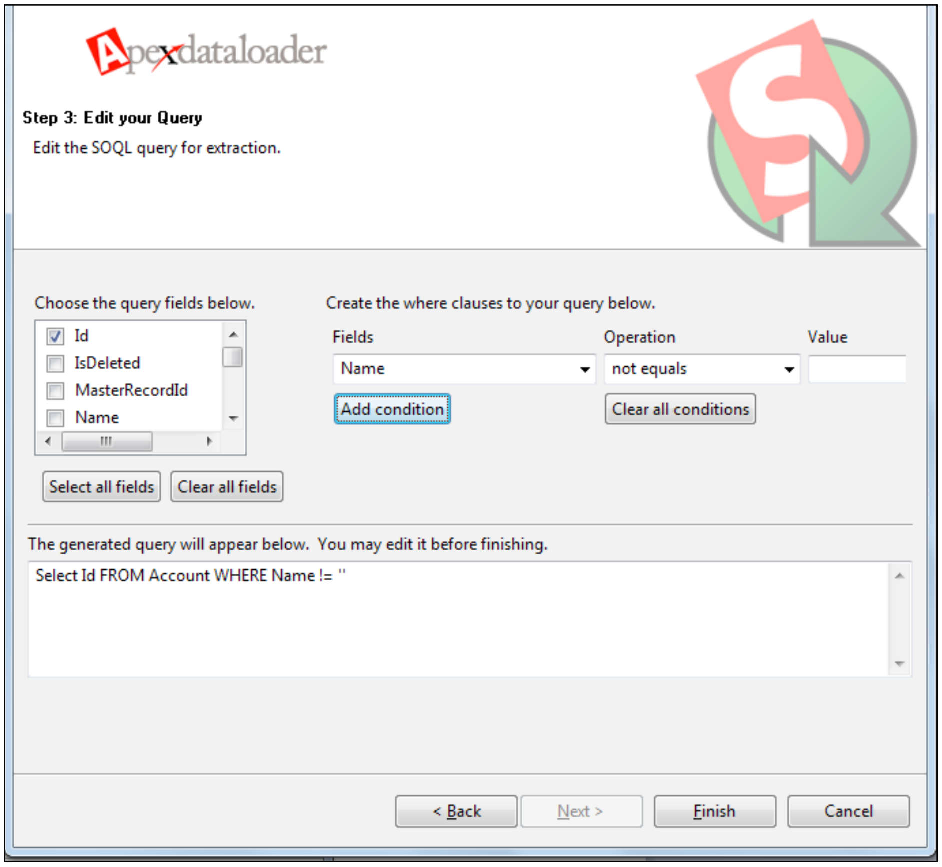
You have already done this, but let's quickly go over it again:

- As before, run the reports to verify correctness and integrity
- Have end users verify the import

Using Data Loader for data export

Data Loader can also be used for exporting your org data. Follow these steps:

1. Click on **Export**.
2. Enter Salesforce username and password, and click on **login**.
3. Select an object whose data is to be extracted; if your object name is not displayed, click on **Show all objects**. Choose the target for extraction. Click on **Next**.
4. Create a SOQL query for data export. Here choose the fields you want to export and add conditions to filter the dataset.
5. Click on **Finish** and a CSV is extracted to the location with all the values of fields selected in the query.


Apexdataloader
Step 3: Edit your Query
Edit the SOQL query for extraction.
Choose the query fields below.
Id
IsDeleted
MasterRecordId
Name
Select all fields
Clear all fields
Create the where clauses to your query below.
Fields
Operation
Value
Add condition
Clear all conditions
The generated query will appear below. You may edit it before finishing.
< Back
Next >
Finish
Cancel

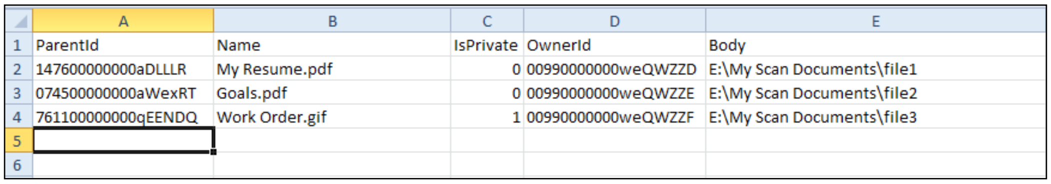
Fields: Name; Operation: not equals

```
Select Id FROM Account WHERE Name != ''
```

Using Data Loader for uploading attachments

Follow these steps for uploading attachments through Data Loader:

1. Log in to Data Loader.
2. Click on **Insert**.
3. Check the **Show all Salesforce objects** checkbox and select the attachment object from the list. Choose the CSV file that contains the following:
    - **ParentId**: The Salesforce ID of the parent record
    - **Name**: The name of the attachment file
    - **IsPrivate**: The value in this field is **0** if the attachment is not private and **1** if it is private
    - **OwnerId**: The Salesforce Id of the record owner
    - **Body**: The Salesforce Id of the attachment
4. Edit the values in the body column so that they contain the full filenames of the attachments as they exist on your computer.



	A	B	C	D	E
1	ParentId	Name	IsPrivate	OwnerId	Body
2	147600000000aDLLLR	My Resume.pdf	0	00990000000weQWZZD	E:\My Scan Documents\file1
3	074500000000aWexRT	Goals.pdf	0	00990000000weQWZZE	E:\My Scan Documents\file2
4	761100000000qEENDQ	Work Order.gif	1	00990000000weQWZZF	E:\My Scan Documents\file3
5					
6					

5. Create mapping for fields in object and on CSV.
6. Select the directory where your success and error fields will be saved and click on **Finish**.

The Import wizard

This is a tool built into the native user interface, allowing bulk data to be imported as new or updated records of custom objects. Import wizards for the leads, accounts, contacts, solutions, and custom objects are located under **Setup** | **Administration Setup** | **Data Management**. The administrator also sees these links in the **Tools** section of certain tabs' homepages. For example, to import leads, click on the **Leads** tab and then click on the **Import Leads** link in the **Tools** section.

- For records you own: The **Import** wizards for the records you personally own are located at:
    - **Your Name | Setup | Import | Import My Contacts**
    - **Your Name | Setup | Import | Import My Person Accounts**
- For your organization's records:
    - **Import My Organization's Accounts and Contacts**
    - **Import My Organization's Person Accounts**
    - **Import My Organization's Leads**
    - **Import My Organization's Solutions**
    - **Import My Organization's Custom Objects**

When to use the Import wizard

Use **Import** wizard over Data Loader when:

- You want to load fewer than 50,000 records
- The object you need to import is supported by the **Import** wizard
- You want to prevent duplicates by uploading records according to account name and site, contact e-mail address, or lead e-mail address
- The Data Loader tool is not installed on your machine and you prefer working over the Web instead of a desktop-based client application

Notes on data import

A few tips taken from Salesforce user guide on the field accessibility and how different field type values are imported:

- **Field accessibility**:
    - In the organization-wide import of accounts and leads, you can import data into any standard or custom field, even if it is hidden or read-only in your page layout or the field-level security settings.
    - For the **Import My Contacts** wizard, you can import data in only those fields that are editable in your page layout or the field-level security settings.

- **New values for picklists**:
    - If you import the data to any picklist field, the wizard warns you when you attempt to load a value that does not match any existing picklist value.
    - If you ignore the warning, the value gets automatically added to the imported record. Your administrator can later add this value to the picklist field.
- **Multiselect picklists**: To import values into this field, separate values by a semicolon in the import file.
- **Checkboxes**: Use `1` for checked and `0` for unchecked values in the import file.
- **Default values**: For the picklist, multiselect picklist, and checkbox fields, the default value gets automatically populated in the new or updated record, if the field is not mapped in the Import wizard.
- **Date/time fields**: Ensure that the format of any date/time field that you are importing matches how they display in Salesforce per your locale setting.
- **Formula fields**: Don't accept any value in the import as they are derived and read-only.
- **Field validation rules**: Salesforce runs validation rules on the records prior to importing them. Records that fail are discarded. You may want to deactivate validation rules before running the Import wizard.

Undoing data import

If you have imported the accounts, contacts, leads, or solutions data by mistake, your administrator can navigate to **Your Name** | **Setup** | **Data Management** | **Mass Delete Records** to delete the items you accidentally or incorrectly imported. The **Mass Delete Records** tools do not support custom objects. For custom objects data, your administrator can use the Data Loader tool to delete the mistakenly imported records.

Third-party tools for data integration

The data import options provided by Salesforce face a limitation that they can only work with the CSV files and have no capability of integrating with any other client's backend databases or ERP applications. They mean an extra overhead whenever it comes to integrating your Salesforce org with real-life customer business processes. For example, if you need to synchronize your Accounts and Contacts data with your client's backend system, be it an on-premise database or ERP application or a SaaS application, the data would need to be constantly exported and imported in the form of CSV files at both ends.

Moreover, there is no capability of data transformations and/or applying complex business logics with the Salesforce wizards and data loader. So an additional middleware is required to perform these transformations. Again, this might mean custom implementations for every specific data integration piece needed between Salesforce and the customer backend.

To overcome these limitations, many reputed ETL or integration experts have developed custom solutions that can exist either on the hosted environment or can run on the SFDC platform as they are written in native Apex/Visualforce. These custom developed solutions allow building very simple to very complex and sophisticated integration processes that span hosted and on-premise applications. They help automate complex business processes such as account/contact synchronization between SFDC and ERP systems, sales transaction history, forecast management, contact integration, product master, pricing master, orders, quotes, cases, price list distribution, and so on. These solutions can help to virtually integrate Salesforce with any customer backend system, such as SAP, Oracle, JD Edwards, Quickbooks, Net Suite, Great Plains, Dynamics, Peachtree, Databases, Web Services, and so on.

A few of the market players that provide integration capabilities are Informatica, Boomi, Cast Iron, and Talend. Except Talend Open Studio community edition, which is a highly feature-rich, free ETL tool available in the market, almost all of the remaining ones are paid. So you may want to explore and evaluate Talend for your data and business process integration requirements.

It is worthwhile to mention here a free and open source data integration tool developed by Ron Hess, which is called Force.com Excel Connector. It is a Microsoft Excel add-in that allows communicating with Force.com API via SOAP and is built in Office Toolkit from Salesforce.com.

- It provides bi-directional access to Salesforce.com from an Excel spreadsheet and you can manage data updates and extractions within the same Excel spreadsheet
- It is also secure as users have to log in with their Salesforce credentials, which means that role and profile permissions are automatically applied and they see only that data which they are authorized to see
- The power of this tool lies in its simplicity and ease of usage as users can manage data without leaving the Excel interface

- You can make use of strong and effective Excel features such as:
    - Formulas to update data
    - Conditional formatting for de-duplication
    - Filters to update specific records
    - Charts for powerful analytics
- In Professional Edition orgs, where Data Loader is not available, the Excel connector is of great help for data import/export

Another powerful data migration tool that is available for free is the Jitterbit Data Loader for Salesforce, which has several powerful features such as automating the import and export of data between flat files, databases, and Salesforce. This tool also works on Group and Professional Editions and is available for Windows and Mac. Please visit the developer's site (`http://www.jitterbit.com/salesforce/data-loader`) for more details.

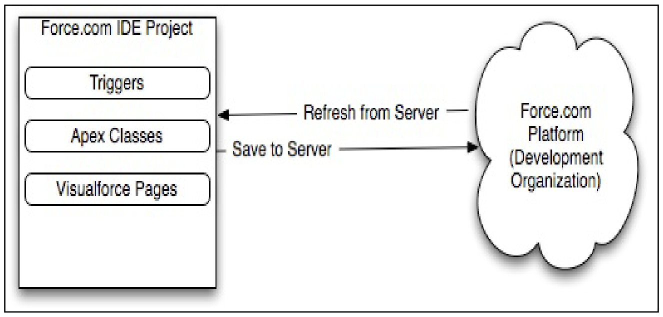
Force.com Integrated Development Environment (IDE)

The Force.com IDE is a free, Salesforce.com-supported Eclipse plugin. It is an extension to the standard Eclipse development tool that helps with developing, modifying, testing, and deploying applications on the Force.com platform. It communicates with Salesforce, using the metadata API, which requires a user's profile to have the **Customize Application** and **Modify All Data** permissions.

The IDE is a plugin for Eclipse, so it requires specific compatible versions of Eclipse. So even if you are already using an older version, which is not supported by the IDE, you can have another newer, compatible version installed on your system as multiple versions of Eclipse can coexist on a system. Instructions for the Force.com IDE installation can be found at `http://wiki.developerforce.com/index.php/Force.com_IDE_Installation`.

IDE communication

The following diagram indicates how the IDE works on a local machine while saving and refreshing the metadata to and from a server:



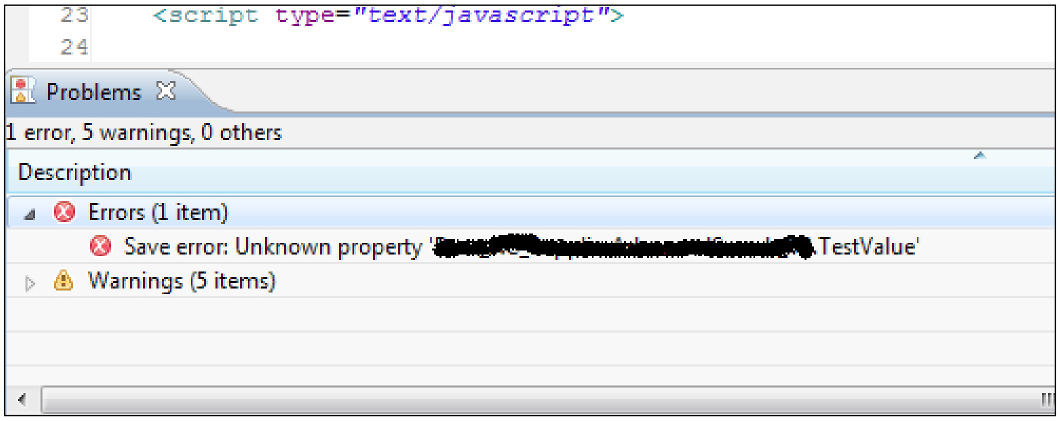
Force.com perspective

When the Eclipse is launched after installing the IDE, the Force.com perspective becomes visible. A **perspective** in the context of Eclipse is a way to organize and view the user interface components associated with a program and it acts like a visual container for a set of views and editors. It presents an environment that is tailored to a specific development task, providing you a perfectly adapted workspace.

The Force.com perspective allows creation of a Force.com project. This project further allows you to create, update, and delete almost all of the metadata-driven components such as Apex and Visualforce code, objects, labels, static resources, and so on. This local editing of metadata is deployed through the metadata Web Services API to the Salesforce server every time a save operation is initiated by the user. What is important here to note is that this API call gets counted in the daily limit of API calls and with many developers working on the same org using the Force.com IDE may mean hitting the API governor limit very soon, thus resulting in its exhaustion and stoppage of work. The work around to this problem is to switch back to the web-based development interface, which could be problematic to a developer who is used to working with the desktop-based IDEs.

Problems view

The Force.com IDE uses the standard Eclipse view called **Problems** to display the compilation errors when the metadata is sent to the Salesforce server for compilation. If it fails, all compilation errors are displayed in this view. In almost all cases, double-clicking on the issue takes you to the file and line of the problematic code.



```
    <script type="text/javascript">
```

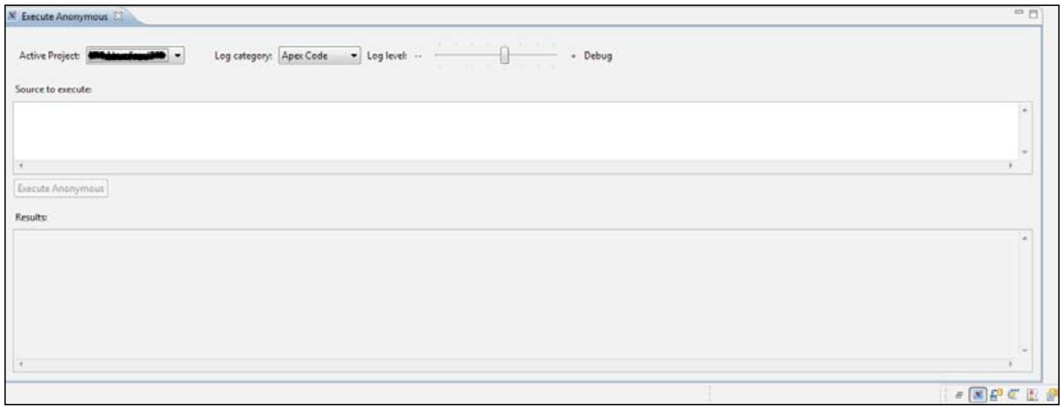
Problems

1 error, 5 warnings, 0 others

Description
Errors (1 item)
Save error: Unknown property '[redacted]TestValue'
Warnings (5 items)

Execute Anonymous view

Another important tool in the IDE is the **Execute Anonymous** view, which allows you to run an anonymous block of Apex on the server. Anonymous blocks quickly help you to evaluate Apex on the fly. This view is usually available on the lower right-hand side of the Force.com perspective. This view provides an equivalent functionality to the system log present in the Salesforce web-based UI.



Execute Anonymous

Active Project: [redacted]   Log category: Apex Code   Log level: Debug

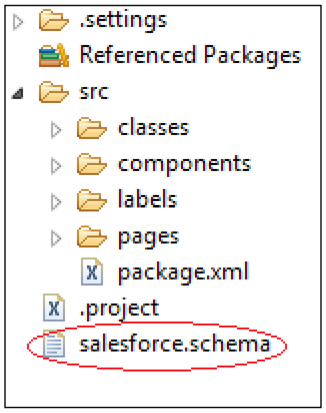
Source to execute:

Apex Test Runner view

Yet another important and powerful view in the Force.com IDE is the **Apex Test Runner** view in which you can run your test methods to see which Apex unit tests are passing or failing, including code performance and test coverage. This view displays a test results summary—which code pieces do and do not meet minimum code coverage requirements for production deployment, including a list of all lines not included. It also displays the output of the Apex debug statements and other system log events captured during test execution. This information is necessary for troubleshooting code, performance tuning, and checking resource usage.

Schema explorer

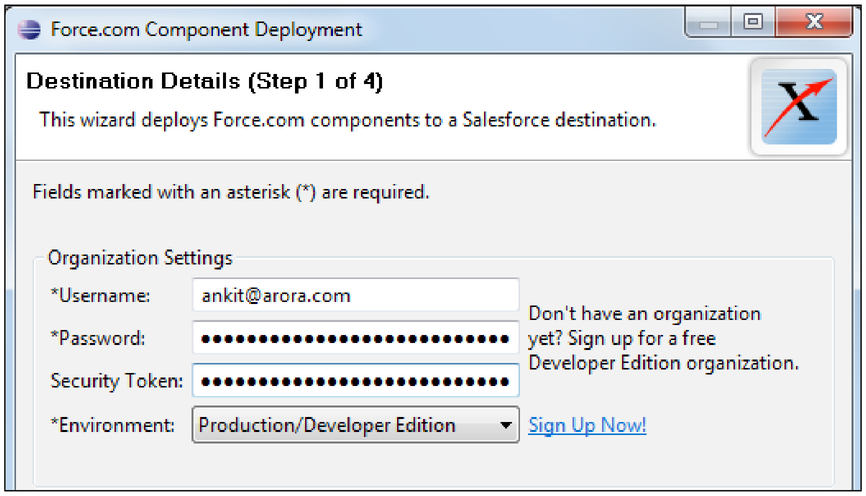
This is a tool for browsing objects and fields that exist in your Salesforce org. It presents the logged-in user's view of the Force.com data model in the hierarchical tree view including object visibility, permissions, data types, and lookup values that are useful in developing the applications on the Force.com platform. The tool can be used to directly interact with the Force.com database by executing queries on your organization's live data and inspecting their results.



Deployment

The IDE allows deployment of the Force.com components to any server for the purposes of testing, staging, publication, or production use by the end users, once the components have been created and unit tested in the development organization. The code can be deployed to a sandbox for testing against the copy of your production data. The IDE allows selective movement of metadata components to the destination org.

After selecting the components to deploy, you can either validate the deployment, which will fully execute the deploy process on the server, without actually committing the changes, or you can execute the deployment so that the metadata component changes are saved. In case of any deployment failures, the list of issues is presented to the user for fixing.



Destination Details (Step 1 of 4)

This wizard deploys Force.com components to a Salesforce destination.

Fields marked with an asterisk (*) are required.

Organization Settings

*Username: ankit@arora.com

*Password: ••••••••••••••••••••••••••••

Security Token: ••••••••••••••••••••••••••••

*Environment: Production/Developer Edition

Don't have an organization yet? Sign up for a free Developer Edition organization.

Sign Up Now!

Force.com Migration Tool

Force.com Migration Tool is a Java- or Ant-based command-line utility for scripting deployments to move metadata between a local directory and a Force.com organization. This tool can be downloaded from your Salesforce org. Navigate to **Your Name** | **Setup** | **Develop** | **Tools** and then click on **Force.com Migration Tool**.

Consider using this tool in the following scenarios:

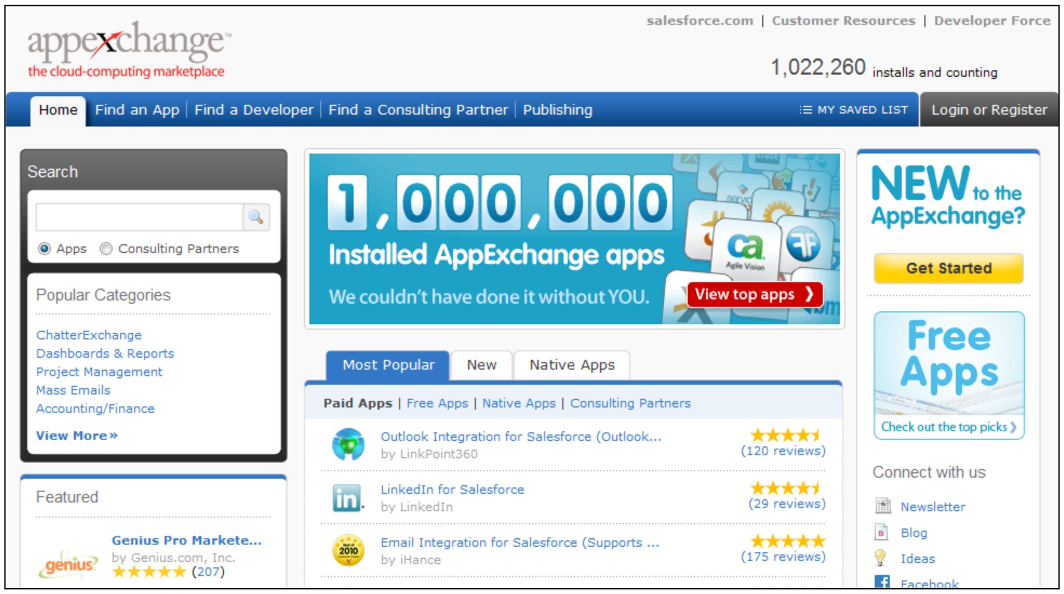
- When you need to populate a test environment with frequent and large volumes of setup changes where the web interface could consume a lot of time
- Where there is a multistage release process, the scripted deployment process is much more time efficient
- When there is a repetitive process for retrieving and deploying a fixed set of metadata components with the same parameters
- When you are already familiar with the scripted deployment process using Apache Ant

The tool uses the configuration file `build.properties` for determining org connections and the `package.xml` file for determining the deployment components. The XML file provides a fine-grained control over what goes into the deployment, where it gets deployed, and what unit tests will be run. The tool also allows destroying the metadata on the destination org through `destructiveChanges.xml` to perform cleanups prior to the deployment. The `build.xml` file provides a series of commands to be executed by Ant. Within the `build.xml` file, there are named targets that process a series of commands when you run Ant with a target name. The sample `build.xml` file contains a number of useful targets for various `retrieve()` and `deploy()` options that you can modify or use as is.

Once the tool is executed, it behaves similarly to the deployment wizard in the sense that it archives the files, sends them to the Force.com services for compilation, and displays the deployment results on the command line.

AppExchange – cloud application marketplace

This is a marketplace for cloud computing applications and services, built for the Salesforce.com community delivered by partners or the third-party vendors. Users can purchase it and add to their Salesforce.com environments.



The AppExchange incorporates the best practices of already-existing marketplaces and makes purchasing of apps a wonderful experience for the potential customers. The applications are accompanied with demos, screenshots, trial versions, specifications, and so on, which can be thoroughly reviewed by customers before they can go ahead and make the decision for the final purchase. When they are satisfied, they can simply go ahead and click on the **Get It Now** button, which guides them through an easy wizard to install the app in their org.

Besides the paid applications, you will notice that a number of free applications exist in the AppExchange and most of them have been published by Force.com Labs and developed internally by the Salesforce employees to enhance the Force.com user experience. These applications range from small, common utilities to large project management applications and chatter plugins. So it is highly recommended that if you plan to build an in-house Salesforce.com application of your own, make sure to search for a similar application on the AppExchange. Chances are that you will find what you are looking for, or at least get a close match for your needs that can be tweaked to suit your requirements.

We have listed some free/paid apps from AppExchange, which we think might be useful for your business as you make progress on the Force.com platform.

DupeCatcher – real-time deduplication

This app is developed by Symphonic Source.

DupeCatcher makes deduplication of Salesforce leads, accounts, and contacts comprehensive and easy. Designed as a complementary utility, DupeCatcher can identify and block duplicate records at the point of entry based on standard and/or custom fields (including Person Accounts).

Milestones PM – project and task management

This app is developed by Force.com Labs.

Milestones PM is a native Force.com app designed to help you track and manage your projects and tasks. Milestones PM has a simple interface, Chatter integration, and detailed reporting capabilities.

Salesforce CRM dashboards

This app is developed by Force.com Labs.

With these CRM dashboards, your sales and service teams can stay on top of every deal or case, and close more business on time than ever before. Sales managers will have more visibility into open opportunities and gain greater pipeline predictability. Dashboards and reports in this package are intended to provide you with a starting point to make reporting easier in your organization.

Salesforce for Twitter and Facebook (v4)

This app is developed by Force.com Labs.

Connect your Salesforce CRM with social channels Twitter and Facebook. Monitor your online footprint, connect with customers, promote your brand, and analyze your social impact—all directly from Salesforce.com!

Appirio Contact Sync for Salesforce and Google Apps

This app is developed by Appirio.

Appirio Contact Sync for Salesforce and Google Apps is a simple tool to select and synchronize contacts between your Google and Salesforce address books (unsupported).

Ribbit for Salesforce

This app is developed by Ribbit.

Ribbit for Salesforce is the only sales productivity tool that unifies mobile voice and SMS communications, Salesforce CRM, e-mail, and voice-to-text transcriptions. Built-in automation helps you work less and sell more.

SnapShot Change And Release Management

This app is developed by DreamFactory Software.

Snapshot is the ultimate tool for change management, change reporting, and compliance documentation for the Salesforce orgs. With SnapShot's rich interactive GUI, admins can compare and differentiate orgs, monitor changes to orgs, and push customizations between orgs.

Salesforce Adoption Dashboards (2011)

This app is developed by Force.com Labs.

Great user adoption doesn't just happen! The Salesforce Adoption Dashboards provide visibility to relevant user login history and trending, adoption of key features such as accounts and opportunities, and critical sales and marketing productivity enhancers.

Survey Force

This app is developed by Force.com Labs.

Survey Force allows you to create, send, and capture customer feedback natively in Salesforce.com. Surveys are deployed via e-mail templates and sites. Survey results are related to the Contact and/or Case. Reports and Dashboards can be leveraged to analyze survey results.

Draggin' Role

This app is developed by Qandor.

Manage your role hierarchy with ease! Now you can drag-and-drop your way through any hierarchy modifications. Draggin' Role is a free application that allows you to view and manipulate users and roles from a single Custom tab.

Find Nearby – Accounts, Contacts, Leads – Managed, PE/EE/UE/DE

This app is developed by Force.com Labs.

With this tool you can:

- Find all your Accounts, Contacts, and Leads in an area
- Map the Custom Search items from list views
- Plan your next Sales Trip and get driving directions to each location

AppExchange Dashboard Pack

This app is developed by Force.com Labs.

It provides Consolidated AppExchange Dashboards for Installation, Sales, Marketing, Support, Adoption, and more.

Action Plans – v3 – Unmanaged – EE, UE, and DE

This app is developed by Force.com Labs.

Action Plans encapsulates best practices into reusable task templates. An action plan can be created for an Account, Opportunity, Contact, or Lead. Template tasks can be pre-assigned to a specific individual or assigned to the running user.

Project and Issue Management

This app is developed by Force.com Labs.

With Project and Issue Management for AppExchange, organizations have a simple framework for prioritizing and managing the logistics of projects and the resources allocated to them.

CloudConverter for Force.com

This app is developed by Model Metrics.

Model Metrics' CloudConverter for Force.com is a cutting edge tool that automates the process of migrating custom applications from a Lotus Notes environment or any other common commercial database to Force.com.

Opportunity Planning Wall

This app is developed by LogicLine.

Got lost in your sales pipeline? Do you need a more visual and interactive tool for sales planning? The Opportunity Planning Wall displays sales opportunities as cards on a virtual wall and lets you filter, group, and modify on one simple screen.

Data Loader – Salesforce integration

This app is developed by Informatica Corp.

Informatica Cloud Data Loader is a free data integration service that enables ad hoc import/export of Salesforce data between databases and files. This app is installed as a tab. The first 30 days provide access to all Informatica Cloud services, including scheduling.

Mass Edit + Mass Update + Mass Delete

This app is developed by VersatileCapitalist, Inc.

This includes the Mass Update + Mass Edit + Mass Delete records from any list view or related list and Mass Edit + Mass Update + Mass Delete Various Entities, for example, Leads, Accounts, Contacts, and Opportunities.

Implementation Cloud – project management app

This app is developed by Implementation Cloud Ltd.

It is a free app to project management and it manages the implementation and ongoing administration of Salesforce. Plan, manage, and deploy your ideal Salesforce setup and developments from within your own org.

S-Docs – free document generator (PDF, Word, Excel)

This app is developed by ME2 Systems.

Create and e-mail custom quotes, tailored contracts, personalized newsletters, and more. This app is incredibly simple to use and set up. It is completely integrated and Chatter compatible. It is 100 percent native Force.com. It can generate the PDF, MS Word, and Excel documents in minutes.

CMSForce 2

This app is developed by Force.com Labs.

This is a native Force.com web content management system. It allows you to define templates for your web pages and create/edit pages with a WYSIWYG HTML editor. It also includes a web form builder to move information entered by visitors into any object in Salesforce.

FormFactory quotes and invoices

This app is developed by DreamFactory Software, Inc.

FormFactory generates business forms including quotes, proposals, invoices, and packing slips. Documents can be delivered as live web forms, PDF, or HTML files. Salesforce users can create professional-quality forms for free with our visual design tools!

Chatter Usage Dashboards – Force.com Labs

This app is developed by Force.com Labs.

Chatter Adoption Dashboard includes 20 dashboard components and reports for a broad view into your org's usage of Chatter. Extend it with your own new reports by using seven Chatter custom report types included in this app!

FinancialForce Accounting for Salesforce

This app is developed by Accounting & PSA from FinancialForce.com.

No more rekeying data to create invoices! FinancialForce cloud accounting app resides on the same platform as Salesforce. The two systems share the same data. Billing, invoicing, accounts payable, reporting; it all works seamlessly inside Salesforce CRM.

Professional Services Automation – PSA for Salesforce

This app is developed by Accounting & PSA from FinancialForce.com.

FinancialForce Professional Services Automation includes timecards, billing, expenses, project management, and accounting. Manage people, customers, projects, and financials in one integrated services management app. Our PSA app is Salesforce native.

CVM Supplier Central Enterprise Edition

This app is developed by CVM Solutions.

CVM Supplier Central Enterprise Edition addresses supplier management needs with a comprehensive solution, empowering companies to centralize supplier information across the organization, cut costs, and mitigate risks in the supplier base.

CVM Supplier Locator

This app is developed by CVM Solutions.

CVM Supplier Locator leverages advanced faceted search technology to empower users to quickly and easily find new or alternate sources of supply. Tap into CVM's Master DB of millions of suppliers to find the suppliers needed to support your business.

Summary

Phew! That was a lot of information, but we tried our best to keep it brief. Let us just summarize what we discussed in this chapter. We talked about Data Loader, when to use it, and a strategy to plan for data import. We then briefly covered the Import wizard, how and when we should use it, and how to undo the data import. After the standard functionalities, we quickly talked about the third-party tools available in the market for achieving data integration with on-premise and other hosted business applications. Next was Force.com IDE, where we discussed some key features of the Eclipse plugin for Force.com development. Then we looked at the marketplace called AppExchange, where developers can publish and showcase their products and offerings. And last but not least, we provided our recommendations for some free/paid app listings that exist on AppExchange.

3
Making Best Use of Salesforce Objects

Now we are ready to take a more detailed look at Salesforce objects. Let us start with a brief introduction of Force.com database. The powerful and reliable data persistence layer in the Force.com stack is known as the Force.com database. This resides at the core of the platform and powers most of it. The declarative web interface, called the Force.com builder, allows users to create objects and fields, define relationships between objects, declare various application components, and generate the native application UI around them without actually writing even a single line of code.

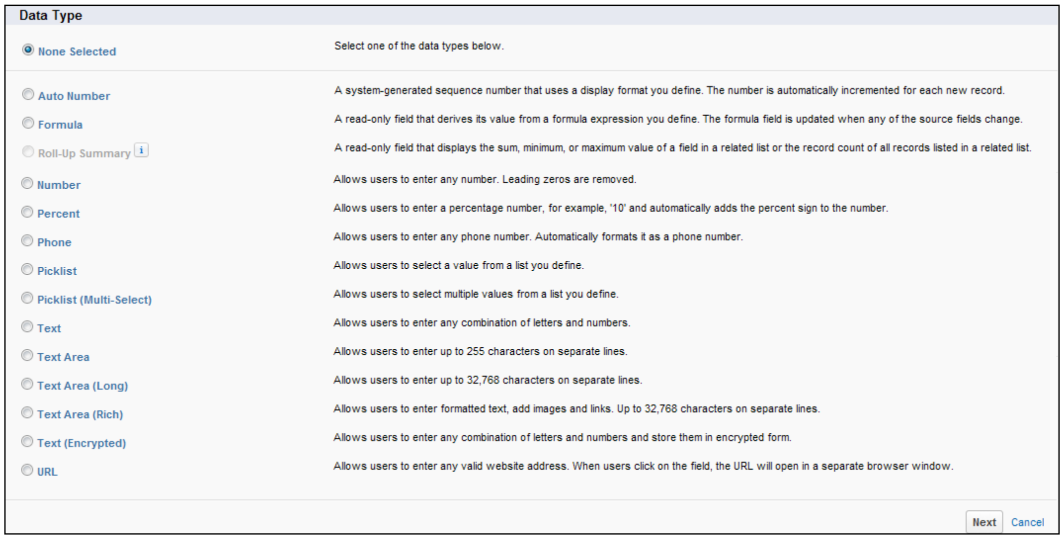
As compared with a relational database, it is similar in the sense that the object (a data instance) and fields are analogous to tables and columns, and Force.com relationships are similar to the referential integrity constraints in a Relational DB. But unlike physically separate tables with dedicated storage, Force.com objects are maintained as a set of metadata interpreted on the fly by the runtime engine and all of the application data is stored in a set of few large database tables. This data is represented as virtual records based on the interpretation of tenants' customizations stored as metadata.

Now that we know the basics of the Force.com database and how it differs from a relational database, let us cover the following items in this chapter:

- Different field data types in Salesforce
- Various considerations for defining relationships between objects
- Key standard objects (for CRM)
- Custom objects—design and implementation strategy
- Record types—what are they and why are they important?

Understanding the field types

The Force.com database offers fields of various data types such as common scalar and special data types, which otherwise either do not exist or make application development a complicated task in a relational database. Again, keep in mind that these exist as logical and not physical fields in the system.



Data Type

[✓ None Selected]	Select one of the data types below.
Auto Number	A system-generated sequence number that uses a display format you define. The number is automatically incremented for each new record.
Formula	A read-only field that derives its value from a formula expression you define. The formula field is updated when any of the source fields change.
Roll-Up Summary	A read-only field that displays the sum, minimum, or maximum value of a field in a related list or the record count of all records listed in a related list.
Number	Allows users to enter any number. Leading zeros are removed.
Percent	Allows users to enter a percentage number, for example, '10' and automatically adds the percent sign to the number.
Phone	Allows users to enter any phone number. Automatically formats it as a phone number.
Picklist	Allows users to select a value from a list you define.
Picklist (Multi-Select)	Allows users to select multiple values from a list you define.
Text	Allows users to enter any combination of letters and numbers.
Text Area	Allows users to enter up to 255 characters on separate lines.
Text Area (Long)	Allows users to enter up to 32,768 characters on separate lines.
Text Area (Rich)	Allows users to enter formatted text, add images and links. Up to 32,768 characters on separate lines.
Text (Encrypted)	Allows users to enter any combination of letters and numbers and store them in encrypted form.
URL	Allows users to enter any valid website address. When users click on the field, the URL will open in a separate browser window.

Basic non-relational field types

Most of the Salesforce field types can be found in many relational databases. Here's a brief summary of the supported data types:

- **Auto Number**: System-generated read-only sequence number, which is useful for generating unique IDs, other than the internal object IDs, which are non writeable fields.
- **Checkbox**: Boolean data.
- **Email**, **Phone**, and **URL**: Format-validated e-mail, phone, and URL string representations.
- **Data** or **Date Time**: Represent dates or date and time combinations.
- **Number**: Represent real numbers, with optional decimal points.
- **Currency**: A formatted number type, with optional multi-currency support.
- **Picklist** or **Multi Select Picklist**: Represent values from a list.
- **Text** or **Text Area**: Represent text of various lengths.

- **Text (Encrypted)**: This field allows users to enter any combination of letters, numbers, or symbols (up to 175 characters) that are stored in encrypted form, encrypted with 128-bit master keys using the **AES** (**Advanced Encryption Standard**) algorithm. Using this field you can enable master encryption key management and contact Salesforce.com.
- **Geolocation**: This is currently in beta release and has some known limitations (Winter 13). This field allows users to specify a location by its latitude and longitude.
- **Formula**: A read-only field holding data generated from a formula expression.

Relational field types

Unlike the relational database, which use keys for maintaining relationships, Force.com utilizes the relationship fields, which hold the ID of the parent record. There are three types of relationship fields in Force.com:

- **Hierarchical Relationship**: This relationship creates a hierarchical lookup relationship between users. This also allows users to use a lookup field to associate one user with another that does not directly or indirectly refer to itself. For example, you can create a custom hierarchical relationship field to store each user's direct manager.
- **Lookup Relationship**: This creates a relationship that links one object to another object. This relationship field allows navigating from records in one object to the related records in another object (both visually and programmatically).
- **Master-Detail Relationship**: This provides tighter binding than the lookup relationship. It creates a special type of relationship between two objects—the child (detail) and the parent (master). For every detail record in a master-detail relationship, Force.com requires a relationship field value, and once the record is created, restricts any further update to it by default. However, administrators can allow reparenting of records to different parent records. A master record deletion triggers cascading delete action for all dependent detail records; for example, deletion of forums should mean removal of forum posts as well.

  The master object in a master-detail relationship can also contain the **Rollup summary** fields. These fields contain values calculated from the aggregate functions performed over the child records in a relationship; for example, count of child records, sum of values in a field of child records, maximum/minimum value of a field in a set of filtered child records, and so on.

Although Salesforce does not natively support the concept of many-to-many relationships between two objects, there is a nice little work around for creating a custom object, called the junction object, containing two master-detail relationships. This is covered in more detail in the *Consideration for relationships* section.

Identity fields

Data in this field is completely managed by the Force.com database. It is either stored in a 15-digit case sensitive form or an 18-digit case insensitive form. Every record has such an identifier and it provides a convenient shortcut to retrieve and display the record. You must have observed Salesforce forming URLs such as `https://na3.salesforce.com/00190000006J48S`.

Notice here that `00190000006J48S` is the record identifier, and when it is used in the preceding format, retrieves the record data and the associated metadata and uses it to render an appropriate user interface on the browser.

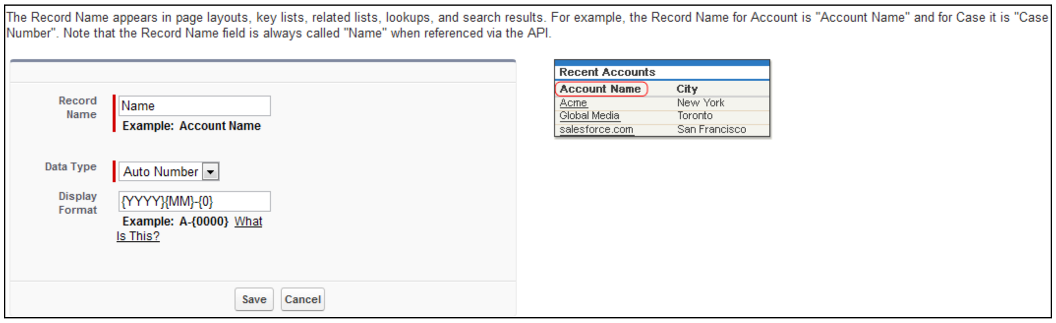
System fields

These are system-managed read-only fields present in all objects, some of which under special circumstances can be updated programmatically. The `ID` field is one of them and the rest are as follows:

- `CreatedDate`: The date and time when the object was created
- `CreatedById`: The ID of the user who created the object
- `LastModifiedById`: The ID of the user who last modified the object
- `LastModifiedDate`: The date and time when the object was last modified by a user
- `SystemModStamp`: The date and time when the object was last modified by a user or process, such as a trigger

The Name field

This is a required field that acts like a human-readable record identifier. Even though it is mandated to be unique, but that's what it is intended for. This field is mandatory on the native page layouts and forms the link for navigating to a record's detail page. It can be of two types: a text string or an auto number field, in which case we have to specify the format of the field and the starting number. With each new record, the auto number field increments by one and is non-writable.


The Record Name appears in page layouts, key lists, related lists, lookups, and search results. For example, the Record Name for Account is "Account Name" and for Case it is "Case Number". Note that the Record Name field is always called "Name" when referenced via the API.

Record Name	Name
	Example: Account Name
Data Type	Auto Number
Display Format	{YYYY}{MM}-{0}
	Example: A-{0000} What Is This?

Recent Accounts

Account Name	City
Acme	New York
Global Media	Toronto
salesforce.com	San Francisco

Additional database features

The Force.com database goes beyond the conventional relation DBs and provides additional features that speed up application development drastically. Let's talk about them briefly:

- **Formulas**: This field reflects calculations based on other fields and operations on those fields. For example:

```
IF(
AND(Payment_Due_Date__c < TODAY(),
ISPICKVAL(Payment_Status__c, "UNPAID")),
"PAYMENT OVERDUE",
null )
```

  This formula determines if the payment due date is past and the payment status is `UNPAID`. If so, it returns the text `PAYMENT OVERDUE` and if not, it leaves the field blank. This example uses a custom date field called `Payment Due Date`, a text custom field called `Payment Status` on contracts, and a number of formula operators and functions, including `IF`, `AND`, `ISPICKVAL`, and `TODAY`.

- **Validation rules**: This field helps prevent users from saving incorrect data and display appropriate error message to the user. These rules utilize the same formula syntax to define a formula that is evaluated every time a record is saved. If the formula evaluates to `False`, the save operation is aborted and an error message is displayed.
- **Labels and help**: Every object and record has a label and can include a description (for internal documentation) and help, which gets automatically included in the natively generated user interface.
- **Triggers**: Similar to those in a relational DB, triggers in the platform-specific languages, such as Apex, can be invoked before/after insert, update, delete, or undelete.

- **Notes and attachments**: Users can view, add, or edit notes and upload attachments for individual records. This functionality can be turned on/off for any object.
- **Field History Tracking**: User can turn on history tracking on certain fields so that when any change is made a new entry gets created in the **History** related list, which tracks the old value, new value, date, time, nature of change, and who made the change.
- **Security**: The database service layer offers various security features for data protection:
    - **Administrative security**: Used to allow or disallow a particular set of users certain areas of the Force.com platform functionality.
    - **Object-level security**: Define CRUD permissions for users.
    - **Field-level security**: Controls the field visibility and editing ability for users.
    - **Record security**: Controls the individual record level security through sharing. Record owners have full control over that record and can share it with others if it is set to private.
    - **Permission sets**: Collection of settings and permissions that extend users' functional access without changing their profiles.

Considerations for relationships

Before defining relationships between objects, review the following considerations:

- **Relationship limits**: Note that each custom object can have up to two master-detail relationships and many lookup relationships.
- **Changing and converting relationships**:
    - After you have created a relationship, you can't change which objects are related via that relationship. If you need to do this, delete the relationship and create a new relationship.
    - You can convert a master-detail relationship to a lookup relationship as long as no roll-up summary fields exist on the master object.
    - You can convert a lookup relationship to a master-detail relationship, but only if the lookup fields in all records contain a value.

- **Self relationships**: You can create a relationship from an object to itself, but it must be a lookup relationship, and a single record can't be linked to itself. However, a record can indirectly relate to itself. You can't create a many-to-many self relationship; that is, the two master-detail relationships on the junction object can't have the same master object.
- **Master-detail relationships**:
    - You can have up to three custom detail levels.
    - Standard objects can't be on the detail side of a custom object in a master-detail relationship.
    - An object can appear once in multilevel master-detail relationships. For example, a subdetail object in one multilevel master-detail relationship can't be the owner of the master object in another multilevel master-detail relationship. Also, a subdetail object can't be the master object of the subdetail object's detail object.
    - You can't create a master-detail relationship if the custom object already contains data. You can, however, create the relationship as a lookup and then convert it to master-detail if the lookup fields in all records contain a value.
    - Converting relationships from lookup to master-detail or from master-detail to lookup behaves the same as for two-object master-detail relationships. That is, the two linked objects in the detail-subdetail or subdetail1-subdetail2 relationship have the same conversion limits as the master-detail relationship.
    - Roll-up summary fields work the same as two-object master-detail relationships. A master can roll up fields on detail records, but it can't directly roll up fields on subdetail records. To achieve this, the detail record must have a roll-up summary field for the field on the subdetail record, allowing the master to roll up from the detail's roll-up summary field.
    - Custom junction objects can't have detail objects. That is, a custom junction object can't become the master object in a multilevel master-detail relationship.
    - When you delete a custom object that is on the detail side of a master-detail relationship, the relationship is converted to a lookup relationship. If you restore the custom object, you must manually convert it to a master-detail.
    - As a best practice, don't exceed 10,000 child records for a master-detail relationship.

- **Many-to-many relationships**:
    - Junction object records are deleted when in associated master record is deleted and placed in the Recycle Bin. If both associated master records are deleted, the junction object record is deleted permanently and can't be restored. Sharing access to a junction object record is determined by a user's sharing access to both associated master records and the **Sharing Setting** option on the relationship field. Refer to the *Custom objects* section in this chapter.

      For example, if the sharing setting on both parents is **Read/Write**, then the user must have read/write access to both parents in order to have read/write access to the junction object. If, on the other hand, the sharing setting on both masters is **Read-Only**, a user with read-only rights on the master records would have the read/write access to the junction object.
    - In a many-to-many relationship, a user can't delete a parent record if there are more than 200 junction object records associated with it and if the junction object has a roll-up summary field that rolls up to the other parent. To delete this object, manually delete the junction object records until the count goes below 200.
    - The first master-detail relationship you create on your junction object becomes the primary relationship. This affects the following aspects of the junction object records:
        - **Look and feel**: The junction object's detail and edit pages use the color and any associated icon of the primary master object.
        - **Record ownership**: The junction object records inherit the value of the **Owner** field from their associated primary master record. Because objects on the detail side of a relationship do not have a visible **Owner** field, this is only relevant if you later delete both master-detail relationships on your junction object.
        - **Division**: If your organization uses divisions to segment data, the junction object records inherit their divisions from their associated primary master records. Similar to the record ownership, this is only relevant if you later delete both master-detail relationships.
    - The second master-detail relationship you create on your junction object becomes the secondary relationship. If you delete the primary master-detail relationship or convert it to a lookup relationship, the secondary master object becomes primary.

  - The roll-up summary fields that summarize data from the junction object can be created on both master objects.
  - The formula fields and validation rules on the junction object can reference the fields on both master objects.
  - You can define the Apex triggers on both the master object and the junction object.
  - A junction object can't be on the master side of another master-detail relationship.
  - You can't create a many-to-many self relationship; that is, the two master-detail relationships on the junction object can't have the same master object.
  - Workflow rules and approval processes on junction objects can be created, but you can't create outbound messages on junction objects.

Types of objects

Two types of object exist in any org:

- **Standard objects**: Objects that are created and made available by Salesforce.com as soon as the org is set up are called standard objects. These objects are mostly customizable to some extent.
- **Custom objects**: Objects that you create in your org to store information unique to your business are called custom objects. These are usually created to build custom applications.

Standard objects

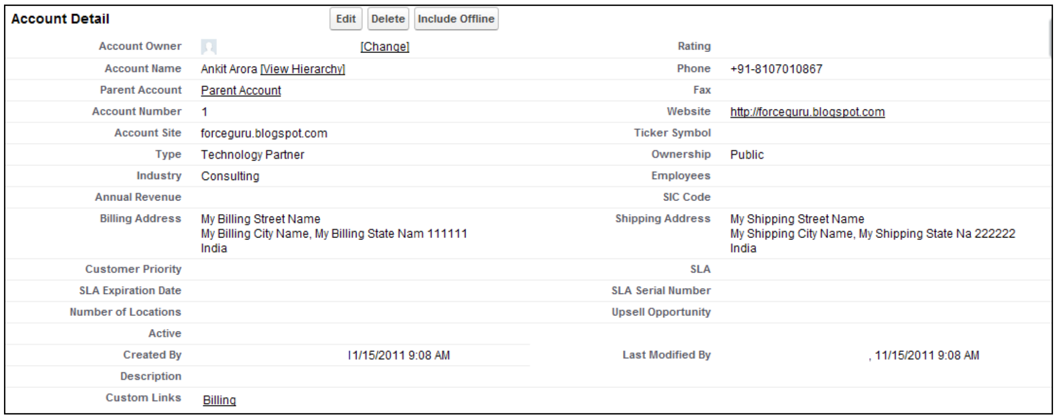
These comprise the heart and soul of the platform and are part of the out-of-the-box CRM application in an org. The list of standard objects is long, but we can discuss the most important ones that form the core of CRM functionality.

Account

Accounts represent your organization's customers, competitors, and any other business entities that you deal with. In CRM application, almost anything and everything rolls up to account in one way or another through a relationship. Each account stores information such as name, address, and phone numbers. For each account, you can store related information such as opportunities, activities, cases, partners, contracts, and notes. An account can also be linked to another account to create an account hierarchy to accommodate different corporate structures representing a company and its subsidiaries and/or sales territories.

You can build an account team on each account that you own. When selecting an account team member, depending on your sharing model, you can specify the level of access each account team member will have to the account and any contacts, opportunities, or cases associated with that account. So, you can give some team members read-only access and others read/write access.

You can grant access to accounts based on the characteristics of the accounts such as zip code, industry, revenue, or a custom field that is relevant to your business. This type of account sharing system is called **territory management**. It enables your company to structure your Salesforce data and users in the same way as you structure your sales territories.



Account Owner	[Change]	Rating	
Account Name	Ankit Arora [View Hierarchy]	Phone	+91-8107010867
Parent Account	Parent Account	Fax	
Account Number	1	Website	http://forceguru.blogspot.com
Account Site	forceguru.blogspot.com	Ticker Symbol	
Type	Technology Partner	Ownership	Public
Industry	Consulting	Employees	
Annual Revenue		SIC Code	
Billing Address	My Billing Street Name My Billing City Name, My Billing State Nam 111111 India	Shipping Address	My Shipping Street Name My Shipping City Name, My Shipping State Na 222222 India
Customer Priority		SLA	
SLA Expiration Date		SLA Serial Number	
Number of Locations		Upsell Opportunity	
Active			
Created By	I1/15/2011 9:08 AM	Last Modified By	, 11/15/2011 9:08 AM
Description			
Custom Links	Billing		

Contact

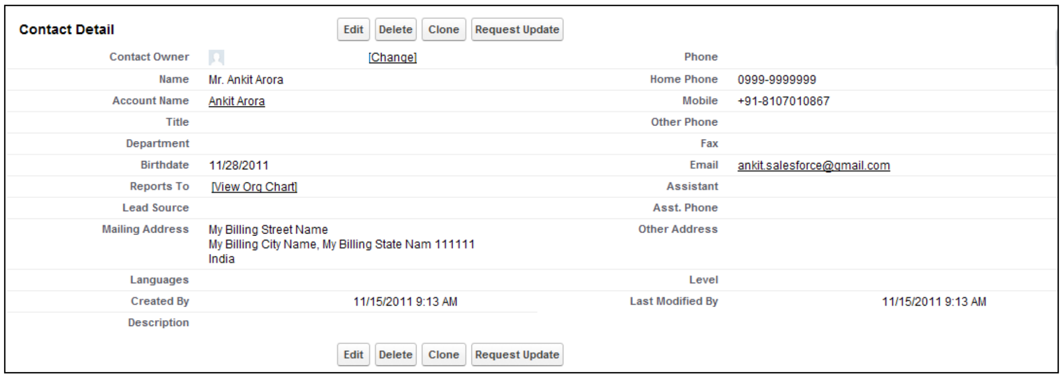
Contacts are the people associated with your business accounts that you need to track in Salesforce. For each contact, you can store various kinds of information, such as phone numbers, addresses, titles, and roles in a deal.

Social contacts have been introduced to enhance your traditional contact data. With social contacts, you can see your contacts' social networking profiles, directly in Salesforce. Easy access to this information helps you know your customers better, so you can solve their problems and build stronger relationships. You can see social information from several social networks such as LinkedIn, Twitter, and Facebook.

You can also link a contact to another contact by specifying the contact's manager in the **Report To** field. This enables displaying an organization chart showing the contact hierarchy when you click on the **View Org Chart** link on any contact's page in the account.

You can assign a contact role to any contact that affects your account, case, contract, or opportunity. Contacts can have different contact roles on various accounts, cases, contracts, or opportunities. A contact role defines the part that a contact or a person account plays in a specific account, case, contract, or opportunity.

You can also allow your customers to access your Salesforce org by creating a portal user for a contact, provided the corresponding account has the Customer or Partner Portal enabled on it.


Contact Detail

Contact Owner	[Change]	Phone	
Name	Mr. Ankit Arora	Home Phone	0999-9999999
Account Name	Ankit Arora	Mobile	+91-8107010867
Title		Other Phone	
Department		Fax	
Birthdate	11/28/2011	Email	ankit.salesforce@gmail.com
Reports To	[View Org Chart]	Assistant	
Lead Source		Asst. Phone	
Mailing Address	My Billing Street Name My Billing City Name, My Billing State Nam 111111 India	Other Address	
Languages		Level	
Created By	11/15/2011 9:13 AM	Last Modified By	11/15/2011 9:13 AM
Description			

Lead

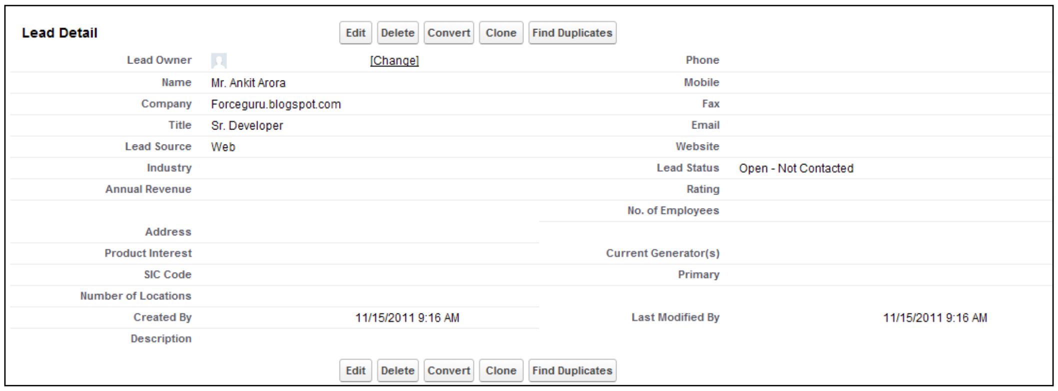
A lead is a prospect or potential opportunity—a person you met at a conference who expressed interest or someone who filled out a form on your company's website.

You can enter into leads manually from the **Leads** tab, or your administrator can import leads or set up Web-to-Lead to gather information from your company's website. Users can also import leads via the campaign import wizards, if they have the **Marketing User** checkbox checked on their user information and the **Marketing User** profile (or the **Import Leads** permission and the **Edit** permission on campaigns).

Your administrator can create a lead assignment rule to automatically assign leads to different users or queues. Your administrator can assign new web-generated leads by using the Web-to-Lead setup, or when creating or editing a lead, you can check a box to assign the lead automatically by using your active lead assignment rule.

A lead can also be converted. When you convert a lead, Salesforce creates a new account, contact, and, optionally, an opportunity by using the information from the lead. Any campaign members are moved to the new contact and the lead becomes read only. If an existing account and contact have the same names as those specified on the lead, you can choose to update the existing account and contact. Information from the lead is inserted only into blank fields; Salesforce does not overwrite existing account and contact data.

All notes and attachments from the lead are converted and attached to the new account and contact. All open activities and the activity history from the lead are converted and attached to the new account, contact, and opportunity. The converted lead record can no longer be viewed, although it does contribute data to the reports.



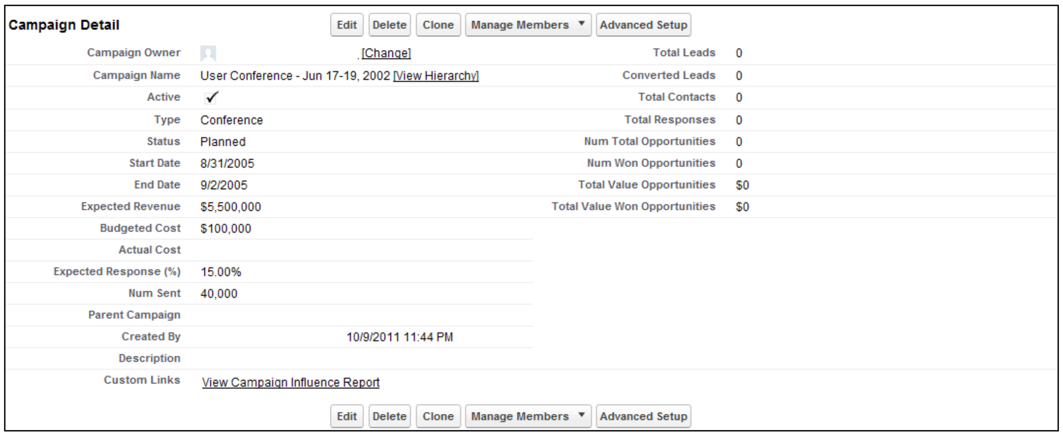
Lead Owner	[Change]	Phone	
Name	Mr. Ankit Arora	Mobile	
Company	Forceguru.blogspot.com	Fax	
Title	Sr. Developer	Email	
Lead Source	Web	Website	
Industry		Lead Status	Open - Not Contacted
Annual Revenue		Rating	
		No. of Employees	
Address			
Product Interest		Current Generator(s)	
SIC Code		Primary	
Number of Locations			
Created By	11/15/2011 9:16 AM	Last Modified By	11/15/2011 9:16 AM
Description			

Campaign

A campaign is an outbound marketing project that you want to plan, manage, and track within Salesforce. It can be a direct mail program, seminar, print advertisement, e-mail, or other type of marketing initiative. You can organize campaigns into hierarchies for easy analysis of related marketing tactics.

By associating campaigns with one another using a lookup relationship, you can group campaigns within a specific marketing program or initiative. A hierarchy can contain a maximum of five levels. Each campaign can have only one parent campaign, but an unlimited number of sibling campaigns. To view the hierarchy for a campaign, click on **View Hierarchy** next to the **Campaign Name** field on the **Campaign Detail** page. If a campaign is not part of a hierarchy, its corresponding **Campaign Hierarchy** page shows only the campaign you have selected.

You can also manually or automatically associate multiple influential campaigns to a single opportunity. You can view influential campaigns from the **Campaign Influence** related list on the **Opportunity Detail** page. The **Primary Campaign Source** field on an **Opportunity Detail** page allows you to designate the most influential campaign for that opportunity.



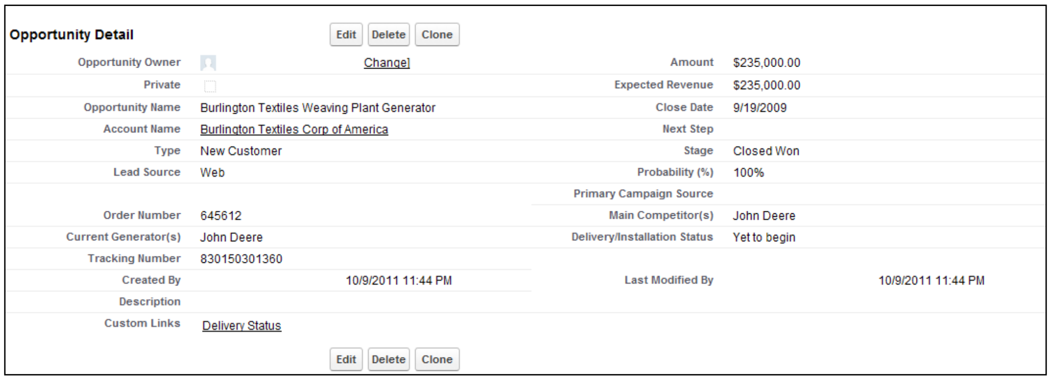
Campaign Owner	[Change]	Total Leads	0
Campaign Name	User Conference - Jun 17-19, 2002 [View Hierarchy]	Converted Leads	0
Active	✓	Total Contacts	0
Type	Conference	Total Responses	0
Status	Planned	Num Total Opportunities	0
Start Date	8/31/2005	Num Won Opportunities	0
End Date	9/2/2005	Total Value Opportunities	$0
Expected Revenue	$5,500,000	Total Value Won Opportunities	$0
Budgeted Cost	$100,000		
Actual Cost			
Expected Response (%)	15.00%		
Num Sent	40,000		
Parent Campaign			
Created By	10/9/2011 11:44 PM		
Description			
Custom Links	View Campaign Influence Report		

Opportunity

Opportunities are the sales and pending deals that you want to track. By adding opportunities, you are also building your *pipeline*, which will contribute to your forecast. You can also link opportunities to campaigns to help measure the **ROI** (**return on investment**) of your marketing programs. In addition, you can create quotes that show proposed prices for products and services, from an opportunity.

You need to identify, at some point of time in your sales process, the decision makers who influence the buying decision, as contacts and their titles don't directly represent the influencers and decision makers. The Contact Roles features help you specify this chain of command in an opportunity.

You can also define sales team and track competitors on an opportunity.



Opportunity Owner	Change]	Amount	$235,000.00
Private		Expected Revenue	$235,000.00
Opportunity Name	Burlington Textiles Weaving Plant Generator	Close Date	9/19/2009
Account Name	Burlington Textiles Corp of America	Next Step	
Type	New Customer	Stage	Closed Won
Lead Source	Web	Probability (%)	100%
		Primary Campaign Source	
Order Number	645612	Main Competitor(s)	John Deere
Current Generator(s)	John Deere	Delivery/Installation Status	Yet to begin
Tracking Number	830150301360		
Created By	10/9/2011 11:44 PM	Last Modified By	10/9/2011 11:44 PM
Description			
Custom Links	Delivery Status		

Forecast

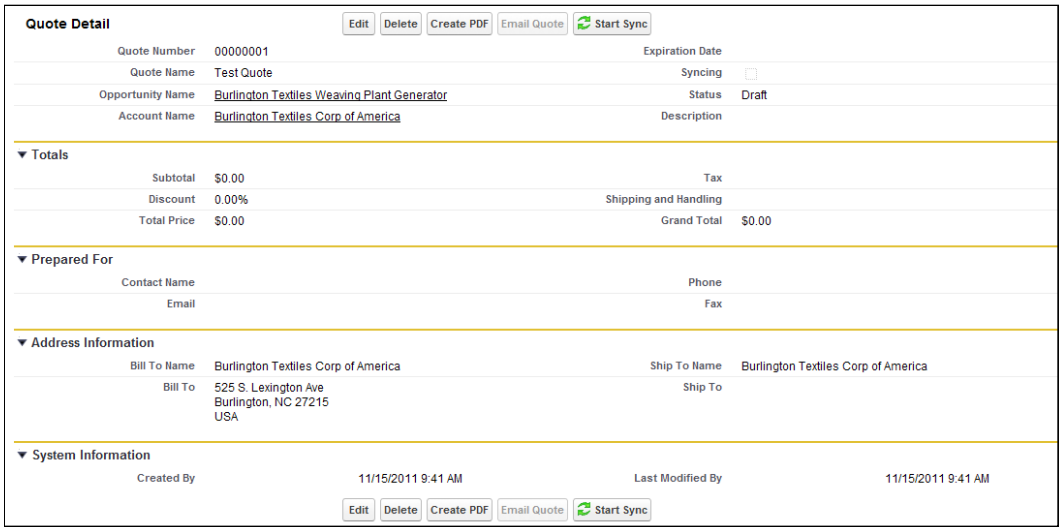
Using forecasts, you can predict and plan your sales cycle from pipeline to closed sales, and manage sales expectations throughout your organization. A forecast is your best estimate of how much revenue you can generate in a quarter. The forecast amount is based upon your pipeline and is the total amount of all Commit opportunities divided by the total amount of all Best Case opportunities.

You can set up customizable forecasting to reflect how your organization forecasts its sales. Customizable forecasting is a flexible solution for estimating how much revenue your organization can generate or how many items your organization can sell. With it, you can forecast on a monthly or quarterly basis, use different dates when applying amounts to forecasts, forecast based on revenue or quantity or both, and define additional quotas based on product families.

Quote

A quote is a record showing proposed prices for products and services. You create a quote from an opportunity and its products. Each opportunity can have multiple associated quotes, and any one of them can be synced with the opportunity. When a quote and an opportunity are synced, any change to line items in the quote will sync with products on the opportunity, and vice versa.

When your quote is complete, you can generate a PDF and e-mail it to your customer. Quote PDFs are based on templates. Salesforce provides a standard template, and you can also create your own.



Quote Detail

Quote Number	00000001	Expiration Date	
Quote Name	Test Quote	Syncing	☐
Opportunity Name	Burlington Textiles Weaving Plant Generator	Status	Draft
Account Name	Burlington Textiles Corp of America	Description	

Totals

Subtotal	$0.00	Tax	
Discount	0.00%	Shipping and Handling	
Total Price	$0.00	Grand Total	$0.00

Prepared For

Contact Name		Phone	
Email		Fax	

Address Information

Bill To Name	Burlington Textiles Corp of America	Ship To Name	Burlington Textiles Corp of America
Bill To	525 S. Lexington Ave Burlington, NC 27215 USA	Ship To	

System Information

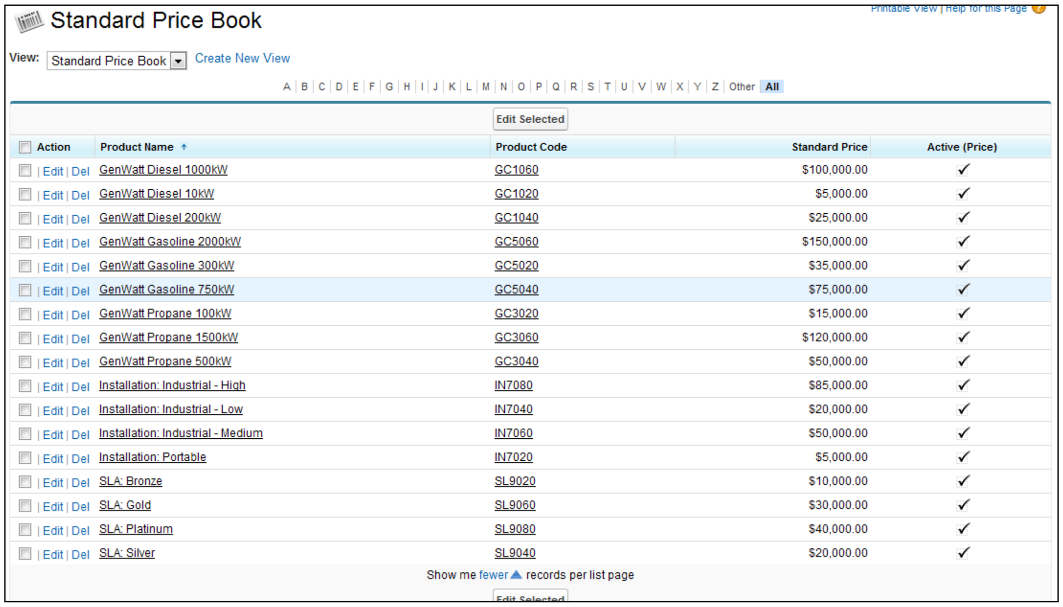
Created By	11/15/2011 9:41 AM	Last Modified By	11/15/2011 9:41 AM

Product and price book

The product and price book objects are closely related. Technically, the product and price book have a many-to-many relationship handled by the junction object `PricebookEntry`.

- **Products**: Products are the individual items that you sell on your opportunities and quotes. You can create a product and associate it with a price in a price book. Each product can exist in many different price books with many different prices. A product that is listed in a price book with an associated price is called a price book entry. The **Products** related list on an **Opportunity Detail** page and the **Quote Line Items** related list on a **Quote Detail** page list the products for that record. Use this related list to associate a price book with the opportunity or quote, add, or edit products, and for opportunities, establish or edit product schedules.

- **Price books**: A price book contains products and their associated prices. Each product with its associated price is referred to as a price book entry. You can use the standard price book or create custom price books. The standard price book is automatically generated to contain a master list of all products and standard prices regardless of the custom price books that also contain them.

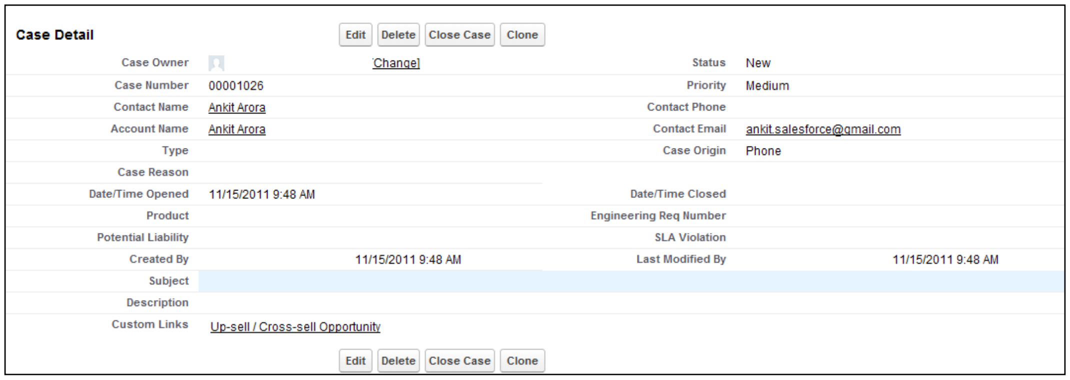

Printable View | Help for this Page
Standard Price Book
View: Standard Price Book
Create New View
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All
Edit Selected

Action	Product Name	Product Code	Standard Price	Active (Price)
Edit | Del	GenWatt Diesel 1000kW	GC1060	$100,000.00	✓
Edit | Del	GenWatt Diesel 10kW	GC1020	$5,000.00	✓
Edit | Del	GenWatt Diesel 200kW	GC1040	$25,000.00	✓
Edit | Del	GenWatt Gasoline 2000kW	GC5060	$150,000.00	✓
Edit | Del	GenWatt Gasoline 300kW	GC5020	$35,000.00	✓
Edit | Del	GenWatt Gasoline 750kW	GC5040	$75,000.00	✓
Edit | Del	GenWatt Propane 100kW	GC3020	$15,000.00	✓
Edit | Del	GenWatt Propane 1500kW	GC3060	$120,000.00	✓
Edit | Del	GenWatt Propane 500kW	GC3040	$50,000.00	✓
Edit | Del	Installation: Industrial - High	IN7080	$85,000.00	✓
Edit | Del	Installation: Industrial - Low	IN7040	$20,000.00	✓
Edit | Del	Installation: Industrial - Medium	IN7060	$50,000.00	✓
Edit | Del	Installation: Portable	IN7020	$5,000.00	✓
Edit | Del	SLA: Bronze	SL9020	$10,000.00	✓
Edit | Del	SLA: Gold	SL9060	$30,000.00	✓
Edit | Del	SLA: Platinum	SL9080	$40,000.00	✓
Edit | Del	SLA: Silver	SL9040	$20,000.00	✓

Case

A **case** is a description of a customer's feedback, problem, or question. You can use cases to track and solve your customers' issues. Customer feedback can be gathered from a company's website either by customers or support representatives (Web-to-Case) and/or direct e-mails from customers (E-mail-to-Case). Your customers can also create cases on your Self-Service or Customer Portal. New cases can be assigned to support agents, case teams, or case queues using assignment rules.

To manage all your cases, the tools that you need for routing, queuing, escalating cases, replying with knowledge articles or solutions, and so on are all provided out of the box with the Case functionality.



Case Detail

Case Owner	[Change]	Status	New
Case Number	00001026	Priority	Medium
Contact Name	Ankit Arora	Contact Phone	
Account Name	Ankit Arora	Contact Email	ankit.salesforce@gmail.com
Type		Case Origin	Phone
Case Reason			
Date/Time Opened	11/15/2011 9:48 AM	Date/Time Closed	
Product		Engineering Req Number	
Potential Liability		SLA Violation	
Created By	11/15/2011 9:48 AM	Last Modified By	11/15/2011 9:48 AM
Subject			
Description			
Custom Links	Up-sell / Cross-sell Opportunity		

Custom objects

As mentioned earlier, these represent the custom data set that is unique and important to you and your organization. For example, you may want to create a custom object called `AccountLocation_c` with custom attributes to store data for your customers' different office locations.

Design approach

It is highly recommended that you first decide your overall approach. Gather and analyze requirements and finalize the design decisions up front to avoid bigger problems in future. Answering the questions discussed in the following section may help you reach to a conclusion.

Data

Analyze the data that is going to reside in the system. Find answers to the following questions:

- What fields and data types would you need?
- Does the functionality relate to the CRM object model? Can any existing object be extended?
- How would you logically group the new custom fields into one or more custom objects?

Relationships

Identify how the objects will be related. Find answers to the following questions:

- Is the custom data object related to standard data, or is it independent? If data is related, it should appear as a related list on a standard tab. If it is independent, it should appear on a custom tab.
- If it is a related data, how tight is the data binding? If a record in one object is deleted, should the related records in the other objects be also deleted?

User interface

Identify how the user interface is going to look like. Here are few questions that you need to answer:

- Do you want to create custom tabs for users to maintain data or is it ok if they maintain the data from related lists on existing tabs?
- What fields should appear in the page layouts?
- What fields should appear in the related lists?
- Do you want users to track tasks and events for the custom object?
- Do you want to imitate the Salesforce UI?

Implementation steps

When building custom objects, tabs, and related lists, follow these steps:

Use the following links to get started:

- For custom objects, click on **Your Name** | **Setup** | **Create** | **Objects**
- For custom tabs for a custom object, click on **Your Name** | **Setup** | **Create** | **Tabs**

To create a custom tab, users can track data that is specific to your business:

- Design approach
- Create custom object
- Create custom tab
- Customize page layouts

To create a custom related list, you can associate your data with standard tabs:

- Design approach
- Create custom object
- Define relationships
- Customize related lists

Tips

Here are a few tips on various topics discussed earlier in this chapter.

Custom objects

A few tips to keep in mind while designing custom objects are as follows:

- First establish object relationships before adding all custom fields, page layouts, and related lists.
- Click on **Edit List Layout** to choose columns for key views and lookups.
- The standard `Name` field is required for custom object related lists and page layouts.
- Provide meaningful names for your custom objects. The plural label of the custom object is also used as the label of the custom tab based on that object.
- Build custom reports and dashboards, using the data in your custom objects.

Relationships

A few tips on object relationships:

- Each custom object can have up to two master-detail relationships and up to 25 lookup relationships
- The Related To entry can now be changed starting from the Summer 12 release
- Create a master-detail relationship before a custom object that contains data

Custom tabs

A few tips to keep in mind while creating custom tabs:

- You can create a certain number of custom object tabs or web tabs based on your Salesforce Edition; for example, 25 in Enterprise Edition, 100 in Developer Edition, and as many as you want in Unlimited Edition
- The title of the custom tab is the same as the plural label of the custom object

- It is recommended to select the **Append** tab to the user's existing personal customizations checkbox
- Set permissions on tabs and page layouts so that users cannot see your changes until they are finalized

Page layouts

A few tips for using page layouts:

- In Enterprise, Unlimited, and Developer Editions, use the field-level security to restrict users' field access; use page layouts primarily to organize pages.
- To reduce the number of page layouts to maintain, use the same page layout for all profiles for a specific record type.
- In Personal, Group, Contact Manager, and Professional Editions, field-level security is not available. Use page layouts to restrict access to fields and to organize pages.
- Mini page layouts contain subsets of items in an existing page layout and are displayed in the **Console** tab, hover details, and event overlays. These inherit record type and profile associations, related lists, fields, and field access settings from their associated page layout.

Record types

Record types is an extremely powerful feature offered by Salesforce.com. This feature allows you to offer different business processes, picklist values, and page layouts to different users based on their profiles. In other words, records of the same object adopt different behaviors in the native user interface.

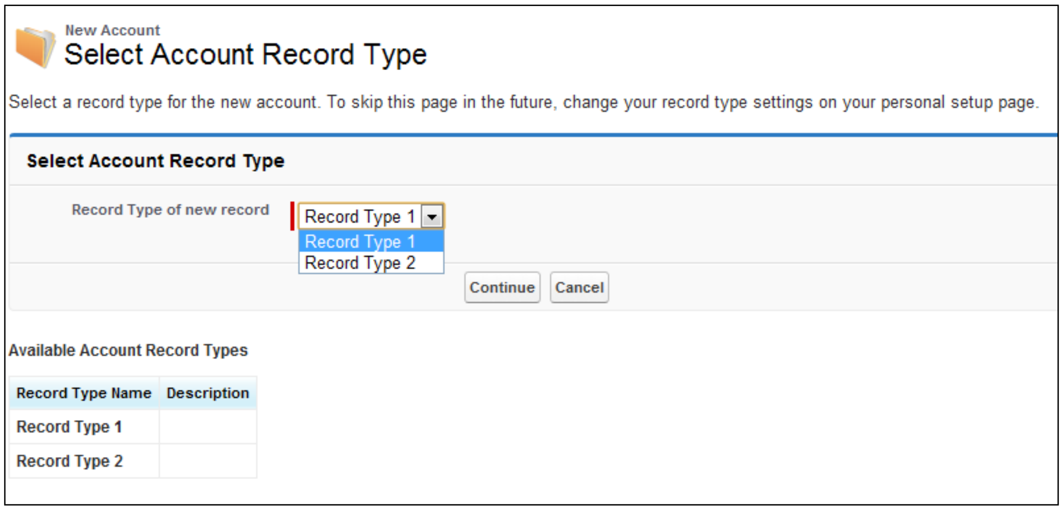
For example, an Account object usually represents business entities that your company deals with, but with minor tweaks it can also be used to store information about individuals, and it's then called a Person Account. Salesforce intentionally decided to go with the same object, just with a different record type.

Similarly, whenever you are tempted to create a new object, you should explore whether it's entirely a new object or whether its data can be merged with another entity with minor tweaks. In the case of minor adjustments, a single object can be used to store variations of business entity and Force.com renders an appropriate user interface based on the record type it finds.

By default, objects do not use record types; but when you define it, an additional standard field called `RecordTypeId` gets added to the object, which can then be utilized extensively either in native functionalities or can also be used in driving custom user interfaces.

Record types can be assigned to profiles and so different page layouts can be shown to different users based on the record's record type. The picklist values can also be assigned to record types so that the appropriate values get displayed on the user interface.

For example, an issue can either be a bug or an enhancement and fields displayed on the user interface for both will vary slightly, like bug will show an additional field of `Steps to Reproduce` whereas it is not required in case of enhancement. So when a user tries to create a new issue in the system, he/she is presented with a page where he/she has to choose from the available record types, in this case `Bug` and `Enhancement`, and the subsequent edit and detail page layouts will vary based on the record type chosen.


New Account
Select Account Record Type

Select a record type for the new account. To skip this page in the future, change your record type settings on your personal setup page.

Select Account Record Type

Record Type of new record: Record Type 1

Available Account Record Types

Record Type Name	Description
Record Type 1	
Record Type 2	

A record type limitation is enforced by the platform only when the record is being created and it can be skipped by specifying a default record type on an object. The record type of a record can be changed at a later point in time by the record owner.

Summary

To recap, we talked about different field types that exist in Salesforce.com—don't forget the master-detail and lookup types as they are key to any business application. We then discussed when to choose master-detail over lookup and vice versa, their pros and cons, and limitations. We then discussed some standard objects that are important from the CRM perspective. Later we looked at custom objects and discussed strategies to define and implement them in an efficient manner. Lastly, we briefly covered record types, their strengths, and how they can be utilized to create powerful business applications with point and click operations only.

4
Understanding Analytics

Data is the key to your customer's business. Almost every application is designed and developed only to collect data that is critical to the business, which can then be analyzed later. There could be a lot of information inside your system that may complicate things and ultimately conceal what you should know. So the data has to be viewed in a logical and user-friendly manner such that it makes sense. What is needed is a simple dashboard of your data inside Salesforce, which can be comprehended even with a cursory look. Salesforce.com provides a comprehensive analytics and reporting system, which can be used to organize, view, and analyze your data so as to provide real-time visibility into the business. The best part here is that it requires neither a steep learning curve, nor a technical person's involvement in developing reports that matter to you and your customers.

It's a very common mistake made by Salesforce developers that when designing the object model, only the transactional needs are considered and analytical requirements are almost always ignored. This leads to an object model that proves ineffective down the line when reporting needs are realized and analyzed in later stages. This may even lead to complete redesign of the application that has a negative impact on client faith and your business.

The Salesforce analytics, which we will cover in this chapter, can be summarized as follows:

- Report types are the templates from which users can build reports
- Reports organize data and are shared via folders
- Dashboard components, which are report driven, are also shared via folders
- Analytic snapshot is a feature that allows reporting on the historical data by persisting a specific report's data in a custom object

Let us now take a closer look at the various terms that we used in the preceding paragraph.

Report types

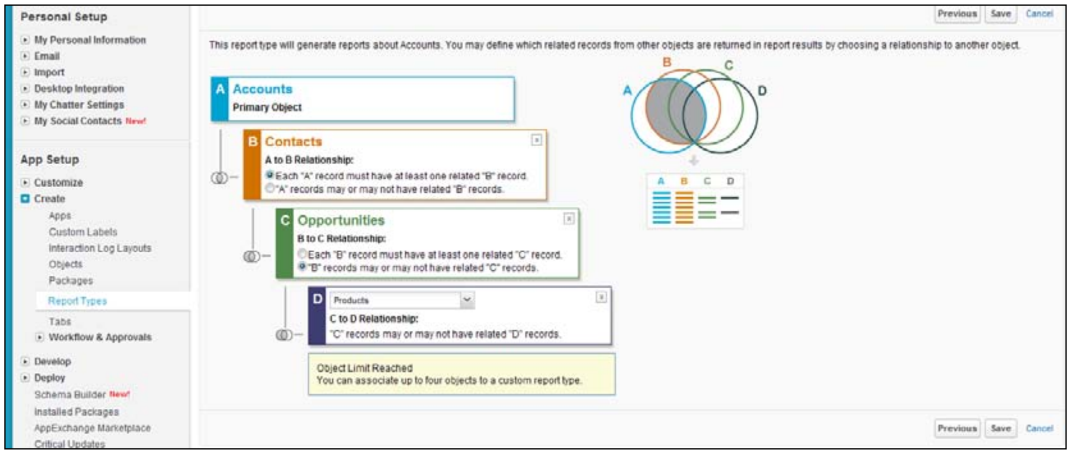
A report type defines the set of records and fields available to a report based on the relationships between a primary object and its related objects; reports then display only those records that meet the criteria defined in the report type.

Salesforce provides a large set of pre-defined or standard report types, but at the same time it also allows you to create custom report types when you want to report on data based on your customizations. Custom report types allow you to build a template or framework in the report wizard, from which users can create and customize reports.

Salesforce allows you to bring together up to four related objects, and create stronger and more impressive reports via custom report types. However, there is a limit to the maximum number of custom report types (regardless of development status) that can be created in an org. This limit varies depending on the org edition; for example, for Enterprise it's 200 whereas for Unlimited it's 2000.

You can create a custom report type via **Your Name** | **Setup** | **Create** | **Report Types**.

See the following screenshot that displays the four level objects relationship based hierarchy, which can be the master-detail or lookup type. Users can choose which standard and custom objects to display to users by creating and customizing reports, define the relationships (type of join) between objects, that is, whether it will be an inner join or a left outer join, and select which objects' fields can be used as columns in reports.



This report type will generate reports about Accounts. You may define which related records from other objects are returned in report results by choosing a relationship to another object.

A Accounts
Primary Object

B Contacts
A to B Relationship:
[✓ Each "A" record must have at least one related "B" record.]
"A" records may or may not have related "B" records.

C Opportunities
B to C Relationship:
Each "B" record must have at least one related "C" record.
[✓ "B" records may or may not have related "C" records.]

D Products
C to D Relationship:
"C" records may or may not have related "D" records.

Object Limit Reached
You can associate up to four objects to a custom report type.

For example, in the preceding screenshot, A and B have an inner join, but B and C have a left outer join.

Once a primary object for a custom report type has been selected, it can't be changed later.

Tips and considerations for report types

Keep the following tips and limitations in mind when you begin creating custom report types for your organization.

Defining report types

Consider the following points when defining report types:

- If the selected primary object is a custom object, its deletion will cause the associated custom report types and reports to be automatically deleted. Report types associated with custom objects in the deleted custom objects list are counted against the maximum number of custom report types you can create.
- Removal of an object from a report type causes all references to that object and its associated objects to be automatically removed from reports and dashboards based on that type.

- As of Spring 2012 release, a custom report type can contain up to 60 object references and 1000 fields.
- Forecasts can't be added to custom report types.
- The following fields can't be added to custom report types:
    - Product Schedule fields
    - History fields
    - Person Account fields
    - The `Age` field on Case and Opportunity

Choosing object relationships

Consider the following points when choosing the child object relationship:

- If you select that object A may or may not have object B, all subsequent (tertiary and quaternary) objects automatically default to the may-or-may-not association on the custom report type.
- Blank fields are displayed in the report results for object B when object A does not have object B. For example, if a user runs a report on accounts with or without contacts, the contact fields are displayed as blank for accounts without contacts.
- On reports where object A may or may not have object B, you can't use the *or* condition to filter across multiple objects.
- In a custom report type with Account as the primary and Contact as the secondary object with may-or-may-not association selected, the **Row Limit** option on tabular reports shows only the fields from Account and not Contacts or subsequent objects.
- Arrange fields on the layout sections as they should appear to users and preview the field display of the layout created. If required, create new sections to logically group similar fields on the field layout.
- Preselect the commonly-used fields by selecting the **Checked by Default** property of the field on the field layout.

Adding lookup fields

Consider the following points when adding fields via lookup to the field layout page of a custom report type:

- A custom report type can contain fields available via lookup through four levels of lookup relationships. For example, for an account you can get the account owner, the account owner's manager, the manager's role, and that role's parent role.
- You can only add fields via lookups that are associated with objects included in the custom report type. For example, for a custom report type with Contact as one of the objects, you can add fields from objects to which Contact has a lookup relationship.
- The fields displayed in the **Add Fields Via Lookup** overlay do not include lookup fields to primary objects. For example, if contacts are the primary objects on your custom report type and cases are the secondary objects, the **Add Fields Via Lookup** overlay does not display the lookup fields from cases to contacts.
- Fields added to the layout via the **Add Fields Related Via Lookup** link are automatically included in the section of the object from which they are a lookup field.
- If you include activities as the primary object on a custom report type, you can only add lookup fields from activities to accounts on the select column layout of the custom report type.

Reports

A report is a set of records that meets certain criteria which is displayed in organized rows and columns. In a report, you can select columns, filter your data, group, subtotal, and limit your data, conditionally highlight it, embed formulas, display it graphically as a chart, and export it in different formats.

Salesforce provides a strong and intuitive report builder that allows even untrained users to create a report by guiding them through a wizard. The report builder allows you to carry out all the functions related to a report almost visually and makes it very easy for a non-technical user. There are three types of report formats that Salesforce supports:

- **Tabular**: These are the simplest and fastest to build and display data just like how you view in a spreadsheet. They consist of an ordered set of fields in columns, with each matching record listed in a row, and are best for creating lists of records or a list with a single grand total. They can't be used to create groups of data or charts, and can't be used in dashboards unless rows are limited.

- **Summary**: They are similar to tabular reports, but also allow for grouping rows of data, view subtotals, and create charts. They can be used as the source report for dashboard components. Summary reports with no groupings are shown as tabular reports on the report run page.
- **Matrix**: Similar to summary reports, but more complex and powerful, these allow you to group and summarize data by both rows and columns. They can also be used as source reports for dashboard components. You can use this report format type for comparing related totals, especially if you have large amounts of data to summarize and you need to compare values in several different fields. Matrix reports without at least one row and one column grouping are shown as summary reports on the report run page.
- **Joined**: These let users view different types of information in a single report. For example, your report could contain Opportunities, Cases, Accounts, and even Data from custom apps and objects; that is, these reports can contain data from multiple standard or custom report types, provided they have relationships with the same object or objects. You can add up to 16 report types, add up to five report blocks, and also create standard and cross-block custom summary formulas, add a chart, filter individual blocks using standard and Boolean filters, sort columns for each block, and more. An example of such a report could be to predict future opportunity revenue based on your sales reps' past performance, using cross-block custom summary formulas!

The information that users see in reports is only the data to which they have access. This includes records they own, records to which they have read or read/write access, records that have been shared with them, records owned by or shared with users in lower roles in the hierarchy, and records for which they have "Read" permissions. In addition, they can view only those fields that are visible in their page layout and field-level security settings. (Field-level security is available only in Enterprise, Unlimited, and Developer Editions.)

Organizing reports

If not planned and managed properly, reports can grow exponentially in your org and you can lose track. So it is advised that you start doing this from day one and that you define a process that provides a tighter control over the creation and deletion of reports.

The best way to organize reports is by placing them in different folders. This will allow you to group them logically and also define who has access to what based on roles, permissions, public groups, and license types. A folder can be made public, hidden, or shared, and can be set to read-only or read/write. You can also make a folder available to the entire organization or make it private so that it's accessible to the owner only.

You should consider the following points to manage your report library in a better way:

- Report names should be intuitive so that your users can identify their contents just by looking at the name.
- You may optionally define a report naming convention, which makes use of numbers embedded in report names so that it's sometimes easier to refer to numbers than names.
- Do a periodic clean up of unnecessary reports in your org, and even if you delete an in-use report, you can still recover it from the recycle within 30 days from deletion date.
- It's good to keep track of reports utilizing the picklist values in the filter criteria. This will help you keep those reports up-to-date as and when picklist values get changed.
- On a periodic basis, keep a check on the number of folders existing in your org and delete or hide them as necessary.

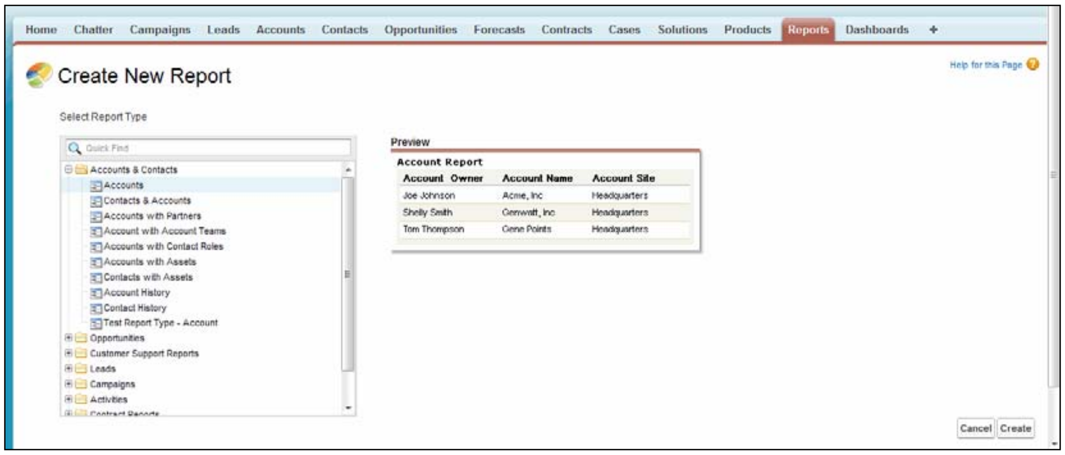
Working with reports

Here is a brief walkthrough of the report creation process and things that you need to consider while designing them.

Creating reports

Salesforce provides a user interface for building reports via the intuitive and easy-to-use report builder, provided you are able to articulate what is needed by the business users and in what format. Reports are created against a primary object and a set of related objects. Salesforce provides certain combinations of primary and related objects that are called **standard report types**. You can also develop your own combinations by using custom report types.

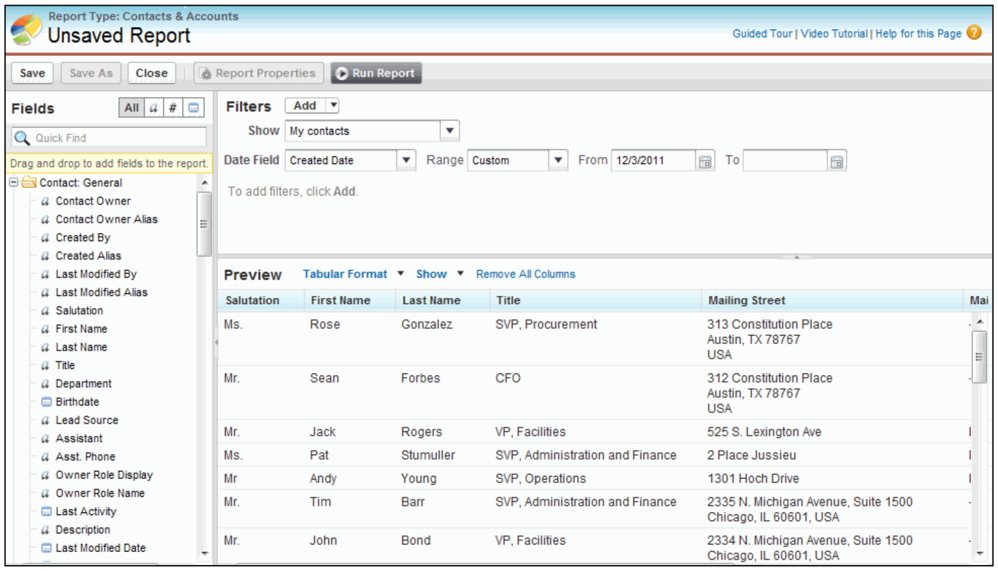
From the **Enhanced Reports** tab, you can create a new report by just selecting a report type categorized by folders, as follows:



Account Report

Account Owner	Account Name	Account Site
Joe Johnson	Acme, Inc	Headquarters
Shelly Smith	Genwatt, Inc	Headquarters
Tom Thompson	Gene Points	Headquarters

After the report type has been selected, define the report fields:

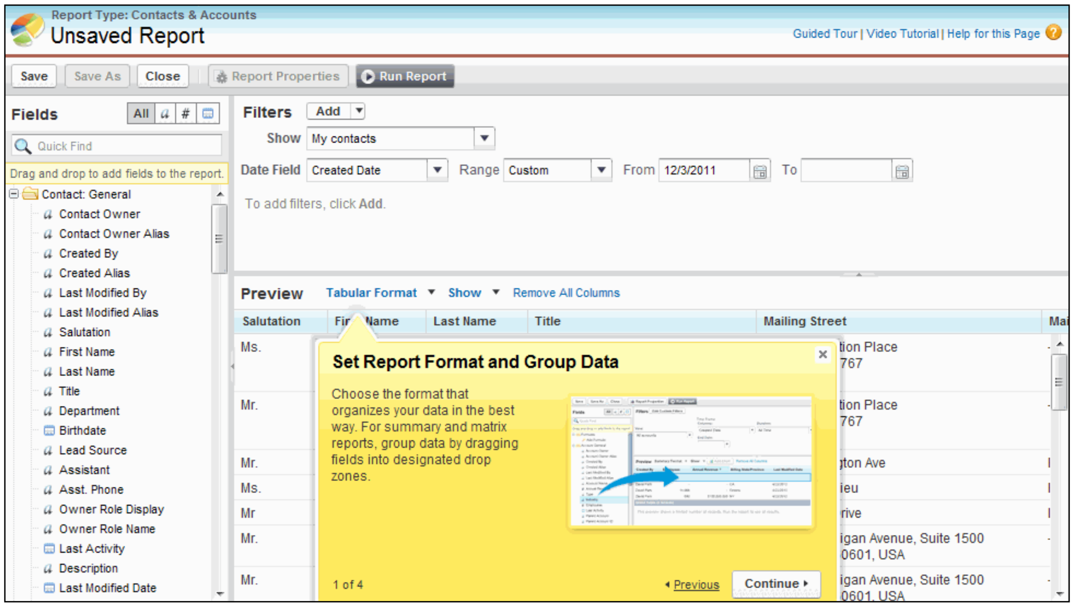


Show: My contacts

Date Field: Created Date | Range: Custom | From: 12/3/2011 | To:

Salutation	First Name	Last Name	Title	Mailing Street	Mai
Ms.	Rose	Gonzalez	SVP, Procurement	313 Constitution Place Austin, TX 78767 USA	
Mr.	Sean	Forbes	CFO	312 Constitution Place Austin, TX 78767 USA	
Mr.	Jack	Rogers	VP, Facilities	525 S. Lexington Ave	
Ms.	Pat	Stumuller	SVP, Administration and Finance	2 Place Jussieu	
Mr	Andy	Young	SVP, Operations	1301 Hoch Drive	
Mr.	Tim	Barr	SVP, Administration and Finance	2335 N. Michigan Avenue, Suite 1500 Chicago, IL 60601, USA	
Mr.	John	Bond	VP, Facilities	2334 N. Michigan Avenue, Suite 1500 Chicago, IL 60601, USA	

After a report has been created, you can then choose between tabular, summary, and matrix formats, add and reorder groupings on summary and matrix reports, and summarize fields on reports as indicated in the following screenshot:


Report Type: Contacts & Accounts
Unsaved Report
Guided Tour | Video Tutorial | Help for this Page

Save | Save As | Close | Report Properties | Run Report

Fields All | Quick Find
Drag and drop to add fields to the report.
Contact: General
- Contact Owner
- Contact Owner Alias
- Created By
- Created Alias
- Last Modified By
- Last Modified Alias
- Salutation
- First Name
- Last Name
- Title
- Department
- Birthdate
- Lead Source
- Assistant
- Asst. Phone
- Owner Role Display
- Owner Role Name
- Last Activity
- Description
- Last Modified Date

Filters Add
Show: My contacts
Date Field: Created Date | Range: Custom | From: 12/3/2011 | To:
To add filters, click Add.

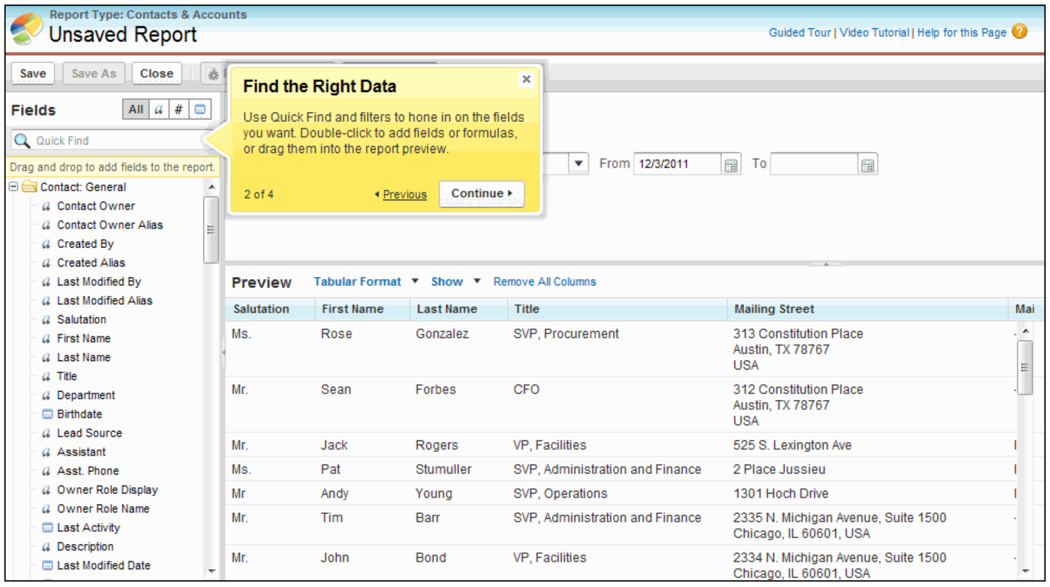
Preview Tabular Format | Show | Remove All Columns

Set Report Format and Group Data

Choose the format that organizes your data in the best way. For summary and matrix reports, group data by dragging fields into designated drop zones.

1 of 4 | Previous | Continue

You can find fields by using **Quick Find** and field-type filters, add fields to the report by double-clicking or dragging them into the **Preview** pane, and sort and reorder columns, as indicated in the following screenshot:


Report Type: Contacts & Accounts
Unsaved Report
Guided Tour | Video Tutorial | Help for this Page

Save | Save As | Close

Find the Right Data

Use Quick Find and filters to hone in on the fields you want. Double-click to add fields or formulas, or drag them into the report preview.

2 of 4 | Previous | Continue

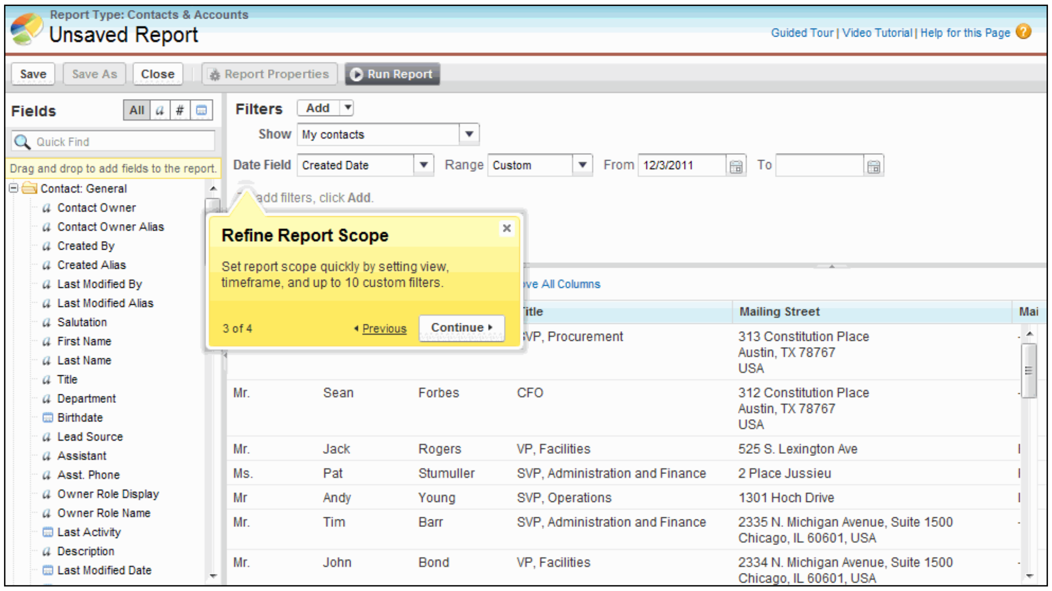
From: 12/3/2011 | To:

Fields All | Quick Find
Drag and drop to add fields to the report.
Contact: General
- Contact Owner
- Contact Owner Alias
- Created By
- Created Alias
- Last Modified By
- Last Modified Alias
- Salutation
- First Name
- Last Name
- Title
- Department
- Birthdate
- Lead Source
- Assistant
- Asst. Phone
- Owner Role Display
- Owner Role Name
- Last Activity
- Description
- Last Modified Date

Preview Tabular Format | Show | Remove All Columns

Salutation	First Name	Last Name	Title	Mailing Street
Ms.	Rose	Gonzalez	SVP, Procurement	313 Constitution Place Austin, TX 78767 USA
Mr.	Sean	Forbes	CFO	312 Constitution Place Austin, TX 78767 USA
Mr.	Jack	Rogers	VP, Facilities	525 S. Lexington Ave
Ms.	Pat	Stumuller	SVP, Administration and Finance	2 Place Jussieu
Mr	Andy	Young	SVP, Operations	1301 Hoch Drive
Mr.	Tim	Barr	SVP, Administration and Finance	2335 N. Michigan Avenue, Suite 1500 Chicago, IL 60601, USA
Mr.	John	Bond	VP, Facilities	2334 N. Michigan Avenue, Suite 1500 Chicago, IL 60601, USA

You can set report filters in the **Filters** pane to help further narrow your result set:



Show: My contacts
Date Field: Created Date Range: Custom From: 12/3/2011 To:

add filters, click Add.

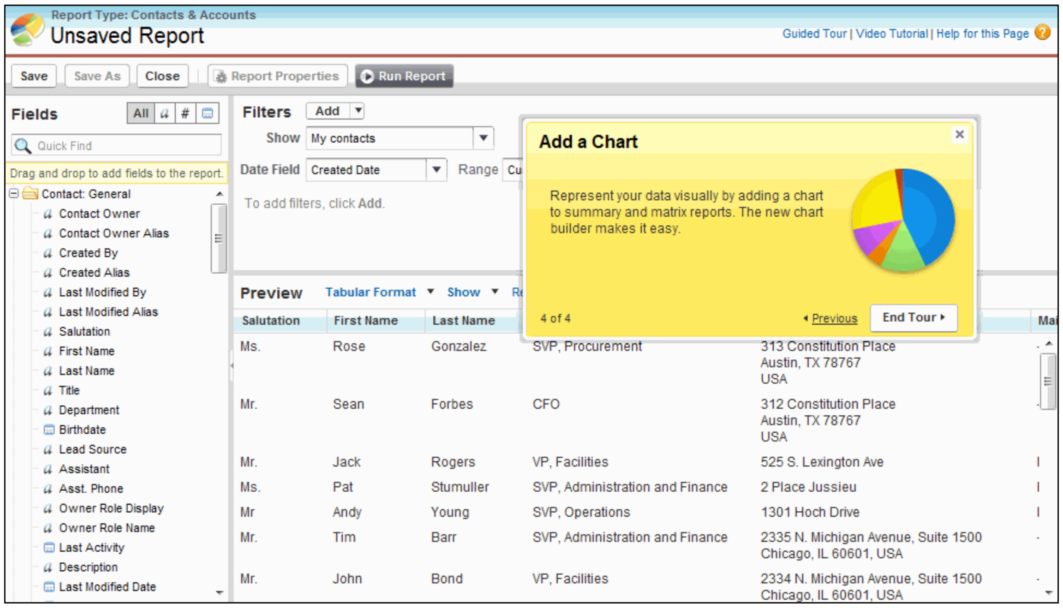
Refine Report Scope

Set report scope quickly by setting view, timeframe, and up to 10 custom filters.

3 of 4 Previous Continue

			itle	Mailing Street	Mai
			VP, Procurement	313 Constitution Place Austin, TX 78767 USA	
Mr.	Sean	Forbes	CFO	312 Constitution Place Austin, TX 78767 USA	
Mr.	Jack	Rogers	VP, Facilities	525 S. Lexington Ave	
Ms.	Pat	Stumuller	SVP, Administration and Finance	2 Place Jussieu	
Mr	Andy	Young	SVP, Operations	1301 Hoch Drive	
Mr.	Tim	Barr	SVP, Administration and Finance	2335 N. Michigan Avenue, Suite 1500 Chicago, IL 60601, USA	
Mr.	John	Bond	VP, Facilities	2334 N. Michigan Avenue, Suite 1500 Chicago, IL 60601, USA	

You can optionally add a chart to your report to add a graphical summary of your results:



Show: My contacts
Date Field: Created Date Range: Cu

To add filters, click Add.

Add a Chart

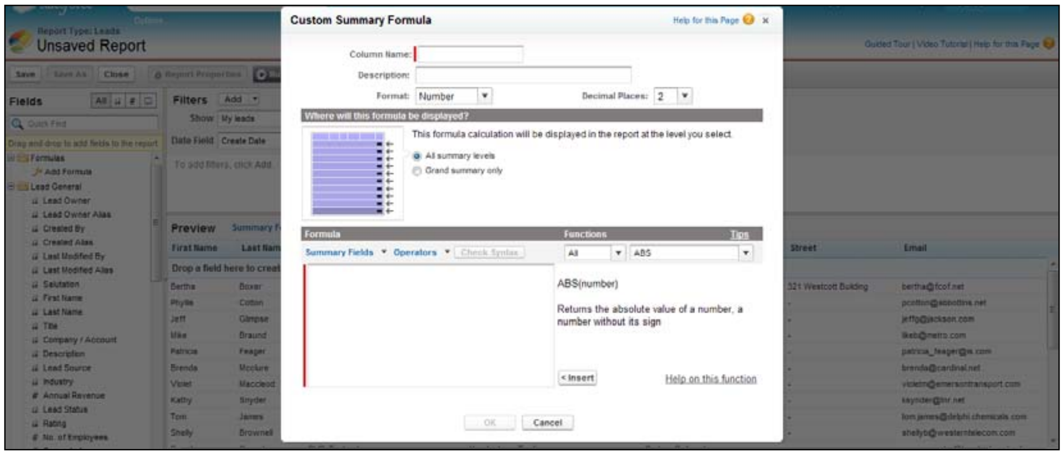
Represent your data visually by adding a chart to summary and matrix reports. The new chart builder makes it easy.

4 of 4 Previous End Tour

Salutation	First Name	Last Name			Mai
Ms.	Rose	Gonzalez	SVP, Procurement	313 Constitution Place Austin, TX 78767 USA	
Mr.	Sean	Forbes	CFO	312 Constitution Place Austin, TX 78767 USA	
Mr.	Jack	Rogers	VP, Facilities	525 S. Lexington Ave	
Ms.	Pat	Stumuller	SVP, Administration and Finance	2 Place Jussieu	
Mr	Andy	Young	SVP, Operations	1301 Hoch Drive	
Mr.	Tim	Barr	SVP, Administration and Finance	2335 N. Michigan Avenue, Suite 1500 Chicago, IL 60601, USA	
Mr.	John	Bond	VP, Facilities	2334 N. Michigan Avenue, Suite 1500 Chicago, IL 60601, USA	

Building custom summary formulas

Salesforce provides an out-of-the-box functionality to calculate the sum, average, and the highest and lowest of the numeric fields selected in your reports. However, there may be a business need to provide additional summary information with some specific calculations. Salesforce allows you to create custom summary formulas to calculate additional totals based on the numeric fields available in the report type. You can create up to five formulas per report, which can't be shared across reports.



Custom Summary Formula

Column Name:

Description:

Format: Number

Decimal Places: 2

Where will this formula be displayed?

This formula calculation will be displayed in the report at the level you select

[✓ All summary levels]

Grand summary only

Formula

Functions: All, ABS

ABS(number)

Returns the absolute value of a number, a number without its sign

Consider the following points when creating custom summary formulas:

- A summary formula can't reference another summary formula.
- Dashboard and report charts that display values from custom summary formulas display decimal places, using your default currency setting instead of what you specified for the formula. For example, if the summary formula specifies zero decimal places, no decimal places appear in columns, but chart values show the number of decimal places specified for your default currency (usually two decimal places). This applies to currencies, numbers, and percentages.
- Regardless of the summary formula data type, your summary formula can contain fields of different data types, including the number, currency, percent, and checkbox (true/false) fields.
- Percents are represented as decimals in summary formulas. 20 percent is represented as 0.20
- When fields are deleted, they are also deleted from the summary formulas that reference them.

- The summary types Sum, Largest Value, Smallest Value, and Average are not available to use with the **Record Count** field.
- `"#Too Big!"` is displayed on report cells if your custom summary formula output is over 21 digits. When this happens, check your formula for calculations that could result in more than 18 digits.
- Formulas treat blank (null) report cells as zero values.
- `"#Error!"` is displayed on report cells whenever an error occurs while calculating a formula's value. `"#Error!"` is also displayed when a formula includes division by zero.

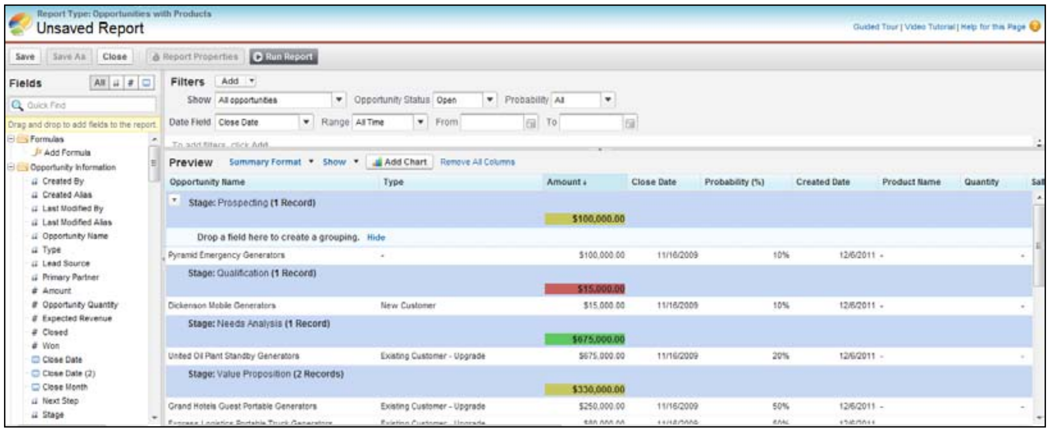
Bucketing in reports

Bucketing is a concept introduced by Salesforce to let you quickly categorize report records without creating a formula or a custom field. There are only a few types of fields that can't be bucketed. You can bucket the numeric, picklist, and text fields. Bucket fields can be used like any other fields to sort, filter, and group your report. The difference is once you create a bucket field in a report, the field is only available in that report. So if you want to use the same bucket field criteria in another report, you need to recreate it in that report. A simple and practical use case of data bucketing could be grouping the Account records based on the value selected in the picklist type **Industry** field into buckets of Industry Type (group of similar industries) in reports. Other use cases could be grouping Opportunities by size of revenue, or grouping cases by their age based on the number of days it was open.

Using conditional highlighting

You can apply conditional highlighting in your summary or matrix reports to highlight values that may be of more importance to your business users, and you may want to immediately draw your users' attention as soon as they look at the report.

You may use this along with custom summary formulas to highlight high or low percentages, averages, and ratios. These are used to visually indicate and differentiate via color coding that a threshold has been exceeded or not, instead of the user identifying manually by comparing various values. Refer to the following screenshot for an example of conditional highlighting:



Show: All opportunities; Opportunity Status: Open; Probability: All
Date Field: Close Date; Range: All Time; From: ; To:

Opportunity Name	Type	Amount	Close Date	Probability (%)	Created Date	Product Name	Quantity
Stage: Prospecting (1 Record)		$100,000.00					
Pyramid Emergency Generators	-	$100,000.00	11/16/2009	10%	12/6/2011	-	-
Stage: Qualification (1 Record)		$15,000.00					
Dickenson Mobile Generators	New Customer	$15,000.00	11/16/2009	10%	12/6/2011	-	-
Stage: Needs Analysis (1 Record)		$675,000.00					
United Oil Plant Standby Generators	Existing Customer - Upgrade	$675,000.00	11/16/2009	20%	12/6/2011	-	-
Stage: Value Proposition (2 Records)		$330,000.00					
Grand Hotels Guest Portable Generators	Existing Customer - Upgrade	$250,000.00	11/16/2009	50%	12/6/2011	-	-

Subtotaling the report results

See the preceding screenshot. In summary and matrix reports, you may group sets of information and compare subtotals for each set against the overall total, to analyze trends in the data. In a subtotal, you may also get a subtotal by multiple fields to give you cascading sets of information.

Salesforce uses *smart* totaling when you run reports that include duplicate data in any of the columns chosen for summing or averaging. **Smart totaling** means that duplicate data is counted only once in any subtotal or total. For example, if an opportunity has two products and you run the Opportunity Product Report with the total opportunity amount selected as a column to sum by, the amount appears twice in the details of the report, once for each product on the opportunity. In this case, smart totaling correctly calculates any subtotals, grand totals, and averages, adding that opportunity amount only once.

Filtering on reports

A report may contain huge data that may not be of importance to the business users. So to limit the data shown in the report, you can set standard filters, field filters, filter logic, and row limits. Depending on your organization's setup, you may see additional filters, such as probability, hierarchy, territory, and others.

To get the results you desire, filter the data in a report by using the following filter options:

- **Field Filter (Filter Criteria)**: Available for reports, list views, workflow rules, and other areas of the application. For each filter, set the field, operator, and value.
- **Filter Logic**: Add Boolean conditions to control how field filters are evaluated. You must add at least one field filter before applying filter logic.
- **Row Limit**: For tabular reports, select the maximum number of rows to display, then choose a field to sort by, and the sort order. Tabular reports that have a limited row count can be used in dashboard tables and charts.

Tips for entering filter criteria

Consider the following points when entering filter criteria:

- For faster performance:
    - Avoid using the "contains" and "does not contain" operators. Use the "equals" or "not equal to" operators instead, for faster results.
    - Instead of filtering by the **Name** field, filter by **Alias**. A **Name** search requires searching two fields, whereas **Alias** only searches one.
    - Avoid adding too many filters. You can include a maximum of 10 filters, but using fewer than five is recommended.
- Note that filtering is not case sensitive.
- If you lose access to a field defined in a filter, Salesforce removes it from the report and displays results based on the remaining filters.
- To filter on picklist values in a report, use either the "equals" or "not equal to" operators for these filters.
- When filtering on the multi-select picklist fields, use a semicolon between values to specify an exact match.
- When searching for numbers or other data that includes commas, place quotation marks around the data. For example, Amount equals "10,000" returns records that have an amount of $10,000.

- To create a filter that includes more than one value, enter your search terms, separated by commas, in the corresponding field. You can enter up to 1000 characters, including the commas.
- To limit results to records that are blank or contain "null" values for a particular field, choose the field and the "equals" or "not equal to" operators, leaving the third field blank.
- Encrypted fields are not available to use in filters such as list views, reports, roll-up summary fields, and rule filters.
- You may further use cross filters, by themselves or in combination with field filters, to fine-tune your results by including or excluding records from related objects. Just keep in mind that cross filters can potentially slow down your report, so you may need to limit data returned by setting filters.

Tips for filter logic

Consider the following tips when entering filter logic:

- When you add filter logic, include each custom filter in the Boolean expression to avoid an input error.
- Make sure all parentheses are closed.
- Enclose conditions that have priority in parentheses. For example, `(1 AND 2) OR 3` finds records that meet either the first two filters or the third. While `1 AND (2 OR 3)` finds records that meet the first filter as well as either the second or third.
- If your filter logic is `(1 AND 2) OR 3` and you add another field filter, the updated logic becomes `((1 AND 2) OR 3) AND 4`.
- You may begin with the term `NOT`, but cannot end with it. For example, `NOT 1 AND (2 OR 3 OR 4)` finds records that meet any of the last three filters and excludes records that meet the first filter.
- If you remove a custom filter, remove the corresponding number from the filter logic to avoid an input error.
- If you lose access to a field defined in a filter, Salesforce removes it from the report and displays results based on the remaining filters. This does not apply to filter logic on lookup filters.
- Filter logic isn't available for all filters. For example, you can't use them for roll-up summary fields.

Running reports

To run a report, you find the report on the **Reports** tab and click on its name. If you are viewing the report, click on the **Run Report** button to run it immediately.

Some notes:

- **No results**: If you do not see any data when you run a report, consider the following:
    - Check filter criteria to ensure it returns some data
    - Check whether you have access to all of the groupings you selected, due to field-level security
    - Check if a custom summary formula's context does not match the chart settings; for example, a formula is calculated for Industry but the chart doesn't include Industry, so no results are returned
    - Check if the values are out of the range of acceptable value
- **Visible records**: Reports show only the information you can access. This includes records you own, records to which you have read or read/write access, records that have been shared with you, records owned by or shared with users in roles below you in the hierarchy, and records for which you have the "Read" permissions.
- **Visible fields**: You can view only those fields that are visible in your page layout and the field-level security settings.
- **Printing report folder contents**: When viewing a list of reports in a particular folder, you can click on **Printable View** to open the current list view in a print-ready format.
- **Running large reports**: If your report returns more than 2,000 records, only the first 2,000 records are displayed. To see a complete view of your report results, click on **Export Details**. For security purposes, Salesforce may require users to pass a CAPTCHA-based user verification test to export data from their organization.
- **Organizations using divisions**: If you have the `Affected by Divisions` permission, you can set your report options to include records in just one division or all divisions.
- **Organizations using multiple currencies**: Amounts in reports are shown in their original currencies, and report totals are displayed in your personal currency or you can change the currency by selecting from active currencies. For any amount, you can also choose to display the converted column (for example, "Annual Revenue (converted)"), which will show amounts in the selected currency.

- **Long or rich text fields truncated**: Only the first 254 characters in a rich text area or a long text area are displayed in a report.

Scheduling a report

You can also schedule reports to run and have the results automatically e-mailed to the Salesforce users. Users with the `View Setup and Configuration` permission can view all scheduled reports for their organization on the **All Scheduled Jobs** page at **Your Name** | **Setup** | **Monitoring** | **Scheduled Jobs**. Users with the `Modify All Data` permission can click on **Del next to a specific scheduled report** to unschedule the report.

Consider the following points when scheduling a report to run:

- There is a daily limit to the number of scheduled reports, which varies with the edition you purchase. Additional scheduled reports may be available for purchase.
- Scheduled reports run in the time zone of the user who schedules them.
- If you schedule a report to run on a specific day of every month, the report runs only on months that have that specific day. For example, a report scheduled to run on the 31st day of every month will not run in February, April, June, and so on. So, to schedule a report on the last day of every month, it is wise to choose last from the **On day of every month** drop-down list.
- The report runs within 30 minutes of the time you select for **Preferred Start Time**. For example, if you select 2:00 PM as your preferred start time, the report runs any time between 2:00 PM and 2:29 PM, depending on how many other reports are scheduled at that time.
- For reports to run as scheduled, the user in the Running User field must have access to the folder in which the report is stored.
- Scheduling reports is not tracked in the audit trail history.

Consider the following points when you plan to e-mail scheduled reports:

- To e-mail a report to other users, the report must be in a public folder with access granted to the other users, because other users can't access reports in your personal folders.
- Report recipients can click on the report name in e-mailed reports to log in to Salesforce and view the report directly.
- Report charts are not included in the e-mailed reports. To e-mail a chart of the report, create a dashboard and schedule a dashboard refresh.

- The maximum size for the e-mailed reports is 10 MB. So to reduce the amount of data in your report, try the following:
    - Filter for your own records rather than all records
    - Limit the scope of the data to a specific date range
    - Exclude unnecessary columns from your report
    - Hide the report details

Printing and exporting reports

Ideally, you will always want to run your reports out of your application, but occasionally you need to print them for some meetings and/or export the report data to do some complex calculation in the excel. All of that is easily possible in few clicks. Note that you can export only up to 256 columns and 65,536 rows of data in one report. For security reasons, Salesforce may optionally ask for a CAPTCHA authentication.

Consider the following points when exporting reports:

- When exporting reports in the **CSV** (**comma-separated values**) format, the locale settings on your **User Detail** page determine the field separator (delimiter) included in the exported file; for example, if English (United States) is the locale, the decimal separator is a "." (period), so the export in the CSV format will have the field separator as "," (comma). For the French locale, where the decimal separator is a "," (comma), the field separator will be a ";" (semicolon).
- When using the 15-character, alphanumeric ID to identify a particular report record in the export, make sure that you use the correct case for the record because the ID is case sensitive.
- If you have set the **Do not save encrypted pages to disk** option in Internet Explorer, you will not be able to view your report online in Excel when you click on **Printable View** or **Export Details**. You must save the exported report to your computer, and then open it in Excel.
- You can also run reports in the background to avoid time-out issues in case of a large number of report results:
    - When the background report finishes, it is available to be viewed for 48 hours, after which it is deleted permanently and not sent to the Recycle Bin
    - You can export an unlimited number of reports to the background
    - You may optionally be presented with a CAPTCHA authentication, when you try to download the exported report results

Report charts

You may optionally add a chart to your report to add a graphical summary of your results. Use a chart editor to choose a chart type, determine what data to represent, and decide how you want to visually present that data.

Consider the following when using charts:

- If you lose access to a field used in a chart, another field may be used in its place. If no other fields are available, the record count is used.
- Decimal-place precision on charts is not customizable. Numeric and currency values round to two decimal places. Percentage values round to one decimal place.
- If numeric values are too large or too small, they are shown in scientific notation. For example, the number 5,750,000,000 is displayed as 5.75E9.
- Negative values are displayed on all line charts and non-stacked bar and column charts. Negative values on pie, donut, funnel, and stacked charts are not displayed.
- Dashboard and report charts that display values from custom summary formulas display decimal places by using your default currency setting, instead of what you specified for the formula. For example, if the summary formula specifies zero decimal places, no decimal places appear in columns, but chart values show the number of decimal places specified for your default currency (usually two decimal places). This applies to currencies, numbers, and percentages.

Combination charts

These are charts that plot multiple sets of data on a single chart. Each set of data is based on a different field, so values are easy to compare. You can also combine certain chart types to present data in different ways in a single chart.

With combination charts, you can:

- Add a line to an existing line, vertical column, grouped vertical column, or stacked vertical column chart
- Add a cumulative line to an existing cumulative line chart
- Add up to three columns to a vertical column chart
- Add up to three bars to a horizontal bar chart

Dynamic reports

As such, the reports are static and filter criteria are specified at the design time, which can't be altered at runtime. So there is no official support provided by Salesforce for report reusability. Although, there exists a hack that is being currently used by the developer community to generate dynamic reports to achieve reusability. The way in which you develop such reports is by creating a report in the normal fashion, but leaving the values of the filter criteria blank. You then invoke the reports via some custom link or button on detail pages or Visualforce pages that pass some parameters, using merge fields or other logic, in the URL of the report as query string, which are then dynamically picked up by the report and used as filter criteria values. These parameters are `pv0`, `pv1`, `pv2`, and so on. Then, `pv0` is used as the value for the first filter criteria, `pv1` is used as the value for second filter criteria, and so on.

So the URL to run the report with dynamic parameters would be something like `https://na7.salesforce.com/[Report Id]?pv0={!Opportunity.Id}&pv1={Opportunity.Division}`.

Dashboards

Dashboards are graphical representations of reports that you create in your org and visually illustrate the key metrics and performance indicators that matter to the business. They display multiple reports at once to give you a snapshot of the data of business importance and you can analyze in a quick glance where you stand and whether or not there are any concrete action items if you notice any slippage against your goals. They allow the top management to make well-informed strategic decisions to improve the overall health of your business by analyzing data trends. They can be configured to be displayed on the home page so that user can review them as soon as he/she logs in to the system.

Dashboards are collections of the dashboard components, which could further be configured to display data from reports or Visualforce pages. Components from reports can be displayed as gauges, charts (horizontal and vertical bar, line, pie, donut, funnel), tables, or metrics. Each dashboard can have up to 20 different components, but only summary and matrix report formats can be used with them.

Just like reports, dashboards are also stored in folders and all the rules that apply on the report folders apply to the dashboard folders as well. Administrators can control access to dashboards by storing them in folders with the restricted visibility settings. These folders can be public, hidden, or restricted to groups, roles, or territories. So simply put, access to a folder lets you view its dashboards. But at the same time, to access dashboard component, users need to have access to the folder for the underlying source report.

There is a concept of *running user* in dashboards, which determines access to the data The running user can be:

- **A specified/named user**: The dashboard runs by using the security settings of a single specific user. All users who can access dashboards see the specified user's view of the data, where their own personal security settings are ignored. You should use this setting when you want to give your users a holistic view of the system across a hierarchy/division or motivate your team members by showing them peer performance within a team. For example, create a sales performance dashboard to share it within the sales department and set the running user as VP Sales or any other user whoever has the highest visibility in the sales department.
- **A currently logged-in user**: These dashboards run by using the security settings of the logged-in user who is currently viewing the dashboard, so users see the dashboard according to their security settings and access level. You can use this dashboard to share one common set of dashboard components to users with different levels of access. These are what are actually called **dynamic dashboards** as they give users that have access to the dashboard a personalized view of the data.

Consider the following points when selecting a running user:

- Dashboard components that use Visualforce ignore the running user; content displays only if the viewing user has access to the Visualforce page. Other components in the dashboard are not affected.
- Consider creating separate dashboards for users with different license types because as an example, users with the Salesforce Platform or Salesforce Platform One license can only view a dashboard if the running user also has the same license type.

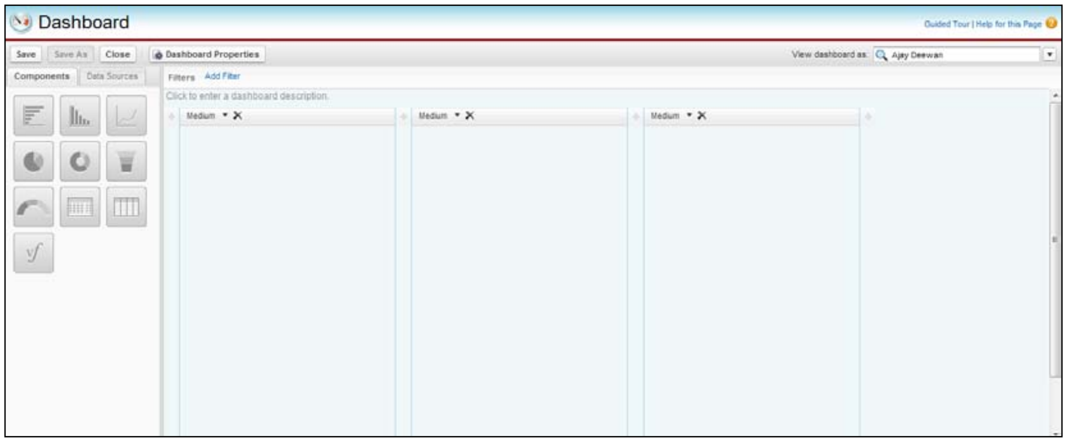
View Team Dashboards is a flavor of dynamic dashboards where managers with the `View My Team's Dashboards` or `View All Data` permissions can set an option to preview the dashboard from the point of view of the users under them in the role hierarchy.

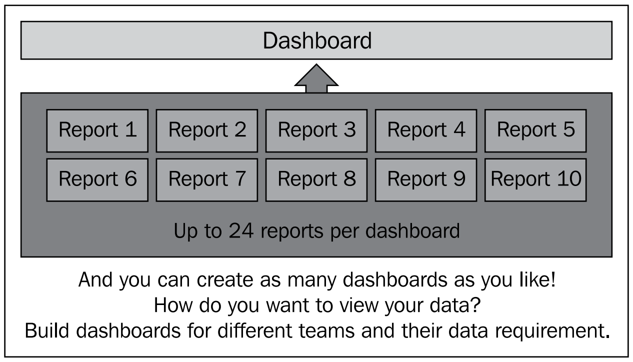
Working with dashboards

As with reports, the **Enhanced Reports** tab has made it easier to access your reports, dashboards, and their folders. From this tab, users can:

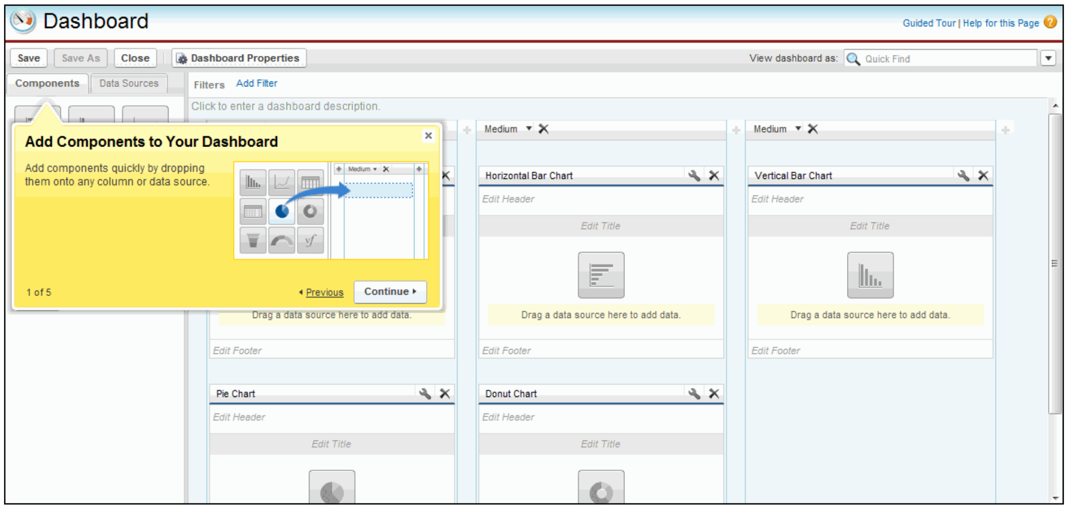
- **Create**: Users can create both reports and dashboards from the same location
- **Find and Use**: The default view displays the most recently used reports and dashboards, which can be changed to see everything and filtered to see just reports or dashboards, or use search or navigate by folder
- **Organize and Share**: Move reports and dashboards into the right folders and share them by providing appropriate (read-only or read/write) access to others
- **Manage**: Both the reports and dashboards can be edited and deleted from the same place, or export report data into an Excel or a CSV format
- **Schedule and Follow**: Users can follow their favorite reports and dashboards, and can also schedule refresh

When users click on **New Dashboard** under the **Enhanced Reports** tab, they land on the following page:




Dashboard
Report 1
Report 2
Report 3
Report 4
Report 5
Report 6
Report 7
Report 8
Report 9
Report 10
Up to 24 reports per dashboard
And you can create as many dashboards as you like!
How do you want to view your data?
Build dashboards for different teams and their data requirement.

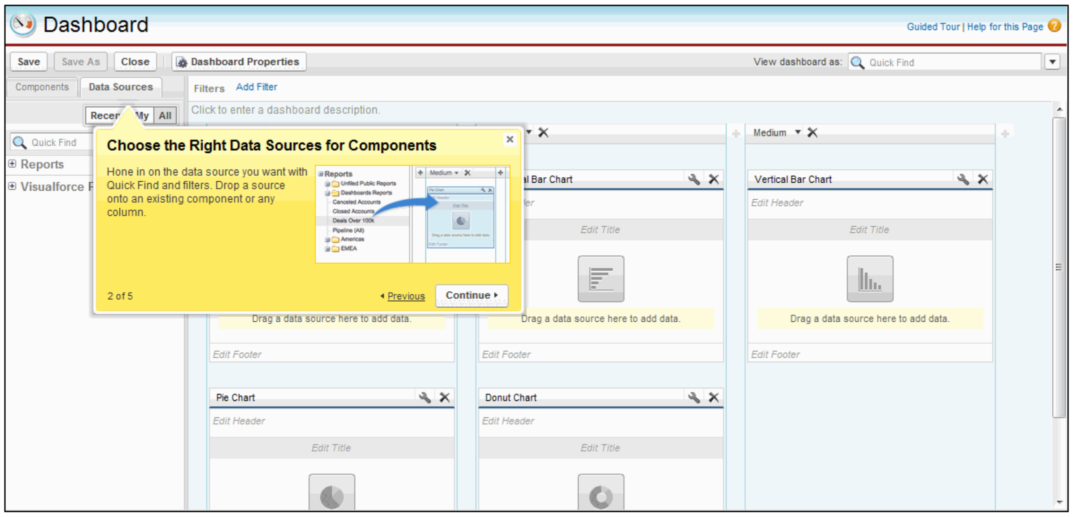
A quick guided tour gives you a jumpstart to using dashboards. Dashboard components such as charts, gauges, and so on, can be dragged-and-dropped onto various columns/zones or a data source:



Add Components to Your Dashboard

Add components quickly by dropping them onto any column or data source.

Data sources can be located with **Quick Find** and by using filters, and can be dropped onto existing components or various columns/zones:

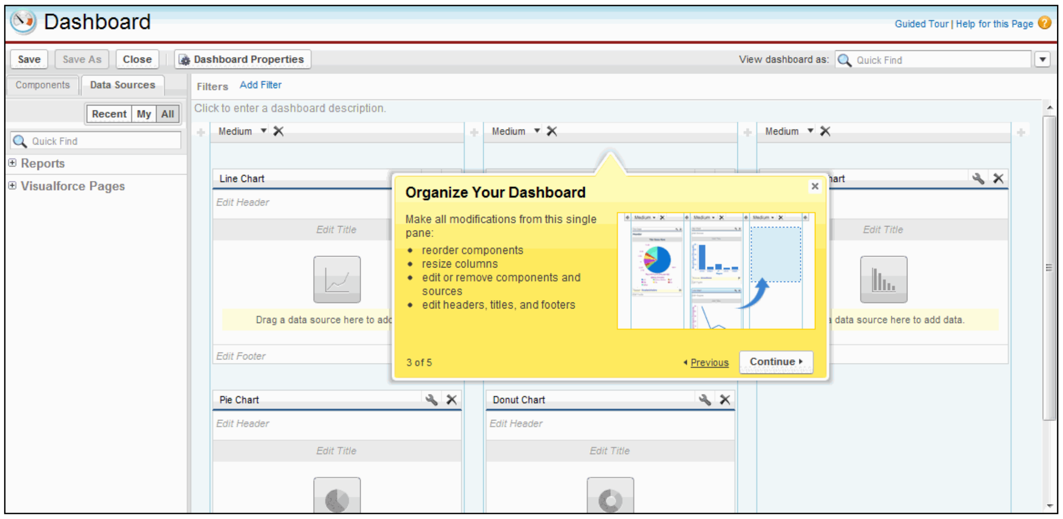


Click to enter a dashboard description.

Choose the Right Data Sources for Components

Hone in on the data source you want with Quick Find and filters. Drop a source onto an existing component or any column.

Organize dashboards by adding/removing components and data sources, reordering components, resizing and deleting columns, and editing headers, titles, and/or footers:



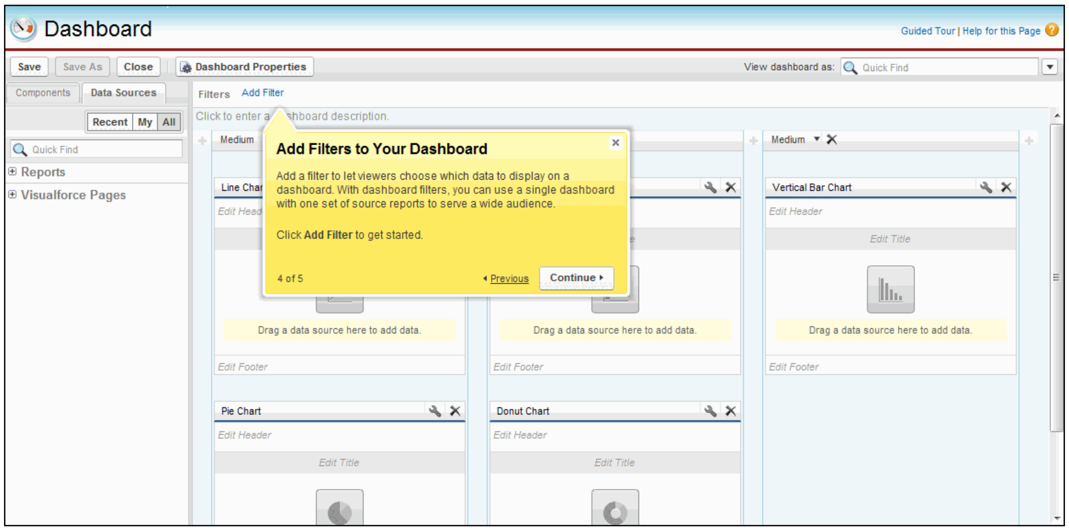
Click to enter a dashboard description.

Organize Your Dashboard

Make all modifications from this single pane:

- reorder components
- resize columns
- edit or remove components and sources
- edit headers, titles, and footers

You can add filters on a dashboard to let your users choose which data to display on a dashboard:



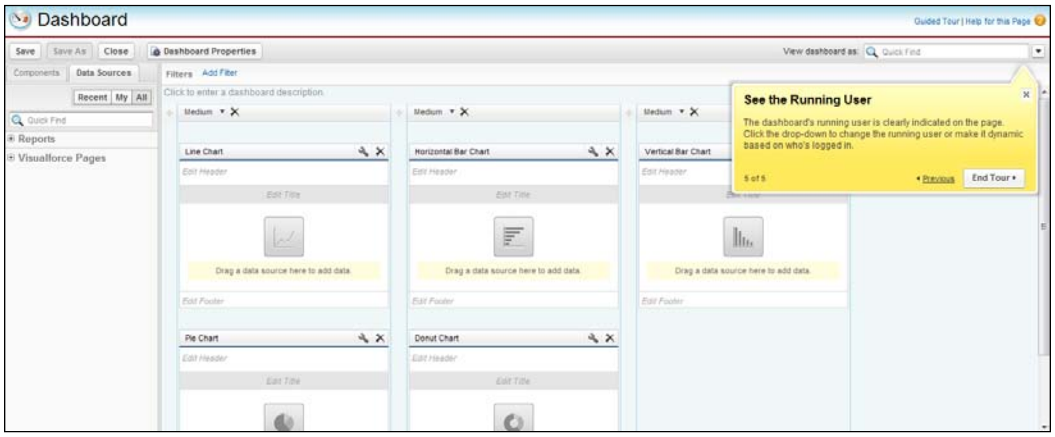
Add Filters to Your Dashboard

Add a filter to let viewers choose which data to display on a dashboard. With dashboard filters, you can use a single dashboard with one set of source reports to serve a wide audience.

Click **Add Filter** to get started.

4 of 5 ◂ Previous Continue ▸

You can view the running user of the dashboard in the field as indicated in the following screenshot:



See the Running User

The dashboard's running user is clearly indicated on the page. Click the drop-down to change the running user or make it dynamic based on who's logged in.

5 of 5 ◂ Previous End Tour ▸

Dashboard filters

End users can change the view on the dashboard by selecting filters on the dashboard using the drop-down menu. So a single dashboard with one set of source reports can be used to serve a wide audience. Without the dashboard filters, you'd have to create multiple dashboards, each with its own set of filtered reports. An example might be creating a single dashboard with key performance indicators such as *Closed Revenue*, and adding a filter on the **Product Name** field, so users can see the performance of a product.

Consider the following tips when using the dashboard filters:

- Each dashboard can have one filter
- You can create filters on fields that are common to all dashboard components or have equivalents
- You can't have filters on dynamic dashboards
- You can't add filters to dashboards containing a Visualforce component
- As of the Winter 12 release, filters can only be created for the picklist, lookup, and text type fields
- Scheduling or e-mailing a filtered dashboard produces unfiltered data

Dashboard data refresh

The data in the dashboard does not reflect the current state, but is always as current as the date and time displayed in the **As of….** field displayed in the top-right corner of the dashboard. You can either manually refresh the data or schedule it (Enterprise and Unlimited Edition only) to happen at a specified frequency. The refresh always happens in the background, so you can carry on with other tasks while the data is being replenished.

Consider the following when scheduling a dashboard refresh:

- An org can have up to 200 scheduled dashboard refreshes. Unlimited Edition users can schedule up to two dashboard refreshes an hour per day whereas Enterprise Edition can have only one.
- Just like reports, if you schedule a dashboard refresh on a specific date of every month, it refreshes only in those months which have that date, so if you have scheduled it for 31st of every month accidentally, it's logical to choose **Last** from the **On day of every month** drop-down list.
- Dashboards will not refresh if the running user does not have access to the dashboard folder.

- Refresh can't be scheduled for dynamic dashboards; they have to be refreshed manually.
- To send a dashboard refresh notification to other users, store them in public folders as other users can't see your personal folders.
- Users can click on the components in a dashboard refresh notification to view the underlying source report in Salesforce.
- Dashboard components including Visualforce pages cannot be displayed in the dashboard refresh notifications and have to be viewed in Salesforce.

Analytic snapshots

An inherent problem with the reports is that they always present the current state of the system and provide no idea of how the system was before the data got changed, and thus there is no scope for analyzing trends. With analytic snapshots, you get the facility to report on historical data, thus allowing users to report on data changes and trends in the org; for example, the number of open cases can be tracked by setting up an analytic snapshot.

They work by simply channeling the output of a report to a custom object, so authorized users can save the tabular or summary report results to a target custom object by providing an appropriate report to object field mapping. Further reports can then be created on the target object to identify trends.

So there is a source report that is scheduled to run and store results as records into a custom object, a target object that receives the results of the source reports as records, and a running user that determines the source report's level of access to data. This bypasses individual security settings, thus allowing users to see data that they may not be able to see otherwise.

Tips for analytic snapshots

Analytic snapshot is a less commonly used yet very powerful feature provided by Salesforce and offers the following benefits:

- Running reports faster by reporting on data that is already summarized
- Creating dashboards that refresh quickly by associating them with the pre-summarized data
- Sorting and filtering specific data summaries via list views
- Viewing trends in data via custom object records

The following sections cover various tips when using analytic snapshots.

Tips on source reports

Consider the following tips when setting up the source reports for analytics snapshots:

- A tabular report with its details hidden does not show up in the analytic snapshot source report selection. If an already-used source report's details are made hidden, the snapshot fails when it runs.
- You can include up to 100 fields in your source report.
- You can delete the schedule of an analytic snapshot. You can also delete the source report of a snapshot provided the report has first been removed from it by choosing some other report in the source report selection.

Tips on target objects

Consider the following tips when setting up the target objects for analytics snapshots:

- Field-level security can be used to make a target object's fields visible to appropriate users (Enterprise, Unlimited, and Developer Editions only).
- You can't delete a custom object, if it's being used by an analytic snapshot as a target object.
- You can include up to 100 fields on the target object and the field mapping availability is limited by the fields available on the target object.
- Target objects cannot contain validation rules or be included in a workflow.
- Analytic snapshots cannot contain target objects that trigger an Apex script to run when new records are created. If a trigger is added to an already existing target object, the snapshot fails when it runs.
- When an analytic snapshot runs, it can add up to 2000 new records to the target object. If the source report generates more than 2000 records, an error message is displayed for the additional records in the Row Failures related list.
- You must map at least one field from the source report to one field on the target object or data will not load from the source report to the target object when the analytic snapshot runs.

- You cannot map fields from the source report to the following fields on the target object:
    - Created By
    - Last Modified By
    - Created Date
    - Last Modified Date
- When you map fields from the source report to the target object, some data may lose its context when it is loaded to the target object. For example, if you map a date and time field from the source report to a text field on the target object, the date and time is loaded to the target object without the time zone.
- When executing an analytic snapshot, if the running user does not have read or write access to a mapped field in the target object, that field is dropped from the mapping, but does not cause the execution to fail.

Summary

Whew! That was a long chapter indeed and we have hopefully conveyed the basics, and tips, and tricks for the key analytics features that Salesforce has to offer. We covered report types which act as report templates, looked at how reports work and their key features, how reports are grouped and placed in folders and considerations and best practices when organizing them, dashboards, their key features and points to keep in mind when planning to create, manage, organize, and schedule them, looked at what analytics are, how they could be beneficial to us, and tips to consider when choosing their source and target objects.

5
Setting Up Development Environments

Force.com offers various different development environments depending on whether you are part of an IT team carrying out development activities for the production org used by your in-house business users, or you are an **independent software vendor** (**ISV**) working to create software for your customers and offer it as a service on the Force.com cloud platform. All these environments serve more or less the same purpose, that is, application development, but differ in their natures and capabilities and without proper guidance there can be confusion on the choice of development environment.

In this chapter we will try to demystify these problems:

- Define an environment
- Look at various development and test environments and their usages in different scenarios
- Provide tips on how to choose an appropriate development environment
- Migrating customizations between different environments, both manually and via Metadata API
- Review various application development management strategies where we take a look at how different combinations of environments fit in various development scenarios

Building apps on Salesforce

There are mainly two types of Salesforce customers, those who want to customize Salesforce and/or develop applications for their own business use, called **Customers**, and secondly those who want to build and distribute commercial applications for selling via AppExchange, called **ISVs**.

Salesforce caters to both types of customers and provides multiple environments for specific usage depending on the nature of the customer's business. For example, the Salesforce.com customers building in-house apps purchase the production org of a specific edition, depending on which they either get access to sandbox orgs or not. They then carry out development, integration, testing, **user acceptance testing** (**UAT**), and so on, mostly on sandboxes and sometimes on the DE orgs where sandboxes are not available.

ISVs intending to build commercial apps start developing on the DE/Partner DE orgs, create packages, and perform testing on the DE, Partner Test, or Sandbox orgs, and finally create released managed packages for production orgs end users.

The following sections cover various development environments that Salesforce.com has to offer and their benefits to help identify specific ways in which they can be leveraged in the application development lifecycle. These sections are explained generically, so the knowledge that you gain from them can be applied to both the in-house application development as well as the AppExchange product development scenarios.

What is an environment?

In Salesforce ecosystem, an environment is a synonym to an organization or an org in short, which is nothing but an instance of the Force.com cloud computing platform and infrastructure that allows developers and administrators to access, create, or deploy applications with edition-specific feature sets.

Environments have the following characteristics:

- Can be used for development, testing, and/or production
- Contain data and customizations
- Are edition based, and vary in features and limits
- All environments are accessible via a web browser, but some can also be accessed via API
- Not all advanced features such as multicurrency and territories are enabled by default; instead a request has to be made to Salesforce support to enable them on your org

Different environments can be used depending on your customer's type of business. For example, customers such as big corporates will typically need a production environment to run their business along with one or more sandboxes to allow parallel development without affecting the live data and users, whereas customers such as ISVs will need multiple development and test environments to build and test the product applications.

Production environment

Production environments are those that store live data and are used by customers to run their business. These environments can be based on various editions such as Group, Professional, Enterprise, or Unlimited Editions, if Salesforce CRM functionality is required. If CRM functionality is not needed, customers can sign up for Force.com Edition. For developing the applications for personal production use or for selling them commercially to customers with the Salesforce production environments, development environments are used. Note that Apex code editing is not allowed in production environments; however, declarative development, such as creating workflows, approval processes, or creating object validation rules can be directly done here.

Development environments

These are environments that are strictly used for development and testing purposes, where you can make changes without affecting end users on the production org. These are necessary for enterprise application development, so there may be multiple environments for various purposes such as development, integration, testing, training, and so on.

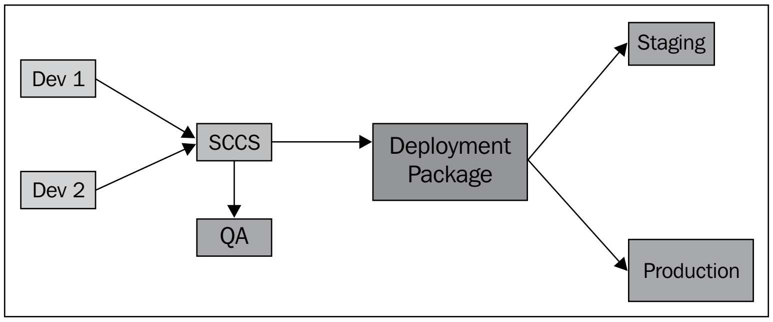
There are two kinds of development environments: sandbox organizations and Developer Edition organizations; these are covered in more detail in the following sections.

Sandbox orgs

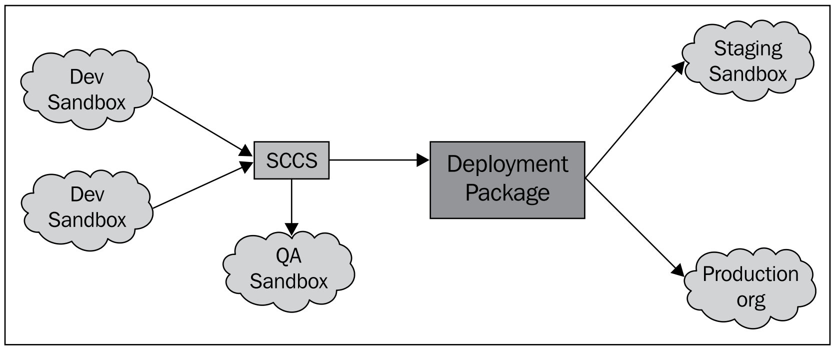
A sandbox is nearly an identical copy of your production environment available to Enterprise or Unlimited Edition customers. It contains the copy of the metadata of your production org and can include data, configurations, or both. Configurations include custom objects, fields, applications, workflows, and anything else that has been created to describe your organization and business processes.

From a developer's perspective, it is highly unlikely that any development happens directly in the production environment unless there are specific limitations such as unavailability of sandboxes in Professional and Group Editions. Accordingly, Salesforce has also provided corresponding environments that enable developers to follow a traditional development approach where they can collaborate with other developers to create or configure applications without affecting end users.

The following diagram illustrates the traditional development:



And here's how the cloud-based software development looks:



Multiple sandboxes can be created for various purposes such as development, integration, and testing, and all sandboxes are isolated from each other, just as they are from production environment.

Sandboxes are ideal if:

- You are a Salesforce customer with Enterprise, Unlimited, or Force.com Edition, which includes Sandbox
- You want to develop an application for in-house, non-commercial production use

- You want to test your beta managed packages, which can be installed on sandbox orgs, but not on production environments

Note that features that automatically send e-mail messages to contacts, customers, and users are disabled in sandboxes and cannot be enabled. For example:

- Case escalation
- Opportunity reminders
- Contract expiration warnings
- Data exports (using **Export Now** or **Schedule Export** from **Your Name | Setup | Data Management | Data Export**)

Salesforce offers three types of sandboxes, namely configuration-only, developer, and full. These are explained in the following sections.

Configuration-only sandbox

These contain a copy of all production org configurations such as dashboards, reports, products, price books, apps, and other customizations, but exclude object records, documents, and attachments. As these are just metadata-based orgs, it takes much less time to create or refresh them. These orgs offer up to 500 MB of data, so they can be used as development integration and QA environments, and can be refreshed once per day. These orgs contain the same number of user licenses as production and have API access enabled.

A customer may have up to a maximum of six configuration-only sandboxes. Unlimited Editions include five, whereas Enterprise Edition customers may procure them at additional cost.

Developer sandbox

These are special configuration-only sandboxes, which copy from production all customizations including application and configuration information but not the data, and are intended to be used by a single developer. The changes made by a developer can be isolated until they are ready to be shared in a bigger integration environment such as a configuration-only sandbox. The developer sandbox has a tighter data storage limit of 10 MB, which is most of the time enough for test data generated by a single developer. These orgs contain same number of user licenses as production and have API access enabled.

Unlimited Editions include 10 developer sandboxes and Enterprise includes one.

Full sandbox

A full sandbox is nearly an exact replica of your production org in terms of data and customization, so it can be used as a staging environment for UAT purposes. These orgs contain the same number of user licenses and data storage, and have API access enabled. A full copy sandbox can be refreshed once every 29 days. From the Summer'12 release, we can configure it not to copy data that is generally not useful in a sandbox.

Salesforce customers can have a maximum of three full sandboxes. Unlimited Edition includes one full sandbox and Enterprise Edition customers can purchase them at additional cost.

Typical uses of sandboxes

Salesforce has provided guidelines for using different kinds of sandboxes and identifying what development role they are best suited for. These are explained in the following table:

Use	**Developer sandbox**	**Configuration-only sandbox**	**Full copy sandbox**
Development	Best for app development	Good for app development	Giving data access to developers may be an issue Slower to copy
Testing	Unit tests Apex tests	Best for feature tests Load standard data for regression tests	Best for production debugging
Testing external integrations	Not a good fit	Special cases only Use sample or subset of data Works well if using external IDs	Best when external system expects full production data to be present
Staging or UAT	Not a good fit	Sometimes appropriate if testing against a subset of production data is acceptable	Best for validation of new apps against production configuration and data
Limitations	10 MB storage		

Tips for creating or refreshing a sandbox

Keep in mind the following considerations when creating or refreshing a sandbox:

- Sandbox refresh should be planned well ahead of time as it may turn out to be a long-running process, especially in case of full sandboxes. Sandbox refresh gets queued and is an asynchronous process, not starting immediately as soon as it is requested. Users are notified via e-mail when the refresh is complete.
- Sandbox refresh from production overwrites any changes made in the sandbox; for example, you have created new reports in your full copy sandbox and when you refresh it from production, the new reports disappear and you have to recreate them. So, it is recommended to take a back up of any work in progress and restore it after refresh is complete.
- Freeze all changes to production while sandbox is being refreshed; otherwise you may get inconsistent sandbox states

Developer Edition orgs

These are primarily used by ISVs where applications can be developed practically for free! (You can sign up for a free dev org at `http://www.developerforce.com/events/regular/registration.php`.)

- This is a free, fully-featured copy of the Enterprise Edition environment, with less storage and users.
- You can register this environment for free and start using it instantly. This environment comes with a number of pre-installed applications, such as Sales, Call Center, Marketing, and Idea, in case you intend to develop extensions to these.
- These environments can be used by developers and partners who want to build commercial applications and want their apps to be distributed via AppExchange.
- They can also be used whenever you are in need of a development environment, either for proof of concepts or for in house apps.
- Developer Editions provide full access to many exclusive features available to Enterprise and Unlimited Editions like API access is enabled so that you can develop integrations with your external systems and applications.
- Developer Editions offer 5 MB of data storage limit, 20 MB of file storage, and 5000 API requests per 24 hours.
- These continue to be in service as long as they had any activity within the previous six months.

Individual versus partner DE

A DE org usually refers to an Individual Developer Edition, which is available to anyone for free and is suited mostly for prototyping! However, as a Force.com ISV or consulting partner, you become entitled to a variety of environments for robust development and testing of Force.com apps. A partner de is a regular DE org, but with enhanced limits in terms of features, user licenses, data storage, and API limits, and hence it is also referred to as a super-sized DE org.

When to use individual DE org

Developer Editions are ideal when:

- You are a partner who intends to build a commercial Force.com app by creating managed package for distribution via AppExchange and/or Trialforce
- You are a Salesforce.com customer with Professional, Group, or Personal Edition and do not have access to a sandbox environment
- You want to explore Force.com for free

When to use partner DE org

Partner DE org is ideal when:

- You are working in a team and want a master environment to manage all the source code. In this scenario, individual developers can work in their own DE orgs and they can check their code in and out from this master repository environment.
- When you expect more than two developers to log in to develop and test.
- You need a bigger environment to allow more users to run robust tests against large data sets.

Tips and best practices

Things to keep in mind while choosing a development environment are discussed in the following sections.

Choosing a development environment

A sandbox environment can be used when:

- You have an Enterprise or Unlimited Edition
- You want to create functionality for a single production org

- You want to test the app on data that is similar to your production org
- You have special features, such as Person Accounts, enabled

A DE org can be used when:

- You want to develop a packaged app for commercial distribution via AppExchange
- All your sandboxes are exhausted
- Your development does not depend on the rest of the organization
- You have a Professional, Group, or Personal Edition, where sandboxes are not available to you

Development considerations

Keep in mind the following points when with working with development environments:

- Always develop in a development environment, either a DE org or sandbox, and then migrate it to production environment
- Plan well ahead whether or not you need a bigger environment for your development, as a regular DE org cannot be upgraded to a partner DE org
- Always take into consideration the features available in different editions while building applications; for example, Professional and Group Editions have certain features missing as compared to a DE org
- Always test your application before deployment in a separate and isolated environment so that development and test environments are separate

Testing environments

It is highly recommended that applications and other functionality must be thoroughly tested before deploying them to production or releasing them to customers. You should promote your changes to dedicated testing environments where you can perform integration tests with large data sets, for various security scenarios for multiple users, roles, and profiles, and stabilize the functionality by detecting issues earlier and fixing them before it goes live.

Dedicated test environments can be created in various ways. You can create a sandbox copy of your production environment where the application can be deployed directly from the development environment. This way you will not only mock the production deployment process, but also test the application in the production-like environment against real-life data.

If a sandbox is not available to you, you can use a DE org as your test environment. The standard DE org may be limiting due to license and storage limits, but you can have partner test environments based on various editions (Enterprise or Platform and Professional Editions) that have higher license and storage limits.

When to use a Partner Test Edition org

Use Partner Test Edition (Enterprise, Platform, or Professional Edition) when:

- You want to test your app in a production-like environment with more users and storage to execute real-life tests
- You want to develop a managed package to release commercially and want to test your beta managed package
- You want to make sure that your app will run smoothly on Enterprise, Force.com, or Professional Editions

When to use a sandbox org

Use a sandbox as a test environment when:

- You want to test your functionality against a copy of your production environment
- You have an Unlimited, Enterprise, or Force.com Edition environment with a sandbox
- You want to test a beta managed package

Various development scenarios

Now that we have gone through all the available environments, we can consider various development scenarios and try to fit them in environments for the development strategy.

Scenario 1

A developer wants to build a free Force.com app:

1. Sign up for a free DE org to be used as the development environment.
2. Build a Force.com app, using available platform features.
3. Sign up for another DE org to be used as the testing environment.
4. Use the Force.com IDE to develop and deploy your application from development to test to production environment.

Scenario 2

A customer wants to build a new Force.com app for a production environment:

1. Get a free DE org or set up a sandbox to be used as the development environment.
2. Build a Force.com app in the development environment.
3. Use another DE org or sandbox environment to test the app functionality.
4. Use Force.com IDE to develop and deploy your application from development to test to production.

Scenario 3

A partner wants to build a native Force.com app or a composite app that integrates with Force.com, for selling:

1. Sign up for a free DE org or partner DE org to be used as the development environment. Both these environments have API access enabled, which allows building the integration.
2. Use the Force.com IDE to build the app.
3. Package the app and upload it as a beta managed package.
4. Test the beta managed package in the sandbox, DE, and/or Partner Test environments.
5. Upload the package as released managed package and deploy it to the customers.

Migrating changes between environments

Migration refers to the movement of configuration changes from one Salesforce org to another. This is done either to keep orgs in sync or to move changes through dev orgs to production or the packaging org. Packaging org is the DE org that you use to create your final packages. For the sake of simplicity, we will refer to both the packaging org and production org as the production org.

Migration can happen either manually or via metadata.

Broadly, the steps to migrate changes are as follows:

1. Determine components—some may be migrated through metadata while others will be required to be moved manually.
2. Migrate components in the desired order maintaining dependency.
3. Optionally modify your Force.com project or outbound change set to deploy only a subset of components.
4. Deploy.

Migrating changes manually

Those components that are not available via API need to be migrated manually and this is achieved by performing setup changes through the Salesforce user interface. An example could be that of an approval process, which when developed in sandbox or DE org is not available in the Metadata API and the only way to migrate it on the destination org is to recreate it via the web interface.

The best way to manage manual migrations is to establish a change process on your production/packaging org and to track the changes that require manual migration.

Establishing a change process for production org

One of the strongest features of all the orgs is that all changes can be done directly on any environment via the web user interface, but this becomes a pain in the long run if you intend to develop enterprise-level apps, because due to this ease of development, there may be a situation that changes may be made directly on the production/packaging org whereas the application is being developed in the DE/sandbox org. To ensure foolproof development and successful deployment, it is essential that all changes should be present on development environments that exist on production, because while migrating from development to production you may overwrite changes made only and directly on production.

In a production/sandbox scenario, this can be made possible via regular sandbox refreshes but this may not be always possible as full sandbox refresh can take place only once every 29 days. In case of ISVs, doing development on DE orgs, migrating changes to QA, and packaging orgs should be tightly tracked as these are more like logical environments and not actually linked, unlike production/sandbox orgs. To streamline things, it is best to define a change process for your production org. A change process determines what kinds of modifications can take place on your production org, when can they occur, and who is responsible for making changes.

Best practices for change processes

As per Salesforce, the following points suggest some best practices for change processes, arranged from simplest to most complex:

- **Allow no changes on production**: This is simplest, but also the strictest and sacrifices immediate setup changes for easier deployment. This means that all development happens on sandbox/DE orgs.
- **Modify only components in the Metadata API**: Limiting changes to components that are accessible in the Metadata API will simplify change tracking, merging, and deployment.
- **Allow only one administrator to make setup changes**: This simplifies keeping track of changes on production and the administrator can replicate back those changes into development environments. This is more flexible approach that allows changes in production and project-based development at the same time. However, this is practical only when the organization is small enough that only one administrator can make all setup changes.
- **Schedule production changes**: If the production org requires more frequent updates, migrating those changes back to development environments can be scheduled.

Tracking changes

Changes that are made to components available in the Metadata API can be tracked and merged easily using desktop tools. When using Force.com IDE or Salesforce.com Migration Tool, you can put the metadata files in the version control system and can track changes by using the built-in functionality of **Source Code Control System** (**SCCS**). However, numerous changes are made to components that cannot be tracked automatically as they are not available in the metadata API. Such changes require manual tracking for both production and development environments.

Thus, it becomes essential to use tools and processes that help keeping a close track of changes. The possible options are as follows:

- Creating a shared Google spreadsheet to log and track changes, which may have the following format:
    - Who made the change
    - Org where the change was made
    - Date and time
    - Which component was changed

- Creating a custom application within Salesforce to record change requests and actual changes made. One such custom app is a free Change Control app on AppExchange.
- Creating a change control request form that is filled by both users and administrators for every enhancement requested and change performed.

Metadata migration

Components that are available in the Metadata API can be migrated by using the desktop tools or change sets.

Migration using change sets

Change sets allow sending customizations from one org to another. They can only contain modifications, such as apps, objects, reports, and so on, which you can make through the **Setup** menu and so they cannot be used to upload records. In other words, they contain metadata, not data. Organizations need to be connected before change sets can be migrated. The outbound change set contains customizations to be sent and the receiving organization sees it as the inbound change set. This can happen only between those orgs that are associated with a common production organization. Along with the connection, an administrator has to authorize each org for sending and receiving change sets.

1. Click on **Your Name** | **Setup** | **Deploy** | **Deployment Connections**.
2. Click on **Edit**.
3. Select **Allow Inbound Changes** and click on **Save**.

Best practices for using change sets

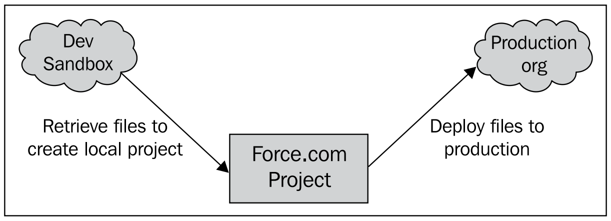
Salesforce suggests the following best practices for using change sets:

- **Deploy all dependent components**: All interdependent components should be included in the change set that doesn't exist on the target org, otherwise the deployment will fail. For example, to deploy a custom object and all of its fields, you must include the custom object and every field in it to the change set, otherwise the deployment will result in an empty custom object on the target org.
- **Add permissions and access settings to outbound change sets**: These allow administrators to migrate permissions for users so that they can access the new functionality.

- **Clone a change set to add dependent components to an uploaded change set**: Contents of an uploaded change set cannot be changed, so if you need to add any dependent components that accidentally got missed during the previous upload, clone the change set, add dependent components, and upload the new change set.
- **Validate change sets before deployment**: It is a good practice to do a test deployment of the inbound change set to view success or failures messages ahead of time and can take necessary steps to rectify errors, if any.
- **Change sets limited to 2500 components and 400 MB**: These limits apply for change sets and if they exceed them, you can create separate change sets, clubbing e-mail templates, dashboards, reports in one change set as these are components with minimum dependencies.
- **Deleting and renaming the components**: Change sets cannot be used for deleting or renaming the components. For this, you have to delete the component on target org first, and then upload the new component in the change set.

Migrating metadata files

Metadata API was designed to support traditional software development tools that operate on source files, text editors, diff/merge utilities, and version control systems, all of which require a local filesystem. So migrating changes from one org to another requires an intermediate tool that interacts with both environments using the Metadata API. Visually the migration process from sandbox to production looks like the following diagram:

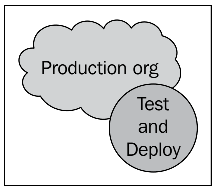

Dev Sandbox
Retrieve files to create local project
Force.com Project
Deploy files to production
Production org

Application lifecycle management

The flexibility of Salesforce.com to allow development in various environments may introduce complexity and lead to confusion as to what development strategy must be used to develop applications. Salesforce has discussed various development scenarios where the release process depends on the complexity of the application being developed, and accordingly there are different ways to manage different development projects.

Production development

Here, you carry out development directly on the production environment, using the powerful declarative web user interface. As development happens directly in production, there is no need for separate development and testing environment.



A typical application lifecycle includes the following steps:

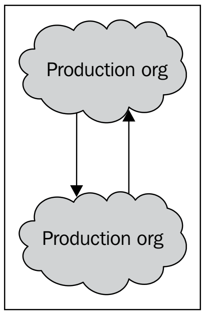
1. Plan functional requirements.
2. Develop them by using Salesforce Web tools.
3. Notify end users of changes.

Typical development projects include creating new or modifying existing:

- Custom fields
- Dashboards, reports, and e-mail templates
- Profiles or permission sets
- Visualforce pages

Developing with sandbox

For slightly more complex project or where isolation from production is required, a single sandbox org can be used. It can be used for both development and testing, and the changes are subsequently promoted to the production org. In this case, it is a good idea to keep track of changes, just like we discussed in the earlier section, to avoid overwriting production changes with sandbox changes.



A typical application lifecycle includes the following steps:

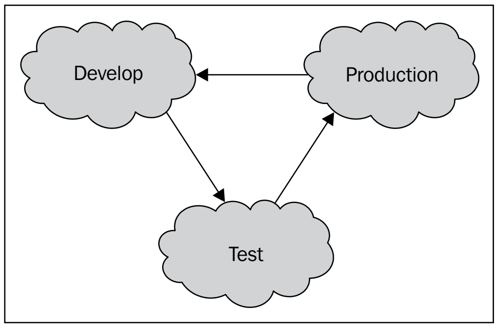
1. Create a development environment.
2. Develop it using Salesforce Web and local tools.
3. Test within the development environment.
4. Replicate production changes in development environment.
5. Schedule the release.

Typical development projects include:

- New custom tabs, applications, and objects
- Integrations with other systems
- Apps involving Apex, Visualforce, workflow, or new validation rules

Isolating development and testing

In a single environment scenario for development and testing, the development has to stop when testing is in progress and you can only resume development after you have deployed the changes to production. This results in inefficient usage of resources. In a more complex and sophisticated development model, development can continue while testing deployment is in progress. The changes, however, made in the production and test environment during testing will have to be brought back to the development environment.



A typical development lifecycle includes the following steps:

1. Create a dev environment.
2. Develop it using web and local tools.
3. Migrate changes from dev to the test environment.
4. Test them.

5. Replicate production changes in the dev environment.
6. Schedule the release.

Typical development projects include:

- New custom tabs, applications, and objects
- Integrations with other systems
- Apps involving Apex, Visualforce, workflow, or new validation rules

Multiple project development with integration, UAT, and staging

If there are multiple project development tracks that are scheduled to go live at the same time, you will certainly need an environment for carrying out the development integration to make sure code gets merged without any conflict. Then you will also need an environment for carrying out UAT to ensure that the original requirements are met. You may optionally need a staging environment where you can ensure that the deployment to production will go exactly as planned.

In this process, it is a good idea to restrict the changes being made only on the development environment as the complexity of the application has increased and so has the number of environments. It will become extremely difficult to keep environments in sync if changes are made in any environment other than the development environment. So a controlled change management process, which is repeatable over time for migrating changes to production, should be defined.

A typical development lifecycle includes the following steps:

1. Create dev environments.
2. Develop it by using Salesforce Web and local tools.
3. Create testing environments including integration and UAT.
4. Migrate changes from devlopment to the integration environment.
5. Test them.
6. Migrate changes from the integration environment to the UAT environment.
7. Perform UAT.
8. Migrate changes from UAT to the staging environment.
9. Replicate production changes in the staging environment.
10. Schedule the release.

Typical development projects include:

- Concurrent development of new applications in multiple environments
- Projects that require team development
- Apps involving Apex, Visualforce, workflow, or new validation rules

Developing enterprise applications

Large organizations usually end up with having complex development processes that span multiple release cycles. It is essential not just to separate out dev and test environments, but also to sync up projects on different release schedules. In this development scenario, you may have multiple development environments that integrate with each other before merging into a staging area. Additional environments could be added for purposes such as production support, training, and so on.

A typical development lifecycle includes the following steps:

1. Create dev environments.
2. Develop it using Salesforce Web and local tools.
3. Create testing environments including integration and UAT.
4. Migrate changes from devlopment to the integration environment.
5. Test them.
6. Migrate changes from the integration environment to roll up and UAT environments.
7. Perform UAT.
8. Migrate changes from UAT to the staging environment.
9. Replicate production changes in the staging environment.
10. Migrate changes to the training environment.
11. Schedule the release.

Typical development projects include:

- Multiple projects of various complexities and durations on different release schedules
- Development teams that include distributed developers and testers

Summary

By the end of this chapter, we hope that we have covered enough to explain various development and testing environments such as sandboxes, DE orgs, partner DE, and test orgs, different scenarios where they can be specifically leveraged, migrating changes between environments both manually and via Metadata API, and tips and best practices for using them. We discussed the application delivery strategies where combinations of various environments and development projects were suggested based on the complexity of the application development for the business.

6
Tools and Destinations that Every Force.com Developer Should Know

Tools are a developer's best friend, as they save time and increase productivity. As a Force.com developer, you're lucky to have lots of tools coming straight from the Force.com team, plus a lot of great tools are developed open source by the community too.

Tools discussed in this chapter will not be limited to IDE, but we will introduce you to various other tools. In this chapter we will cover:

- Schema Explorers, which are similar to database clients; for example, **Tool for Oracle Application Developers** (**TOAD**)
- Toolkits to easily integrate Force.com with other clouds for example, Google, Amazon, and so on
- Data migrators
- Libraries for various languages such as Java, .NET, Ruby, and so on

Destinations are various locations inside and outside the Force.com portals; they save time in troubleshooting problems or finding solutions and best practices. In this chapter, we will not limit the discussion to classic Force.com forums, looking at the following as well:

- Twitter
- Cookbook
- DeveloperForce wiki
- Open source projects

Tools for developers and admins

As a Force.com developer, you will find plenty of tools for various categories such as **integrated development environment** (**IDE**), schema browsers, data migrators, cloud integration libraries, and toolkits. These tools are continuously evolving and upgrading with every Force.com platform release; this applies to community-driven open source tools as well.

To stay updated with the latest on tools, you should keep an eye on the following:

- Official tools page from Salesforce at `http://wiki.developerforce.com/page/Tools`
- Open source projects on popular sites such as:
    - **Github**: Similar CVS and SVN, Git is comparatively newer, but is now the industry's most popular and preferred method for code versioning. Github provides free hosting for open source projects and is used by a lot of talented Force.com engineers and Developer-force team themselves.
    - **Code Share**: This is the official Force.com open source project directory. Developers upload their projects on various open source sites such as Google Code, Github, and so on, and link them up on Code Share for listing.

Choosing the right IDE

Force.com developers have a few options when it comes to IDE. These options range from desktop applications to browser-based web apps. This section will give you details and relevant pointers on the key IDEs available.

Force.com IDE based on Eclipse

This is a desktop application based on Eclipse (plugin). As it's based on Eclipse, you can use it on any operating system, that is, Windows, Mac, or Linux. The following are the salient features:

- Syntax coloring for Apex and Visualforce code, with some code assistance for sObjects and classes in the Apex editor
- It works connected to a Force.com org; that is, after saving a file, all your metadata (class, page, trigger, and so on) changes are reflected in the org
- One can run Apex unit tests here and see the code coverage for the test and execution results

- Schema Explorer can be used to fire SOQLs and explore structure of available sObjects
- One can deploy metadata to other orgs, using deployment wizards

For complete details about Force.com IDE, please visit the official wiki page at `http://wiki.developerforce.com/page/Force.com_IDE`.

Using Eclipse already?

It's quite possible that you already have an Eclipse installation that is in active use for developing projects in Java/J2EE, Android, and so on. This existing Eclipse installation can be used because Force.com IDE is also available as an Eclipse plugin, just like various other Eclipse plugins.

Installing Force.com IDE as a plugin is pretty easy; just add the update site `http://www.adnsandbox.com/tools/ide/install/` and follow the usual plugin installation steps. If you need detailed instructions about plugin installation, more specifically for your version of Eclipse, please check out the online guide available at `http://wiki.developerforce.com/page/Force.com_IDE_Installation`.

Developer console based on your browser

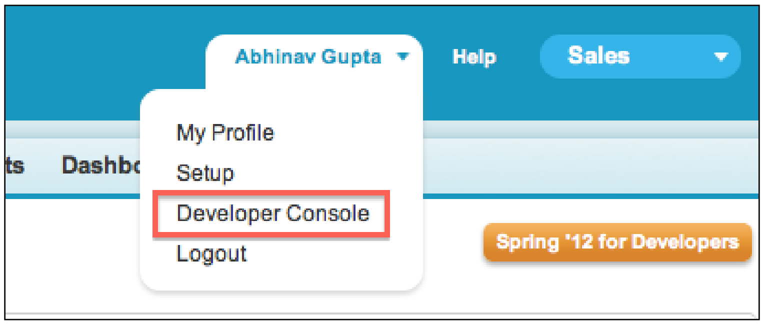
Developer console is a browser-based IDE, which means there are no software installations, it's cross platform, and developers can use it on a machine without a power user configuration. It allows developers to:

- Explore source repositories such as Apex classes, triggers, and Visualforce pages.
- Create or edit source files, with syntax coloring for both Apex and Visualforce code, plus auto completion or content assist for Visualforce tags.
- Debug application issues by viewing rich logs about application flow, which includes database events, workflow, callouts, validation logic, Apex method calls, and so on. Apart from that developers can set markers and view heap dumps for them.

No more System.debug(); use heap dumps instead!

If they are getting bugged by logical issues, such as `Attempt to de-reference a null object`, developers can nail down the buggy code by setting a heap dump capture marker at the specific line of code. On executing the process again, a request can be inspected at that specific line in the execution to get the full context of what is causing the error. Please note that developer console won't pause execution like a classic debugger of Java or .NET.

Developer console can be launched in a pop-up window by navigating through **Your Login Name** | **Developer Console**, as indicated in the following screenshot:



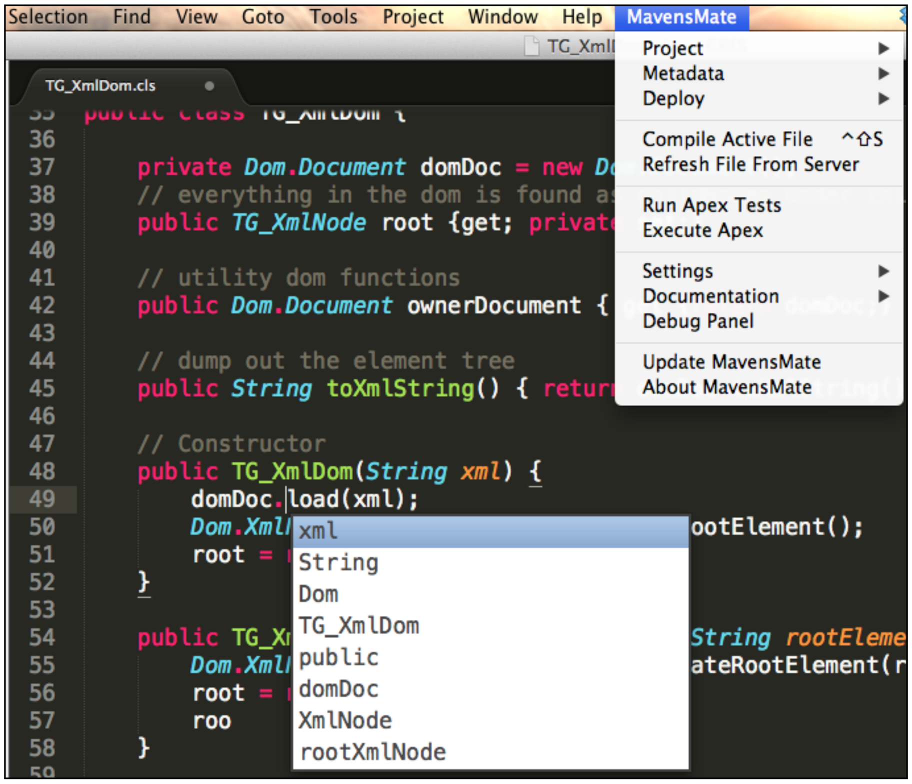
Users need the following permissions to make the most out of the developer console:

- `View All Data`: To use the developer console
- `Author Apex`: To use the **Execute anonymous text entry** box
- `Author Apex`: To save changes to Apex classes and triggers
- `Customize Application`: To save changes to the Visualforce pages and components

For more details about the developer console, please check the official documentation available at `https://help.salesforce.com/apex/HTViewHelpDoc?language=en&id=code_system_log.htm`. If login is required, please use your Developer Edition credentials.

Sublime Text plugin for Force.com

Just like Eclipse, if you are already using Sublime Text (`http://www.sublimetext.com/`) on your machine, you can use a community-developed open source plugin called MavensMate.



```
public class TG_XmlDom {

    private Dom.Document domDoc = new Do
    // everything in the dom is found as
    public TG_XmlNode root {get; privat

    // utility dom functions
    public Dom.Document ownerDocument {

    // dump out the element tree
    public String toXmlString() { retur

    // Constructor
    public TG_XmlDom(String xml) {
        domDoc.load(xml);
        Dom.XmlN                          ootElement();
        root = 
    }

    public TG_X                           String rootEleme
        Dom.XmlN                          ateRootElement(r
        root = 
        roo
    }
```

The following are the salient features of this bundle:

- Create Salesforce projects with specific package metadata
- SVN and Git support
- Create and compile Apex classes, Apex trigger, Visualforce pages, and Visualforce components
- Compile and retrieve other Salesforce metadata
- Run Apex test methods and visualize test successes/failures and coverage
- Supports code completion for sObject fields and Apex primitive methods (Alpha)
- Deploy metadata to Salesforce orgs

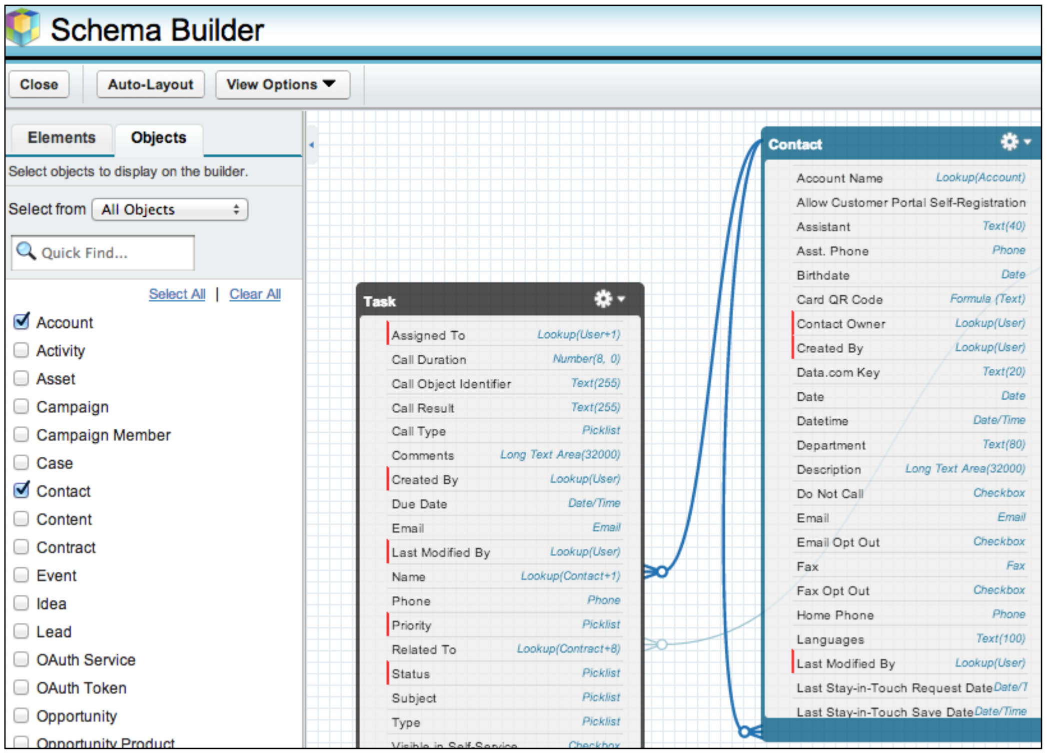
For detailed docs and installation instructions, please visit `https://github.com/joeferraro/MavensMate-SublimeText`.

Exploring objects, fields, and relationships using Schema Browsers

Every developer must have tried this previously, when working in Java, .NET, or PHP. Schema Browsers such as TOAD and other clients such as Squirrel are pretty popular. Similar tools are also available for browsing Force.com schemas. Almost all these tools allow browsing schemas such as subject and fields, with the ability to build and fire SOQL queries. Let's look at the popular browsers.

Schema Builder

Schema Builder is a browser-based tool, which is available as part of other setup screens in the org. It lets you visually view (as an entity relationship diagram), design, and explore your objects and relationships between them; it also lets you define new objects, fields, and relationships between them.



Select objects to display on the builder.

Select from: All Objects

- [✓ Account]
- Activity
- Asset
- Campaign
- Campaign Member
- Case
- [✓ Contact]
- Content
- Contract
- Event
- Idea
- Lead
- OAuth Service
- OAuth Token
- Opportunity
- Opportunity Product

Task

Assigned To	Lookup(User+1)
Call Duration	Number(8, 0)
Call Object Identifier	Text(255)
Call Result	Text(255)
Call Type	Picklist
Comments	Long Text Area(32000)
Created By	Lookup(User)
Due Date	Date/Time
Email	Email
Last Modified By	Lookup(User)
Name	Lookup(Contact+1)
Phone	Phone
Priority	Picklist
Related To	Lookup(Contract+8)
Status	Picklist
Subject	Picklist
Type	Picklist
Visible in Self-Service	Checkbox

Contact

Account Name	Lookup(Account)
Allow Customer Portal Self-Registration	
Assistant	Text(40)
Asst. Phone	Phone
Birthdate	Date
Card QR Code	Formula (Text)
Contact Owner	Lookup(User)
Created By	Lookup(User)
Data.com Key	Text(20)
Date	Date
Datetime	Date/Time
Department	Text(80)
Description	Long Text Area(32000)
Do Not Call	Checkbox
Email	Email
Email Opt Out	Checkbox
Fax	Fax
Fax Opt Out	Checkbox
Home Phone	Phone
Languages	Text(100)
Last Modified By	Lookup(User)
Last Stay-in-Touch Request Date	Date/T
Last Stay-in-Touch Save Date	Date/Time

One can access Schema Builder by using any of the following approaches:

- Navigate to **Your Name** | **Setup** | **Schema Builder**.
- Navigate to **Your Name** | **Setup** | **Create** | **Objects**. Then in the custom objects page, click on **Schema Builder**.
- Navigate to **Your Name** | **Setup**. In the **Quick Links** box on the Force.com home page, click on **Schema Builder**.

You can learn more about Schema Builder at `https://help.salesforce.com/htviewhelpdoc?id=schema_builder.htm&siteLang=en_US`.

Force.com Explorer

This is an Adobe AIR based application, and it's installable on any OS that supports Adobe AIR runtime. For fine details, please visit `http://wiki.developerforce.com/page/ForceExplorer`.

SoqlXplorer

This is a third-party (by SimonFell) tool similar to Force.com explorer for Mac OS, you can install it from `http://pocketsoap.com/osx/soqlx/`.

Data Loaders

Data Loaders help you to migrate sObject data to **CSV** (**comma-separated values**) files, spreadsheets, and relational databases. The following are the salient features of Data Loader:

- Provides both wizard-based GUI and command-line interface
- Supports batch operations and files with millions of rows
- Supports both standard and custom objects, with an easy drag-and-drop interface for mapping fields
- Fine logs about success and failure of operations

Different flavors of Data Loader are available for various operating systems, such as Windows, Linux, and Mac OS. Now we'll go through the complete list.

Data Loader for Windows

This version is developed and maintained by the Force.com tools team. It's compatible with Windows XP and Windows 7, and it supports all the features mentioned in the preceding section.

For more details, please check the following links:

- **DeveloperForce wiki page for Data Loader**: `http://wiki.developerforce.com/page/Data_Loader`
- **User guide**: `http://na1.salesforce.com/help/doc/en/salesforce_data_loader.pdf`

Force.com Excel Connector

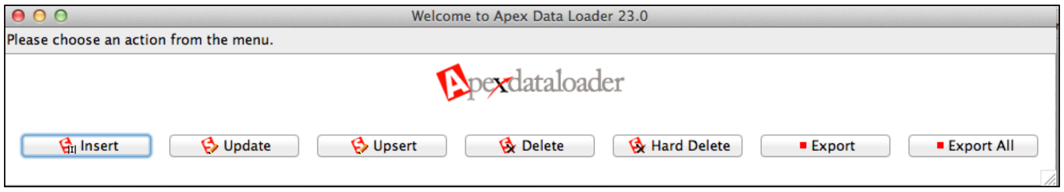
This is a Microsoft Excel add-in, useful for mass updating and cleaning up of Salesforce data from spreadsheets. This add-in is based on Force.com Office toolkit (`http://www.salesforce.com/us/developer/docs/officetoolkit/index.htm`) and uses Salesforce web services (SOAP APIs) internally.

This add-in is open source (`http://code.google.com/p/excel-connector/`) and written in **VBA** (**Visual Basic for Applications**). One can make most out of this plugin by either using as it is or by altering the source to best fit the business requirements.

For more details, please visit `http://wiki.developerforce.com/page/Force.com_Excel_Connector`.

LexiLoader for Mac OS

Official Data Loader application from Salesforce is not available for Mac OS, but a good third-party application (by Simon Fell) named LexiLoader is available. This app comes with a similar interface as of official Salesforce Data Loader, as shown in the following screenshot:


Welcome to Apex Data Loader 23.0
Please choose an action from the menu.
Apexdataloader
Insert
Update
Upsert
Delete
Hard Delete
Export
Export All

LexiLoader is compatible with Mac OS X 10.6 or later. For more details and download links, please visit `http://www.pocketsoap.com/osx/lexiloader/`.

Utility tools and apps for productivity boost

These apps are utilities such as browser plugins and password managers. Force.com community is pretty active in developing various open source tools and utilities. Let's take a look at a few popular apps.

Force.com migration tool

This tool lets you quickly push metadata components from one Force.com org to other, this tools save a lot of time under various scenarios:

- Pushing code across various dev, test, staging, and production orgs.
- Quick replication of standard org configuration for testing, to a brand new org. Doing the same will take a lot of effort from web interface.
- Setting up other automation tools such as CruiseControl or Hudson to call migration tool (ANT tasks) on a time trigger.
- Eclipse also supports metadata migrations, but as this tool can remember the components to be deployed, it's preferred approach for doing repeated deployments for the same metadata.

The online guide available at `http://www.salesforce.com/us/developer/docs/daas/index_Left.htm` has all the details to get you up to speed with this tool.

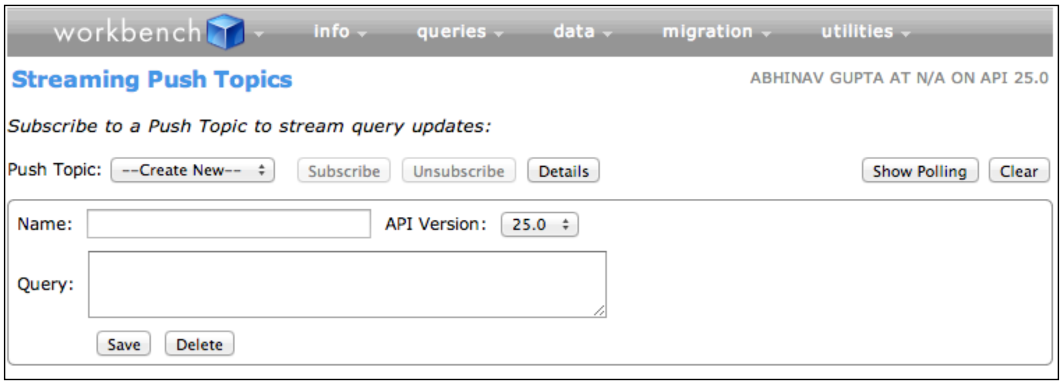
Learning from videos is usually faster. There is a good screencast video available at `http://wiki.developerforce.com/page/Migration_Tool_Guide` that explains the deployment process.

Salesforce Workbench

Workbench is a browser-based suite of utilities for admins and developers.
It includes support for:

- Working across different API versions of platforms
- Testing, troubleshooting, and exploring various Force.com APIs such as Force.com Partner, Bulk, Rest, Streaming, Metadata, and Apex
- Data and metadata migrations
- Testing single sign-on integrations

The following screenshot shows how you can add a Streaming API topic via Workbench:



Streaming Push Topics

ABHINAV GUPTA AT N/A ON API 25.0

Subscribe to a Push Topic to stream query updates:

Push Topic: --Create New--

Name:

API Version: 25.0

Query:

Workbench is an open source tool and is not supported by Salesforce—the tool to be used against production data. This tool is best utilized for development prototyping and research.

Learn more about Workbench at `https://workbench.developerforce.com/about.php`.

Force.com Security Code Scanner

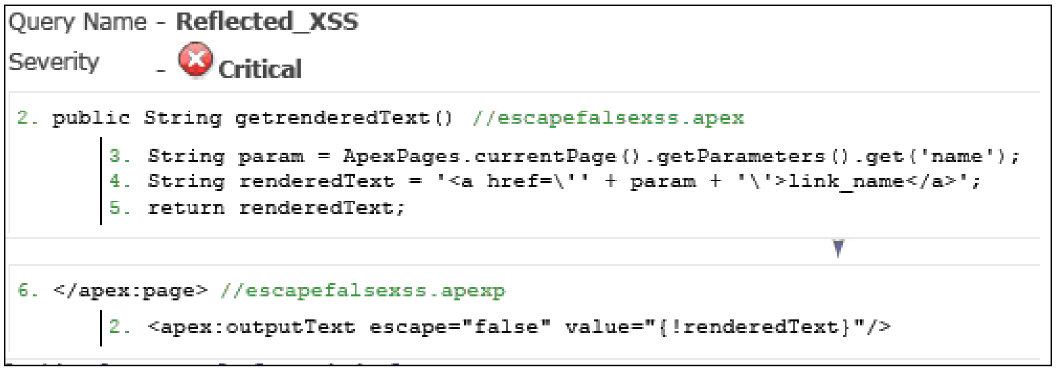
This tool scans your Apex and Visualforce code for any security vulnerabilities as per the various rules for security and code quality. This tool is very useful for anyone to keep hold on code from both quality and security standpoint. It's specifically useful when developing apps for listing on AppExchange, as it saves time by informing you about the issues that the Force.com security team will otherwise report.

Here are some key issues this tool can identify:

- **Security issues**: Cross site scripting, SOQL injection, SOSL injection, frame spoofing, and access control issues
- **Code quality issues**: DML statements inside loops, SOQL/SOSL inside loops, hardcoding `Trigger.new[0]`, queries with no `Where` clause or no `LIMIT` clause, not bulkifying Apex methods, Async (`@future`) methods inside loops, hardcoding IDs, multiple triggers on the same object, and static resource referencing
- **Other issues**: CRUD/FLS violations, open redirects, and hardcoded passwords

To scan your code via this tool, please submit it with the correct username on the website, `http://security.force.com/security/tools/forcecom/scanner`.

Once the scanning is done, the tool will e-mail the results in a PDF file to the e-mail address associated with the username provided on the website at the preceding link. The results point out errors very neatly, as indicated in the following screenshot:



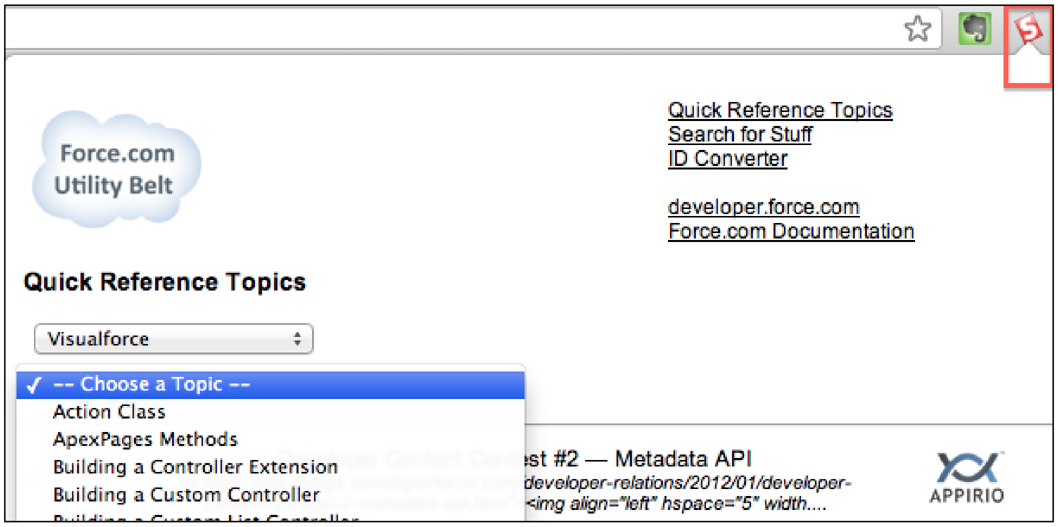
Query Name - **Reflected_XSS**

Severity - **Critical**

```
2. public String getrenderedText() //escapefalsexss.apex
        3. String param = ApexPages.currentPage().getParameters().get('name');
        4. String renderedText = '<a href=\'' + param + '\'>link_name</a>';
        5. return renderedText;

6. </apex:page> //escapefalsexss.apexp
        2. <apex:outputText escape="false" value="{!renderedText}"/>
```

Force.com Utility Belt

This is a handy browser plugin developed by Jeff Douglas (`https://twitter.com/jeffdonthemic`). Here is a screenshot of the plugin showing **Quick Reference Topics**:

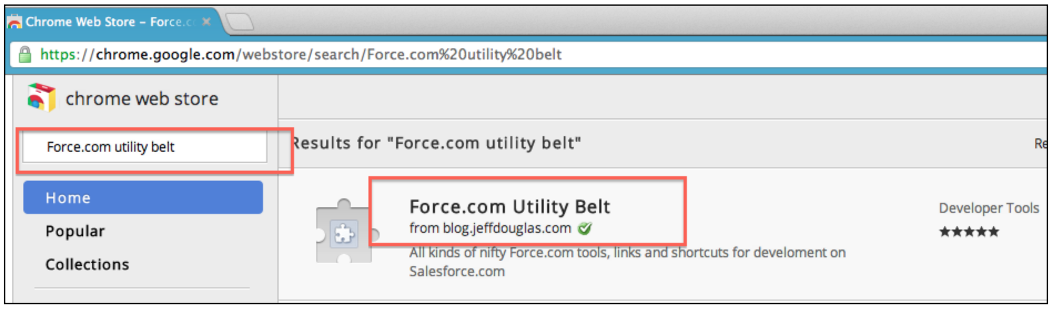


The following are the salient features of this plugin:

- **The Quick Reference Topics interface**: It gives quick pointers for Force.com documentation about Apex, Visualforce, SOQL, Ajax Toolkit, and Web Service API
- **Search for stuff**: Easily search for required piece in various Force.com developer guides, cookbook, AppExchange, Snipplr, Code Share, and so on
- **ID convertor**: For 15 digit to 18 digit conversions

In Google Chrome, install this plugin from `https://chrome.google.com/webstore/detail/bchgkjmjnmekbampjoenadmoekocpbhp`.

As the preceding URL is not easy to memorize, you can search for **Force.com utility belt** in **Chrome web store**. It's listed under the **Developer Tools** category, as shown in the following screenshot:

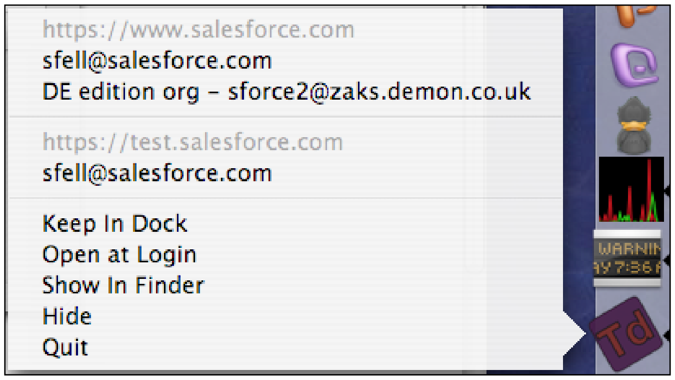

Chrome Web Store - Force.c
https://chrome.google.com/webstore/search/Force.com%20utility%20belt
chrome web store
Force.com utility belt
Results for "Force.com utility belt"
Home
Popular
Collections
Force.com Utility Belt
from blog.jeffdouglas.com
All kinds of nifty Force.com tools, links and shortcuts for develoment on Salesforce.com
Developer Tools
★★★★★

Trapdoor (Mac OS only)

It's pretty common for the Force.com developers to end up with lots of Force.com orgs and credentials—remembering each of them is a big pain.

Trapdoor can help you by giving a quick dock context menu for your sandbox, developer, and production org logins. Clicking on any one of the login usernames will directly log in you in the default browser for that username.

The following screenshot shows the Trapdoor dock context menu:


https://www.salesforce.com
sfell@salesforce.com
DE edition org – sforce2@zaks.demon.co.uk
https://test.salesforce.com
sfell@salesforce.com
Keep In Dock
Open at Login
Show In Finder
Hide
Quit

Trapdoor automatically scans all Salesforce login entries stored in the OS-provided keychain, so you will find most of your existing browser logins without any manual entry into the Trapdoor menu, plus you can add your new logins too.

For more details and installation instructions for Trapdoor, please visit `http://pocketsoap.com/osx/trapdoor/`.

Toolkits and libraries

Force.com is a platform with an open mindset; it offers a rich collection of toolkits to easily develop solutions across platforms and languages. These toolkits are libraries developed in Apex, Python, Java, and other languages, which give you decent integration and foundation code with code samples. This lets developers focus more on solving business problems, rather than developing the integration. Toolkits are available for:

- Cloud platforms such as Amazon, Facebook, Google, and Azure. For more details, please check `http://wiki.developerforce.com/page/Tools#Cloud_Integration_Tools`.
- Languages such as PHP, Java, .NET, Ruby, and Adobe AIR/Flex. For more details, please check `http://wiki.developerforce.com/page/Tools#Language_Integration_Libraries`.

Destinations

Quite often, when working with any language or platform you will need solutions for queries and problems, and the classic destination for getting those resolved is forums. With the Force.com platform, a developer is not only limited to rich suite of forums, but he can also use various other media sources such as wiki pages, cookbooks, and Code Share—interacting with community is even more social via twitter.

developer.force.com

This is the ultimate destination that each Force.com developer should know and visit often to keep up with latest happenings in the platform. It can be used for the following:

- **Technical Library**: You can find all the documentation related to platforms here. It lets you explore information by functional categories such as user interface, database, security, and so on.
- **Boards**: Rich suites of forums or discussion boards on topics such as Apex, Visualforce, Java, .NET, and many more.
- **Cookbook**: Yummy code samples and best practices for day-to-day or complex problems can be found here.
- **Blogs**: Follow these blogs to stay updated about upcoming events and updates to platform.
- **Events**: Get ideas about upcoming events such as webinars, code talks, and so on.

Open source goodness

One can learn a lot from what is already done by going through the various open source projects developed by experts from Force.com community. Fortunately, you will find amazing number of open source apps developed by community for native Force.com solutions, as well as in other languages such as Java, .NET, Objective – C, Ruby, and so on. It's not possible to cover each of them in this book, and you can find most of them at the following links:

- Github.com is the new destination for all open source actions. Follow these accounts for the latest updates:
    - `https://github.com/forcedotcom`: Official Force.com account on Github.
    - `https://github.com/ForceDotComLabs`: The account of the Force.com labs team that develops App Exchange apps and the associated open source code.

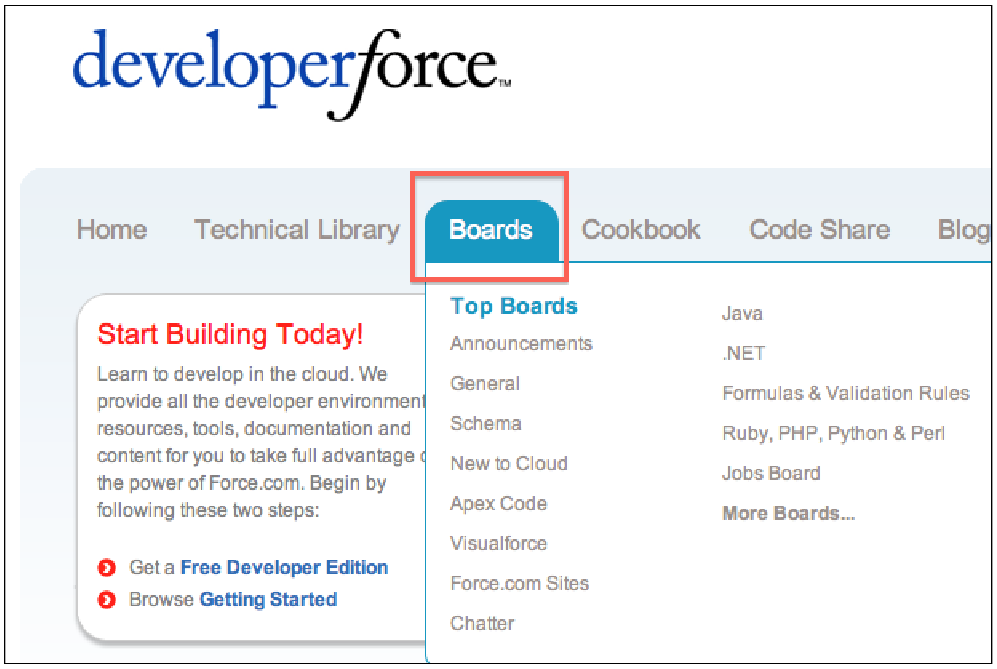
    ◦ `https://github.com/cloudspokes`: Cloudspokes.com is pretty popular for running challenges on cloud platforms such as Force.com, Google, Heroku, and so on. All the source code for submissions in these challenges is available here.

- Force.com Code Share is the official open source hub from Salesforce, but it's less active and popular as compared to Github now. For more details please visit `http://developer.force.com/codeshare`.

Queries and troubleshooting

This section explains approaches to troubleshooting problems during development. Here is the list of useful links:

- `http://developer.force.com`: As discussed in the preceding section, this portal is the official Force.com forum for troubleshooting and queries. Developers will find not only a huge community, but also that the Force.com team itself is out here. On landing at this portal, please go to the **Boards** tab as indicated in the following screenshot:



Start Building Today!

Learn to develop in the cloud. We provide all the developer environment resources, tools, documentation and content for you to take full advantage the power of Force.com. Begin by following these two steps:

- `http://stackoverflow.com`: This is a very popular online destination for troubleshooting problems about any language. There is a good chance that you will have already used Stackoverflow, if you have background in Java, PHP, and so on. The Force.com community is pretty active on this portal; for better responses ask questions with tags such as `salesforce`, `apex-code`, or `visualforce`.
- `twitter.com`: Use the hash tag `#askforce` with your question tweets. Many Force.com community members watch this hash and you might get responses sooner than on other forums.

Summary

This chapter covered a range of tools and destinations, starting from tools for coding, that is, IDE with both browser (Developer Console) and desktop (Eclipse/SublimeText) options, Schema Browsers and ERD tools, Data Loaders, and various other utility tools, such as Workbench and browser plugins, to make you more productive.

Knowing good destinations for code snippets, information, and troubleshooting is equally important. This chapter walked you through most the popular and key destinations such as `developer.force.com`, troubleshooting via discussion boards and social media, and learning from quality code in various open source channels such as Github and Code Share.

In *Chapter 7*, *Writing Better Apex Code*, you will learn a variety of tips and tricks about Apex language basics, designing triggers, getting the most out of SOQL/DML statements, smoother integrations using XML/JSON, writing secure code that respects governor limits, testing and debugging your code, and making AppExchange listing or production deployments easier.

7
Writing Better Apex Code

Apex is the world's first on-demand, strongly-typed programming language. It lets developers execute UI flow controls and backend transactions on the Force.com platform. Apex is usually not the first programming language for most developers, as they may have prior experience in languages such as C/C++, C#, Java, and so on, and together with this experience comes expectations from Apex. It's important to understand that Apex is different from other languages, mostly because of syntax, but mainly because it's meant to compile and run in a multitenant cloud platform. Because of this multitenancy, no one completely owns the server and everyone is free to write any ridiculously crappy code in Apex. There is always a *governor* sitting in Apex runtime that will stop the show in case the code tries to cross the limit. This means your code is given a runtime quota and it must comply with that. This quota has limits on various key areas in every flow, some of which are listed as follows:

- Number of code/script lines to be executed
- Number of **Salesforce Object Query Language** (**SOQL**) queries to be fired
- Heap memory used
- **Data manipulation language** (**DML**) statements issued

For more details on limits, please visit `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_gov_limits.htm`.

This chapter attempts to cover some key best practices, tips, and tricks to write better code in Apex and maintain good realtionship with the governor, that is, as a good tenant. Here is the outline of topics that will be covered in this chapter:

- Understanding the language basics such as:
    - Using the correct file nomenclature
    - What is different about static keywords in Apex?
    - Smart looping tricks
    - Using enums for constants
    - Reducing script statements
- Getting a hold on Advanced Apex practices such as:
    - Data, FLS, and CRUD security in Apex classes
    - Writing better SOQL queries and DML statements
    - Working on metadata information in Apex
    - Writing better Apex triggers
    - Handling XML and JSON responses in Apex
    - Considerations when packaging Apex code
    - Making the most out of API versions
    - Apex testing tips
    - RESTful web service tips
    - Understanding limits

Understanding the language basics

To write better Apex code, it's important to have the basics in place. In this section, the focus will be on using the correct nomenclature for Apex code files and language constructs (static, loops, constants, and so on).

Using the correct nomenclature for code files

Discipline in choosing file nomenclature is important in Apex, as all the Apex classes are stored at the same level and the programmer doesn't get multilevel nested packages/namespaces. If you are working on a moderately big project, it's not difficult to end up having more than 50 Apex classes. The situation is more complex when there are multiple developers; if no nomenclature is there, it's difficult to know which Apex class file is for what purpose, unless someone peeks into the code. Here are few tips:

- **Follow one standard for naming classes, methods, and variables**: Java standards are strongly recommended by the Force.com team. Use Camel Case (`http://en.wikipedia.org/wiki/Naming_convention_(programming)#Java`), for example, when:
    - Classes start with a capital letter, for example, `AccountTriggerHandler`
    - Methods start with a lowercase verb, for example, `loadAllContacts()`
    - Variable names should follow `lowerCamelCase`, for example, `Integer numberOfContacts`
    - Constants' names should be all uppercase with words separated by underscores, for example, `MAX_NO_CHILDS = 10`
- **Use a prefix or suffix describing the nature of the Apex class**: Here nature means if it's a Batch, Scheduled, Future, Trigger, and Page Controller/ Extension class. Having some form of prefix or suffix makes it easier to locate and know the purpose of the class without peeking into the code. For more details on this topic, please visit `http://www.tgerm.com/2011/11/apex-class-naming-convention-suggestion.html`. Here are a few Apex class names using prefixes based on purpose:
    - `trgr_ContactDuplicateCheck`: Class used with a Trigger on a Contact
    - `trgr_OpportunityAmountValidator`: Class used with a Trigger on an Opportunity
    - `page_OppLoadController`: Custom Controller class used with an Opportunity sObject
    - `page_ContactMergeExtension`: Custom Visualforce Extension class used with a Contact sObject
    - `ws_ContactService`: Apex class exposed as a SOAP web service
    - `btch_BulkRecordCleaner`: Batch Job Apex class for cleaning up records
    - `schd_NightlyAccountSync`: Scheduled job that runs on a nightly basis

- **Avoid too many class files**: Developers from a Java background are especially habitual in splitting functionality across multiple class files. This doesn't work well in Apex, and we strongly recommend keeping functionality limited to minimal Apex classes, because:
    - Too many files make it difficult to manage dependencies; we don't have options in Apex to know where in other Classes a given class, method, or variable is referenced
    - Having too many classes slows down an Apex flow, because it fires a validation routine on an invocation to make sure the dependencies are correct

Language constructs

Most of the languages have basic language constructs behaving similarly; it's slightly different with Apex, for example, with static variables. This section sets the correct foundation about the Apex basics.

Finding what's different with static keywords

Static in Apex is very different from other languages, typically in Java, a static variable once set persists until the lifetime of application, that is, until the app, server, or container is shut down or a new version of app is hot deployed. But in Apex static variables, the lifespan starts and ends with a request/flow because Apex classes are not cached in memory forever, they are loaded back in memory when a request comes via a page or trigger. So don't expect patterns such as Singleton (`http://en.wikipedia.org/wiki/Singleton_pattern`) to work correctly in Apex.

Static can be used for good in the following situations:

- Sharing information across classes in a flow, I know this is not a good design, but might be helpful in certain situations, typically when locking on some shared variables.
- Control triggers and avoid firing the same triggers again. Static persists across multiple trigger executions in the same flow, that is, if Object X trigger performs a DML operation on Object Y, the static variable value will be persisted for all changes during the course of the flow from `X > Y`. This can be used for chained triggers to fire indefinitely.
- A static variable never becomes part of a view state, so apart from constants some variables can also be declared as static.

Learn more about static at `http://www.tgerm.com/2010/09/visualforce-apex-static-instance.html`.

Simplifying the loops

All forms of looping, that is, do-while, while, and index-based for and for-each are available in Apex. For the integer array, for example:

```
Integer[] ants = new Integer[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

I frequently notice developers using traditional index-based iteration via for loops:

```
for(Integer idx = 0; idx < ants.size(); idx++) {
  System.debug('Iterating :' + ants[idx]);
}
```

This loop can be easily simplified to for-each as follows:

```
for(Integer ant : ants) {
  System.debug('Iterating :' + ant);
}
```

[For-each looping is recommended, unless you explicitly need an index for some calculations, it might avoid some runtime errors such as `Index Out of Bounds`.]

Making constants better with enums

Developers usually declare constants in the form of static final variables as follows:

```
public static final String WEATHER_SPRING = 'SPRING';
public static final String WEATHER_SUMMER = 'SUMMER;
public static final String WEATHER_WINTER = 'WINTER;
```

This approach is fine, but the constants are related in nature, that is, all of these are representing weather. So these are better candidates for enums. The same constants can be easily declared as the following enum:

```
public enum Weather {WINTER, SPRING, SUMMER}
```

[Declaring related constants as `enum` gives more comparison/type safety (for which enums are known).]

Reducing script statements

Apex code is always under strict control of a governor, so unlike other languages we can't run infinitely long code in Apex. As of now, the limit is 200,000 statements per flow; this means we can run Apex code spanning multiple classes, but the total number of executed script/code lines would be 200,000. This is a very important limit to understand and keep in mind when coding complex logic in Apex.

For example, if you are creating a map, as shown in the following code snippet. it will take four script lines:

```
Map <Integer, String >aMap = new Map <Integer, String>();
aMap.put(1 , 'a');
aMap.put(2 , 'b');
aMap.put(3 , 'c');
```

However, the same can be reduced to one script line as follows:

```
Map <Integer, String>aMap = new Map <Integer, String> {1 => 'a', 2 =>
'b', 3 => 'c'};
```

Similarly, one can create `Set` and `List` in the following manner. This type of instantiation is counted as a single script statement.

```
Set <String> aSet = new Set <String>{'a', 'b', 'c'};
List<String> aList = new List<String>{'a', 'b', 'c'};
// Creating Map directly from SOQL query
Map<ID, Contact>contacts = new Map<ID, Contact>([SELECT Id, LastName
FROM Contact]);
```

Keep an eye on the script lines of a code executed in loops; they are multiplied by the number of iterations.

The preceding code can easily be tested for script statement usage by executing it in the Developer Console and keeping an eye on the logs (logs are explained in the *Knowing the limits* section discussed later in this chapter).

Advanced Apex

This section will discuss advanced topics in Apex, such as SOQL, security, XML/JSON handling, web services, and so on.

Security data access via the with sharing keyword in classes

Apex runs in the system context, which bypasses security, that is, the current user's permissions, field-level security, and sharing rules. Apex gives the user options on enabling/disabling sharing rules. This can be done at the time of declaring the class, that is, the following Apex class respects the sharing rules, so the queries will only return the rows that the current logged-in user is entitled to see.

```
public with sharing class AccountExtension {
  public Account[] init() {
    // Apart from matching criteria, only those accounts visible to
    current user will be returned
    return [Select Id, Name from Account Where Name like '%corp'];
 }
}
```

If this code was written using the `without sharing` keyword, it would have returned all the Accounts. So this is a big security breach and very damaging to the enterprise customers.

Please note if a class is declared with none of the `with sharing` or `without sharing` keywords, it inherits the sharing settings of the calling class or the parent class (inheritance). For more details, please check an article from the Apex Developer's Guide available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_classes_keywords_sharing.htm`.

Enforcing CRUD and FLS

Apex has provision to support data security using only the "with sharing" or "without sharing" keywords as discussed in the preceding section, but there is no direct way to enforce sObject **create, read, update, and delete** (**CRUD**) and **field-level security** (**FLS**). Some of the Visualforce standard tags, such as `<apex:outputField />` or `<apex:inputField />` enforce FLS automatically.

But in general, it's a good idea to fail gracefully with missing permission errors, in case logged-in user's profile is not having the required permission. To enforce CRUD/FLS, developers can use Force.com ESAPI Library; this library gives handy functions to check CRUD/FLS permissions on an sObject.

To know more about ESAPI, please check `http://code.google.com/p/force-dot-com-esapi/`.

I strongly recommend you to read more about CRUD/FLS here: `http://wiki.developerforce.com/page/Enforcing_CRUD_and_FLS`

Writing better SOQL queries

This section shares tips about how to make the most out of SOQL queries and still stay within governor limits.

Query-required fields only

I usually notice developers use Eclipse Schema Explorer to build a query with all the fields in an sObject, for example:

```
Account[] accounts = [Select a.mycustomfield__c, a.Website, a.Type,
a.TickerSymbol, a.SystemModstamp, a.Site, a.Sic, a.ShippingStreet,
a.ShippingState, a.ShippingPostalCode, a.ShippingCountry,
a.ShippingCity, a.Rating, a.Phone, a.ParentId, a.Ownership, a.OwnerId,
a.NumberOfEmployees, a.Name, a.MasterRecordId, a.LastModifiedDate,
a.LastModifiedById, a.LastActivityDate, a.JigsawCompanyId,
a.Jigsaw, a.IsDeleted, a.IsCustomerPortal, a.Industry, a.Id, a.Fax,
a.Description, a.CreatedDate, a.CreatedById, a.BillingStreet,
a.BillingState, a.BillingPostalCode, a.BillingCountry, a.BillingCity,
a.AnnualRevenue, a.AccountNumber From Account a];
```

The preceding SOQL loads the fields that will never be referenced in the code in most cases, such as `mycustomfield__c`, `Ownership`, `SystemModstamp`, and so on. Loading too many unrequired fields will impact in many ways, such as:

- Heap size consumption will be high, typically if there are many records or unnecessary text area fields.
- Visualforce page will be slow, if the preceding Accounts collection is used on Page.
- Invalid dependency will be created to custom fields, such as `mycustomfield__c`. In case this field was mistakenly/accidentally created it would create problems on rename or deletion.

[A SOQL query should contain only the bare minimum fields required for the purpose; it's always better to change SOQL to add more fields if the requirements change.]

Using SOQL for loops

If you are expecting a large number of records from a SOQL query, it's a good idea to refactor that SOQL to work via SOQL for loops.

For example, the following is a standard SOQL query to load Accounts:

```
Account[] accounts = [Select Id, AccountNumber, Remarks__cFrom
Account];
for (Account acc: acocunts) {
…
…
}
```

Let's say this SOQL query returns 20,000 rows, and the `Remarks__c` custom field is a 255 character text area. This code could easily cross the maximum allowed heap size limit (6 MB); here is how:

Approximate memory required to hold 1 Account Row = ID (18 characters) + AccountNumber (40) + Remarks__c (255) = 300 characters

So for keeping 20,000 Account rows, we need almost 6 MB of heap. The same code can be refactored to use SOQL for loop, to make it work within the heap limits:

```
for (Account acc: [Select Id, AccountNumber, Remarks__cFrom Account])
{
…
}
```

[SOQL for loop loads data in chunks of 200 records. Check out this link to know more: `http://www.salesforce.com/us/developer/docs/apexcode/Content/langCon_apex_loops_for_SOQL.htm#soql_for_vs_soql`.]

Executing selective and index-based queries

SOQL should mostly be selective in nature should have filters on indexed fields. The following SOQL will fail if Account sObject has more than 50,000 records:

```
Select Id, Name from Account
```

This should be a rare use case where you need to process all Accounts in one go, if the required Batch Apex is better option for dealing with such data volumes.

Official Apex documentation states that a SOQL query with filters on indexed fields performs better. The following standard fields are auto indexed:

- Primary keys (ID, Name, and Owner)
- Foreign keys (lookup or master-detail relationships)
- Audit fields (Last Modified Date)
- Fields marked as External ID or Unique

If you are frequently filtering on a custom field, asking Salesforce support to create a custom index on them can lead to good performance. For more details, visit `http://www.salesforce.com/us/developer/docs/apexcode/Content/langCon_apex_SOQL_VLSQ.htm`.

Combining multiple SOQLs to avoid governor limits

The following code snippet costs two SOQL calls out of the total 100 available:

```
Account[] accs = [Select Id, Name from Account Where Name like 'a%'];
Contact[] contacts = [Select Id, Name from Contact Where AccountId in
:accs];
```

The same output can be achieved in one SOQL call:

```
Account[] accs = [Select Id, Name, (Select Id, Name from Contacts)
from Account Where Name like 'a%'];
for (Account acc: accs) {
    // get specific contacts to an account easily
    Contact[] contacts = acc.Contacts;
}
```

Combining SOQLs on multiple related objects reduces the risk of breaching governor limits, especially when too many SOQL queries are fired in a code flow.

Fixing SOQL that returns a single record

The following SOQL query loads a single record:

```
Account acc = [Select Id from Account where Name like '%Corp'];
```

The prceding query is prone to the following exceptions under certain conditions:

- `System.QueryException: List has no rows for assignment to sObject`: When there are no matching records for the criteria
- `System.QueryException: List has more than 1 row for assignment to sObject`: If there are multiple rows matching the criteria

The preceding problem can be handled gracefully in a couple of ways, depending on the requirement, such as:

- **Query as a list**:

  ```
  Account acc = null;
  Account[] accs = [Select Id from Account where Name like '%Corp'];
  if (accs != null && !accs.isEmpty()) {
   // note if more than 1 row is coming, that needs to be handled
  here.
   acc = accs[0];
  } else {
   System.debug ('No account found');
  }
  ```

- **Use try-catch block:**

  ```
  Account acc = null;
  try {
    acc = [Select Id from Account where Name like '%Corp'
    limit 1];
  }catch (System.QueryException qe) {
    // if record should exist throw an error here
    // if record is optional, do nothing in catch block
  }

  if (acc != null) {
    // rest of the logic that depends on acc
  } else {
    System.debug ('No account found');
  }
  ```

[Single row SOQL should be used carefully and tested under all possible criteria situations to avoid any runtime issues.]

Making the most out of dynamic SOQL

Dynamic SOQL lets the developer craft a SOQL string at runtime depending on the flow state and user-provided criteria. If you haven't used dynamic SOQL yet, learn more at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_dynamic_soql.htm`.

Variable binding in SOQL

One common gap in understanding that is common with most Apex developers is about variable binding not being supported in dynamic SOQL, which is not true. The following code snippet proves the point:

```
String name = '%Acme%';
String qryStr = 'SELECT Id FROM Account WHERE Name like :name' ;
Account[] accs = Database.query(qryStr);
```

This variable binding works like normal SOQL, and scans the binding variable in various scopes such as local variable, parameter, instance level attribute, and so on.

Taking care of errors and security

Developers should take extra care of the following points when using dynamic SOQL:

- Query syntax errors appear at runtime (no compile-time checks), if you are building a complex SOQL query based on lot of conditions. Make sure you test the generated query under all conditions, as code will break at runtime in case of any gap.
- Give a first try to a normal SOQL statement to solve the purpose. In case dynamic SOQL is the only way to a query then only it should be used.
- Incorrect use of dynamic SOQL can open your application to SOQL injection attacks. The following code snippet shows how:

```
<apex:page controller="AccountSearchController" >
    <apex:form>
        <apex:outputText value="Account Name" />
        <apex:inputText value="{!name}" />
        <apex:commandButton value="Query" action="{!query}" />
    </apex:form>
```

```
</apex:page>

public with sharing class AccountSearchController {
    public String name { get;set;}
    public Account[] queryResult{get;set;}

    public void query() {
        String qryString = 'SELECT Id FROM Account WHERE
        Name like \'%' + name + '%\'';
        queryResult = Database.query(qryString);
    }
}
```

The preceding code snippet leaves a hole for a hacker to tweak the SOQL query as desired. A better way to write the same code would be as follows:

```
public void query() {
String queryName = '%' + name + '%'
    queryResult = [SELECT Id FROM Account
    WHERE Name like :queryName];
}
```

Taking control on DML operations

This section shares some unknown tips about DML operations, such as transaction control, All or None behavior, and error handling.

Controlling transaction commits and rollbacks

Apex runtime transaction control works on flow or request level. Changes are committed to the database only if the complete flow succeeds, otherwise everything done is rolled back. Sometimes as part of a flow, a fine control over a bunch of DML statements is required. The blog post available at `http://blogs.developerforce.com/developer-relations/2010/05/tricky-transactions-on-forcecom.html` from Force.com blogs illustrates a good example and usage of savepoint.

For learning more about transaction control, please refer the article from the Apex Developer's Guide available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/langCon_apex_transaction_control.htm`.

Fine-tuning with extra DML options

The `Database.DMLOptions` instances can be passed on with DML operations to provide the extra configuration needed to fine-tune it. These extra configurations include:

- **All or None behavior**: This controls if an operation allows partial success. If it's true, all changes are rolled back, when any record causes errors.
- **Field truncation**: By default, DML fails if the value for a string is too large. This behavior can be tweaked by setting this property to `true`.
- **Assignment rule usage**: Tells DML on Lead and Case to use a specific assignment rule.
- **E-mailing behavior**: Controls if e-mail should be sent on various events during DML.

The following code snippets show how to use `DMLOptions`:

- **DML statement**:

```
Database.DMLOptions dmlOpts = new Database.DMLOptions();
dmlOpts.allowFieldTruncation = true;

Account acc = new Account(Name = 'Abc');
acc.setOptions(dmlOpts);

insert acc;
```

- **DML via database methods**:

```
Database.DMLOptions dmlOpts = new Database.DMLOptions();
dmlOpts.allowFieldTruncation = true;

Account acc = new Account(Name = 'Abc');
Database.insert(acc, dmlOpts);
```

To learn more about various other properties and usage samples of `DMLOptions`, please check the article from the Apex Developer's Guide available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_methods_system_database_dmloptions.htm`.

Error handling during DML operations

DML means statements such as insert, upsert, update, or delete. These statements are prone to failures because of various reasons. Some of these reasons are as follows:

- Fields marked as required are not populated
- Custom validation rules on objects
- Trigger logic for validation

If you are working with standard objects, give special attention to error handling, as the target org where your code will be deployed might have any of the preceding reasons. For example, the customer could create new validation rules.

The important thing here is to gracefully fail for such unknown error, and here is where error handling comes into the picture. Ideally, DML should be executed in a try-catch block with errors logged, handled, and reverted back to the user gracefully. Let's take the example of the following code:

```
insert new  Account(Name = 'XYZ corp');
```

This code will fail, if the website is marked as mandatory in some org. One way to handle that in Visualforce page is as follows:

```
try {
insert new  Account(Name = 'XYZ corp');
}catch(DmlException dmle) {
  ApexPages.addMessage(new ApexPages.message
  (ApexPages.severity.ERROR,'Failed to save Account,
   because of : ' + dmle.getMessage()));
}
```

Similarly, consider the following scenarios:

- **Trigger**: Trigger flow should be stopped and the operation should be rolled back or reverted back to the user as an error. The way to stop the trigger execution flow for a given record is to use the `Sobject.addError()` method. For example:

```
trigger AccountValidator on Account(before delete) {
  try {
  Account_Inovice__c[] childObjRecords =
  [Select Id from Account_Inovice__c Where AccountId__c IN:
  Trigger.newMap.keySet()];
  if (!childObjRecords.isEmpty()) delete childObjRecords;
  delete childObjRecords;
  } catch(DmlException dmle) {
    for (Account a : Trigger.new) {
      a.addError(dmle);
    }
  }
}
```

Learn more about trigger exception handling at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_triggers_context_variables.htm`.

- **Apex SOAP/REST webservice**: Create a Type or JSON/XML response indicating the error with required contextual information.

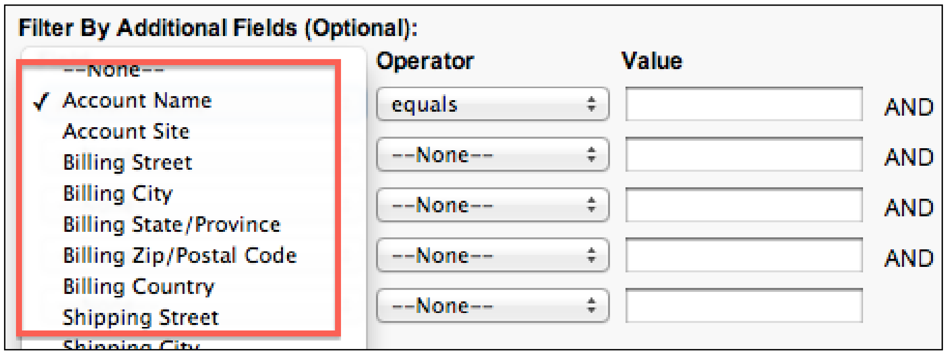
  Learn more about correct exception handling on the developer force wiki at `http://wiki.developerforce.com/page/An_Introduction_to_Exception_Handling`.

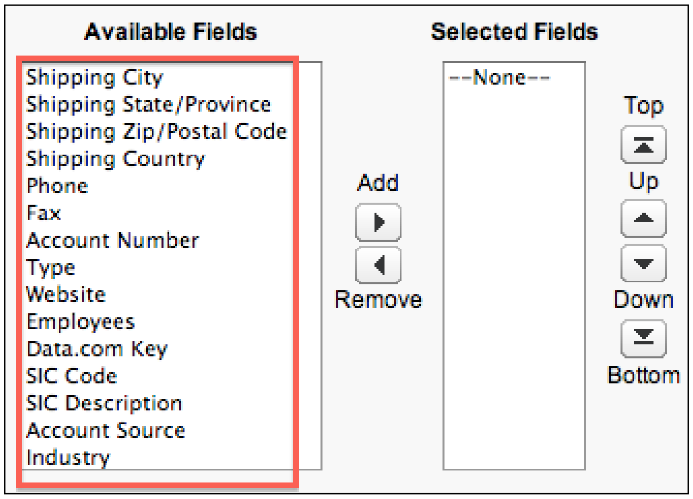
Decoding the Apex Describe Information

Apex offers a rich API to work on sQbject metadata and write extensible code. A few use cases for this are listed in the following sections.

Getting info about sObjects and associated fields

This is typically required in generic rule engines in Visualforce, which lets the user select which sObject they want to pick out of all available in the org and then create rules based on fields available in the same sObject. For example, creating a Visualforce page with the UI elements indicated in the following screenshots (something similar to view filters for sObjects):





The following code snippet shows how to generate such a field list for rendering in VF page DRO windows:

```
// Map of all objects in the org
Map<String, Schema.SObjectType> gd = Schema.getGlobalDescribe();
// if the type is known at compile time, one can use
// Schema.SObjectType.Account
Schema.SobjectType accType = gd.get('Account');
// alternate way to do the same if type is known
// Schema.SObjectType.Account.fields.getMap();
Map<String, Schema.SObjectField> fldMap = accType.getDescribe().
fields.getMap();
// selectoptions to show in dropdown on vf page
List<SelectOption> options = new List<SelectOption>();
for (Schema.SobjectField fld: fldMap.values()) {
    Schema.DescribeFieldResult fldDesc = fld.getDescribe();
    options.add(
        new SelectOption(fldDesc.getName(), fldDesc.getLabel()));
}
```

To learn more about describe information, please check the article available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_dynamic_describe_objects_understanding.htm`.

Accessing/updating the sObject records based on field names

In some scenarios, we have field names not known at compile time, so writing such a piece of code is not possible.

```
public void updateOpp(Opportunity opp, String fldName, Object val) {
  // update Opp if val is diff
  if (opp.fldName != val)
  opp.FldName = val;
}
```

To achieve this, one can make good use of the sObject system class methods, such as get() and put(), as shown in the following code snippet:

```
public void updateOpp(Opportunity opp, String fldName, Object val) {
  // update Opp if val is diff
  if (opp.get(fldName) != val)
    opp.put(fldName, val);
}
```

You can find more details on the sObject methods at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_methods_system_sobject.htm`.

Getting SobjectType from ID

This requirement required some decent code until the new `ID.getSObjectType()` method was introduced in the Winter '13 release. For code snippets and more details, please check the Apex document available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_methods_system_id.htm`.

Instantiating classes dynamically

Sometimes, the required class name is not known at compile time. This is typically required only in the code of framework- or extensions-based code, for example, a managed package that offers tax calculations on shipments, but wants to let your customers provide their own tax calculation logic. One possible way to achieve that using the Apex Type class is discussed here.

The following is the contract for tax calculation defined via the interface. This interface is part of the managed package.

```
global interface I_TaxCalculator {
  Decimal calculate (String itemSku, Integer quantity, String
  address);
}
```

The managed package ships with a default implementation of the contract. The default is used if no custom is provided by the customer. Here is the managed package code for this scenario:

- **Default tax calculator in managed package**:

  ```
  public class DefaultTaxCalc implements I_TaxCalculator {
         public Decimal calculate (String itemSku, Integer quantity,
  String address) {
      // some default tax calculations
          return 1;
      }
  }
  ```

- **Managed package logic for tax calculation**:

  ```
  // taxCalcClassName is passed after reading some config like
  // custom settings
  public static Decimal calculateTax(String taxCalcClassName, String
  itemSku, Integer quantity, String address) {
      I_TaxCalculator calc = null;
      if (taxCalcClassName == null) {
          calc = new DefaultTaxCalc();
      } else {
          Type typ = Type.forName(taxCalcClassName);
          calc = (I_TaxCalculator)typ.newInstance();
      }

      return calc.calculate(itemSku, quantity, address);
  }
  ```

Now in the target customer org, they can provide their own implementation. One example for this is as follows:

```
global class IndiaTaxCalc implements I_TaxCalculator {
  global Decimal calculate (String itemSku, Integer quantity,
   String address) {
    // some custom tax calculations
    return 0;
  }
}
```

Uploaded managed package code would be something like this (kept simplified for illustration):

```
TaxCalcConfig__c custSett = TaxCalcConfig__c.getInstance('Default');
// this could be 'IndiaTaxCalc'
String taxCalcClassName = custSett.Class_Name__c;
Decimal tax = TypeForName.calculateTax(taxCalcClassName, 'SKU-234', 2,
'New Delhi, India');
```

To learn more about the Type methods, please check the Apex document available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_methods_system_type.htm`.

Writing better triggers

It's pretty easy to get it wrong when starting up with Apex triggers. The key is to understand handling bulk data, re-using SQOL calls, and understanding the flow of events with triggers.

Understanding the order of execution for triggers

Apart from triggers, a couple of events happen on DML, which include validation rules, workflows, approvals, assignment rules, and so on. Knowledge of this order and impact of all events on each other is must to understand for writing good trigger code. It's strongly recommended to go through the guide from the Apex documents available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_triggers_order_of_execution.htm`.

Writing triggers to handle bulk data

It's easy to write triggers assuming only one record will be available in context variables such as `Trigger.new` or `Trigger.old`. This practice should never be followed, because of the following factors:

- A Visualforce page lets you perform DML on multiple records at the same time
- The same sObject might be exposed as a web service (SOAP/REST) that takes multiple records as input for DML operations
- If a data loader is used to populate the sObject

So if you have written some code such as the following, with hardcoded references to indexes, you are for sure in trouble.

```
trigger AccountTrigger on Account (after insert) {
    Contact[] cs = [SELECT Id FROM Contact WHERE AccountId =
    Trigger.new[0].id];
    …
}
```

Some recommended reading on this topic:

- **Trigger and bulk request best practices**: `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_triggers_bestpract.htm`
- **Using maps and sets in bulk triggers**: `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_triggers_bulk_idioms.htm#trigger_map_sets`

Multiple triggers on an sObject

It's quite easy to end up creating multiple trigger files (on various DML events) on the same sObject. This practice is still manageable if the DML events are different, that is, no two trigger files are on the same event. But it could be difficult to handle optimizations, ordering of events and flow, if events are the same, that is, three triggers on Account's `before insert` event.

Having a single trigger file per sObject is good practice, because:

- SOQL queries and DML on the same or related events can be executed at once, for example, if all three `after update` Account triggers are loading related Contacts, this could be easily done in one SOQL query, if a single Trigger file is used.
- If there are multiple triggers on same event, there is no guarantee in which order these events will fire. Using a single trigger file gives good flow control.
- It avoids recursion on triggers.

For the same cause, various trigger templates are suggested to keep a single trigger file across the same sObject; here are the references:

- **Gokubi's template**: `http://gokubi.com/archives/two-interesting-ways-to-architect-apex-triggers`
- **Mike Leach's template**: `http://www.embracingthecloud.com/2010/07/08/ASimpleTriggerTemplateForSalesforce.aspx`
- **My template based on the builder pattern**: `http://www.tgerm.com/2012/01/salesforce-apex-trigger-template.html`

Handling XML in Apex

There are a few options available in Apex for XML handling. The following table discusses all of these options. You need to make choices based on the pros/cons listed for each. Please note that these options are available in two flavors, System and Apex classes. System classes are provided by Apex runtime and not written in Apex language.

Option	Details	Pros	Cons
`XMLStreamReader` and `XMLStreamWriter`	System classes based on Java StAX is the oldest available XML parsing option. More details are available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_classes_xml_XmlStream.htm`.	Good for handling large XML files as it's a token-based streaming parser, so it consumes a lesser heap in parsing big XMLs.	Too many lines of complex code is required for parsing an XML.
`XmlDom`	This is an Apex DOM-based XML library developed by the Force.com team. This library is a wrapper on top of the `XMLStream` classes. More details are available at `http://developer.force.com/projectpage?id=a0630000002ahp5AAA`.	Gives a simple DOM interface to complex Apex XML Stream API.	As it is written in Apex, it consumes too many script statements if you are parsing moderately big XML. This library is deprecated now, so it's highly recommended to use the `Dom.Document` class discussed next.

Option	**Details**	**Pros**	**Cons**
DOM classes	These are the system classes that provide native DOM functionality via two core classes, `Dom.Document` and `Dom.XmlNode`. More details are available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_classes_xml_dom.htm`.	The best XML handling library available so far in Apex. It gives a simple DOM access to both read/write XML.	This library is a little different from other DOM libraries such as the W3C DOM model, so you might find it a little difficult to learn and use. More details are available at `http://www.tgerm.com/2010/02/apex-dom-document-xmlnode-bad-design.html`.
Fast XML DOM	These are the Apex classes that wrap the system DOM classes to expose a well known W3C DOM model. More details are available at `http://developer.force.com/projectpage?id=a0630000062Z2kAAE`.	Ensures that the developer is quickly productive via his existing knowledge of W3C DOM parsing APIs such as `getElementsByTagName()`. This library also simplifies the namespace handling in XML.	None as of now, getting good community feedback on Fast XML DOM.

Handling JSON in Apex

The options available in Apex for JSON handling are listed in the following table:

Option	Details	Pros	Cons
`JSONObject`	This is an open source Apex class that was the only option to handle JSON until Apex came with the native system classes described later in this chapter. More details are available at `http://code.google.com/p/apex-library/source/browse/trunk/JSONObject/src/classes/JSONObject.cls`.	This was the only option to parse JSON a few releases back.	Consumes lot of script statements, plus has some stability issues. It's deprecated now, because Apex has a native system class named `JSON` for this.
`JSON`	This is a system class that provides an API for easy JSON generation and parsing. More details are available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_methods_system_json.htm`.	This is the best API to use for JSON handling, gives support for direct serialization/deserialization of Apex types. It supports handling loosely-typed or untyped JSON with the Summer '12 release.	None as of now.

Packaging considerations with Apex

These considerations apply when you are uploading a managed-released package. This form of packaging locks many items and this locking is irreversible in nature. Here is a list of considerable items:

- `global`: Use this keyword very rarely, as once a class or method is marked `global`, its visibility can't be reduced to `public` or `lesser` in next releases.

- **Interface and Virtual/Abstract classes**: Here is what the Apex developer guide says:

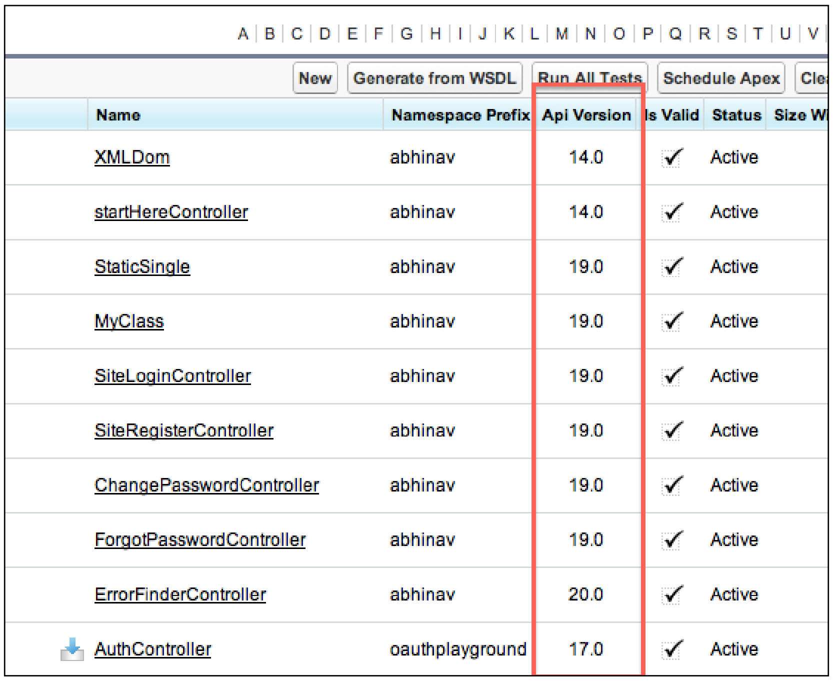
    You cannot add a method to an interface or an abstract method to a class after the interface or class has been uploaded in a Managed - Released package version. If the class in the Managed - Released package is virtual, the method that you can add to it must also be virtual and must have an implementation.

- `final`: This keyword can be pulled off, but can't be added to the global classes.

For more details and a better understanding about Apex and considerations in managed packages, please read the *Developing Apex in Managed Packages* article available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_manpkgs_dev.htm`.

API version

Each Apex class has an API version, which relates to the Force.com release features available. For example, if you want to use Winter '13 features, the API version should be at least 26. The following screenshot shows one way to check the API version (go to **Setup** | **Develop** | **Classes**).


A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V
New
Generate from WSDL
Run All Tests
Schedule Apex
Cle

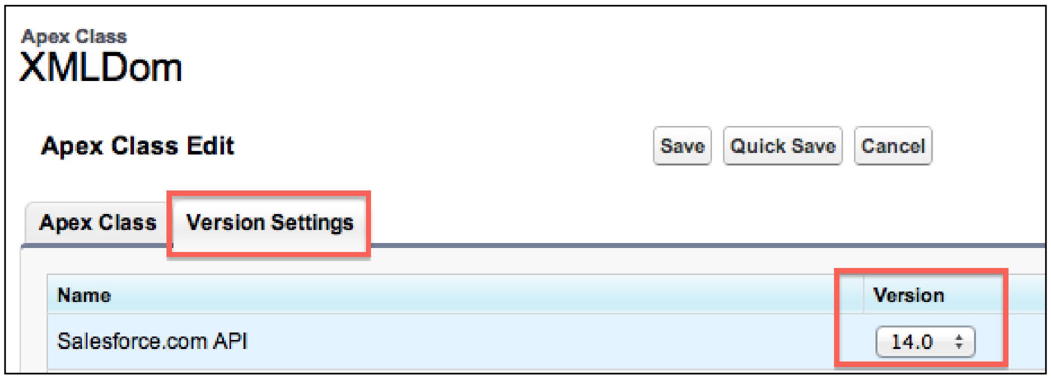
Name	Namespace Prefix	Api Version	Is Valid	Status	Size W
XMLDom	abhinav	14.0	✓	Active	
startHereController	abhinav	14.0	✓	Active	
StaticSingle	abhinav	19.0	✓	Active	
MyClass	abhinav	19.0	✓	Active	
SiteLoginController	abhinav	19.0	✓	Active	
SiteRegisterController	abhinav	19.0	✓	Active	
ChangePasswordController	abhinav	19.0	✓	Active	
ForgotPasswordController	abhinav	19.0	✓	Active	
ErrorFinderController	abhinav	20.0	✓	Active	
AuthController	oauthplayground	17.0	✓	Active	

Changing API versions

If you want to change the API version, you can do that in a couple of ways available to edit the version settings of a class.

Editing a class in your browser

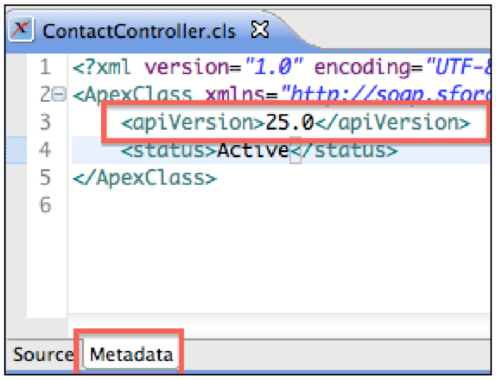
This applies if you have opened an Apex class for editing in your browser via **Setup | Develop | Apex Classes**. Click on the picklist under the **Version** column for the required change.



Name	Version
Salesforce.com API	14.0

Editing a class in Force.com IDE (Eclipse)

Changing the Apex class version is equally easy in Eclipse; just notice the **Metadata** tab beneath the opened class. Then you can change the **<apiVersion>** tag value to the desired API version and save.



```
<?xml version="1.0" encoding="UTF-8
<ApexClass xmlns="http://soap.sforc
    <apiVersion>25.0</apiVersion>
    <status>Active</status>
</ApexClass>
```

Important tips

If you are creating new Apex classes, it's highly recommended that you use the latest available API version. This will enable all the latest Force.com features in your Apex class.

If you are editing an old or existing Apex class, changing the Apex version, that is, upgrading the version, might be risky. Make sure you do a proper testing after such changes. One case study is available at `http://www.tgerm.com/2012/02/apex-integer-decimal-tostring-issue.html`.

Apex testing tips

This could be a vast topic for discussion, so the focus will be on key APIs and tricks to keep an eye on writing better test cases.

Isolating test data from org data

This means creating your own test data without relying on the presence of any records in the org. This helps to keep your test cases portable across orgs. With API version 24.0, pre-existing data in the org is not available to the test case. This includes all the records from standard, custom objects, and custom settings. However, in extremely odd situations this behavior can be turned on at both class and method level by using the `@isTest` notation. All you need to do is demarcate the test class or method with the `IsTest(SeeAllData=true)` annotation. For more details please visit `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_testing_data_access.htm`.

Testing with various profiles

By default, Apex runs in system mode, that is, the permissions and record sharing of the current user are not taken into account. Your application might have a couple of profiles with different data visibility, sharing, and CRUD/FLS behavior; it's recommended that the test case simulates this via `System.runAs()`. This method lets you run the test with security settings and permissions of a given use. For more details please visit `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_testing_tools_runas.htm`.

Letting your class know about the test context

Sometimes, it's important for Apex (trigger, batch, or Controller) code to know if it's executing normally or in test mode. This is especially important because some operations are not permitted in test mode, such as web service callouts. To handle such situations one can use the `Test.isRunningTest()` API for giving test cases mock data. More about this is explained at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_classes_restful_http_testing.htm`.

Apex REST web services

Apex comes with simple annotations such as `@RestResource`, `@HttpPost`, and `@HttpGet`, to expose any class as a RESTful web service. Here are some important considerations:

- A general rule of thumb is that the `with sharing` keyword should be used when declaring web service classes. In exceptional conditions, `without sharing` can be used for bypassing the security (sharing rules, and so on), but this should be done under extreme conditions, as doing this would breach security.
- Use Force.com ESAPI or Apex Describe Information to enforce the CRUD/FLS settings in Apex REST. They are again not enforced by default.
- Make good use of the `RestContext` class, it gives you handle to both inbound `RestRequest` and outbound `RestResponseobjects`.

API versioning with REST web services

We love Force.com API versioning and how they maintain backward compatibility in previous versions. We could easily achieve the same in Apex; let's say this is our first release of the REST service and Apex class:

```
@RestResource(urlMapping='/CoolestApexService/v1/*')
global with sharing class CoolestApexService
{
…
}
```

Now let's say in `v2` of the API, there are significant changes in the Apex code and thus API. We don't want to break any client code depending on `v1` of the API. We could easily achieve this by creating a copy of the preceding Apex class and doing major changes in the copy and maintaining the backward compatibility. For example:

```
@RestResource(urlMapping='/CoolestApexService/v2/*')
global with sharing class CoolestApexServiceV2
{
…
}
```

Please note that we renamed the class and rest annotation's URL mapping.

Knowing the limits

It's important at certain times to know the limits available in a given context. Here context means execution context; it could be different in the Visualforce page controller, trigger, anonymous blocks, and test cases. For example, the heap size limit is usually 6 MB, but in case of e-mail services it becomes 36 MB.

Apart from the total limits available, Apex runtime gives details of the limits consumed so far in the execution. One can use the `Limits` system class to know total and consumed limits on various factors such as:

- Script statements
- Heap size
- Number of SOQL queries

This Limits API could be used to fail gracefully under certain situations, that is, to show a clean error message instead of the Salesforce crash screen. Here is some example code that shows how to use Limits API to know available SOQL calls:

```
// Prints 100.
System.debug('Available Queries : ' + (Limits.getLimitQueries() -
Limits.getQueries()));
// Consume 1 Query limit out of typical quota of 100
Contact[] contacts = [Select Id from Contact];
// Consume 1 more Query limit out of typical quota of 100
Account[] accounts = [Select Id from Account];
// Prints 98 now.
System.debug('Available Queries : ' + (Limits.getLimitQueries() -
Limits.getQueries()));
```

To know more about the `Limits` system class and its methods, please refer to the Apex document available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_methods_system_limits.htm`.

The next section explains about debugging the limits in more detail.

Tracking resource (limit) usage

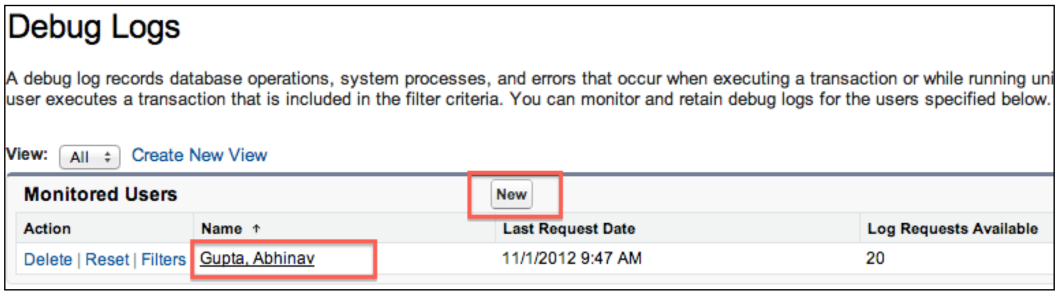
It's always good practice to know and keep an eye on how much of the quota of various platform resources/limits are consumed by the Apex code. Just like any other language such as Java, to track resource usage, one needs to check the debug logs. If you have never used debug logs before, it's recommended to go through the following articles first:

- *Understanding the Debug Log*: `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_debugging_debug_log.htm`
- *Setting Debug Log Filters*: `http://www.salesforce.com/us/developer/docs/apexcode/Content/code_setting_debug_log_levels.htm`

Debug logs can be checked at a couple of locations, which are discussed in the following sections.

Using Debug Logs in the Setup area

You can enable logging for a given user by navigating to *Your Name* | **Setup** | **Monitoring** | **Debug Logs**. Click on the **New** button and enable logging for a given user. The following screenshot indicates logging enabled for a user named `Abhinav Gupta`.



Debug Logs

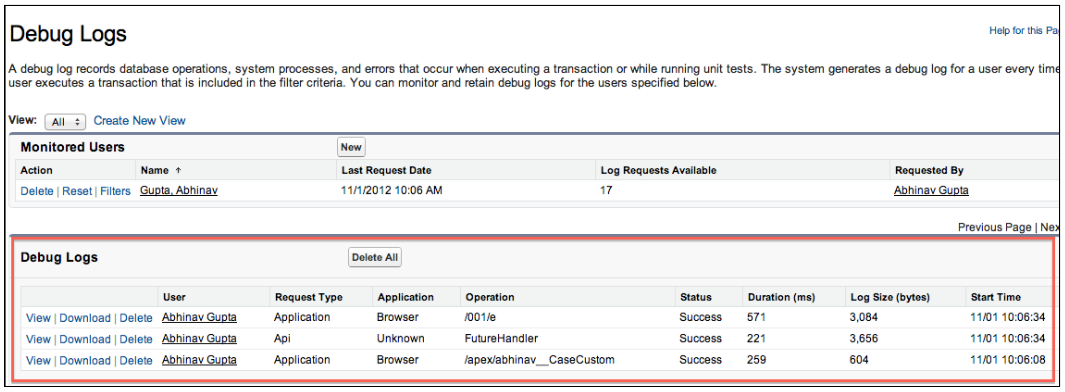
A debug log records database operations, system processes, and errors that occur when executing a transaction or while running uni
user executes a transaction that is included in the filter criteria. You can monitor and retain debug logs for the users specified below.

View: All Create New View

Monitored Users New

Action	Name ↑	Last Request Date	Log Requests Available
Delete | Reset | Filters	Gupta, Abhinav	11/1/2012 9:47 AM	20

Once the logging is enabled, all code executions (controller, trigger, batch, future, and so on) can be seen in the **Debug Logs** grid. Please refer to the following screenshot for more details:



Debug Logs

Help for this Pa

A debug log records database operations, system processes, and errors that occur when executing a transaction or while running unit tests. The system generates a debug log for a user every time user executes a transaction that is included in the filter criteria. You can monitor and retain debug logs for the users specified below.

View: All Create New View

Monitored Users New

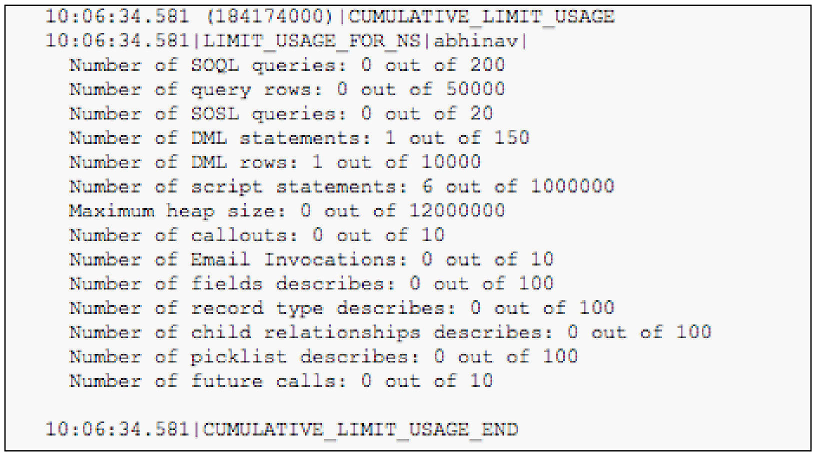
Action	Name ↑	Last Request Date	Log Requests Available	Requested By
Delete | Reset | Filters	Gupta, Abhinav	11/1/2012 10:06 AM	17	Abhinav Gupta

Previous Page | Nex

Debug Logs Delete All

	User	Request Type	Application	Operation	Status	Duration (ms)	Log Size (bytes)	Start Time
View | Download | Delete	Abhinav Gupta	Application	Browser	/001/e	Success	571	3,084	11/01 10:06:34
View | Download | Delete	Abhinav Gupta	Api	Unknown	FutureHandler	Success	221	3,656	11/01 10:06:34
View | Download | Delete	Abhinav Gupta	Application	Browser	/apex/abhinav__CaseCustom	Success	259	604	11/01 10:06:08

To get into details of resource usage, one can click on the **View** link on any log record. In the log details, the last section shows the cumulative usage for various platform resources, as indicated in the following screenshot:



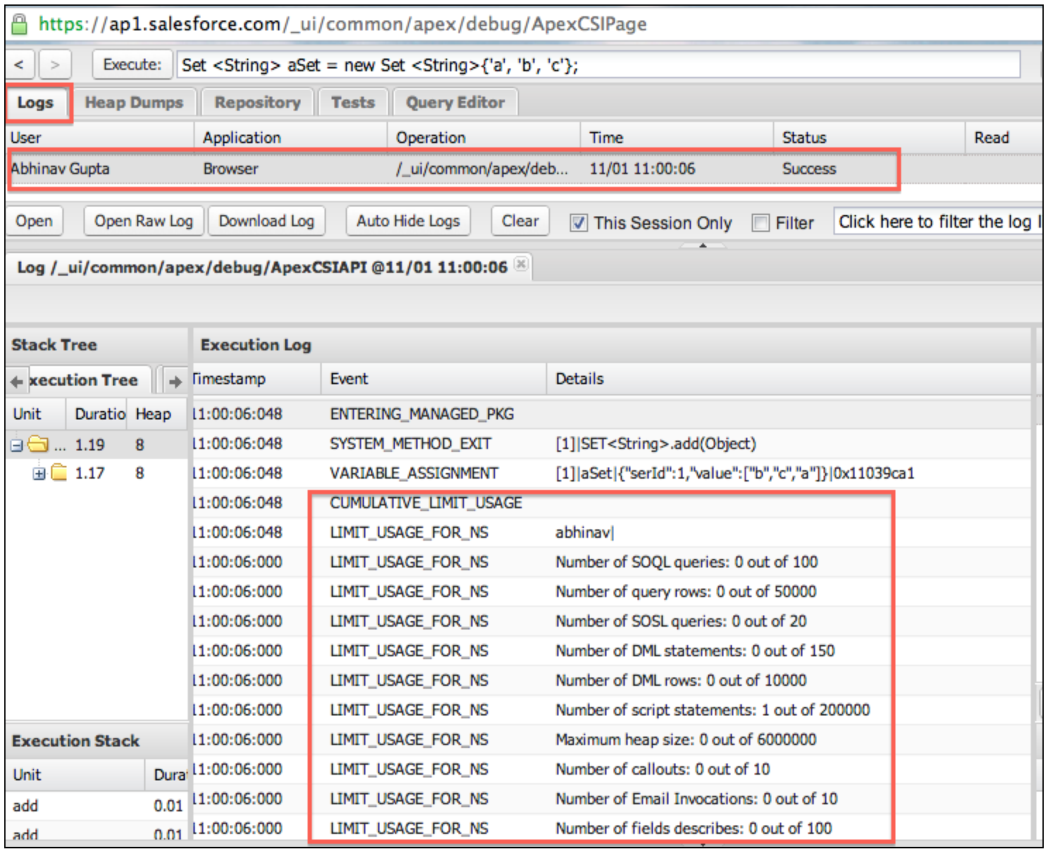
```
10:06:34.581 (184174000)|CUMULATIVE_LIMIT_USAGE
10:06:34.581|LIMIT_USAGE_FOR_NS|abhinav|
  Number of SOQL queries: 0 out of 200
  Number of query rows: 0 out of 50000
  Number of SOSL queries: 0 out of 20
  Number of DML statements: 1 out of 150
  Number of DML rows: 1 out of 10000
  Number of script statements: 6 out of 1000000
  Maximum heap size: 0 out of 12000000
  Number of callouts: 0 out of 10
  Number of Email Invocations: 0 out of 10
  Number of fields describes: 0 out of 100
  Number of record type describes: 0 out of 100
  Number of child relationships describes: 0 out of 100
  Number of picklist describes: 0 out of 100
  Number of future calls: 0 out of 10

10:06:34.581|CUMULATIVE_LIMIT_USAGE_END
```

Any resource usage approaching near limits is a good hint to tune the application before production crashes. It's advised to do the required refactoring or clean up in time, to keep a decent margin for resource limits.

For more details, please refer to the *Viewing Debug Logs* article from Salesforce docs. This article is available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/code_viewing_log_details.htm`.

Using the logging features of the Developer Console

Similar to the preceding debugging, one can use the Developer Console for the same thing. The following screenshot shows what logs can look like in the Developer Console:



https://ap1.salesforce.com/_ui/common/apex/debug/ApexCSIPage

Execute: Set <String> aSet = new Set <String>{'a', 'b', 'c'};

User	Application	Operation	Time	Status	Read
Abhinav Gupta	Browser	/_ui/common/apex/deb...	11/01 11:00:06	Success	

Log /_ui/common/apex/debug/ApexCSIAPI @11/01 11:00:06

Unit	Duratio	Heap
...	1.19	8
	1.17	8

Unit	Dura
add	0.01
add	0.01

Timestamp	Event	Details
11:00:06:048	ENTERING_MANAGED_PKG	
11:00:06:048	SYSTEM_METHOD_EXIT	[1]|SET<String>.add(Object)
11:00:06:048	VARIABLE_ASSIGNMENT	[1]|aSet|{"serId":1,"value":["b","c","a"]}|0x11039ca1
11:00:06:048	CUMULATIVE_LIMIT_USAGE	
11:00:06:048	LIMIT_USAGE_FOR_NS	abhinav|
11:00:06:000	LIMIT_USAGE_FOR_NS	Number of SOQL queries: 0 out of 100
11:00:06:000	LIMIT_USAGE_FOR_NS	Number of query rows: 0 out of 50000
11:00:06:000	LIMIT_USAGE_FOR_NS	Number of SOSL queries: 0 out of 20
11:00:06:000	LIMIT_USAGE_FOR_NS	Number of DML statements: 0 out of 150
11:00:06:000	LIMIT_USAGE_FOR_NS	Number of DML rows: 0 out of 10000
11:00:06:000	LIMIT_USAGE_FOR_NS	Number of script statements: 1 out of 200000
11:00:06:000	LIMIT_USAGE_FOR_NS	Maximum heap size: 0 out of 6000000
11:00:06:000	LIMIT_USAGE_FOR_NS	Number of callouts: 0 out of 10
11:00:06:000	LIMIT_USAGE_FOR_NS	Number of Email Invocations: 0 out of 10
11:00:06:000	LIMIT_USAGE_FOR_NS	Number of fields describes: 0 out of 100

To learn more about using the Developer Console for watching debug logs, please check the complete guide available at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_debugging_system_log_console.htm`.

Summary

This chapter covered a range of Apex tips and tricks starting from language basics to pretty advanced topics. A lot of emphasis was given on core concepts and basics such as file nomenclature, trigger, SOQL, and DML, which are mostly mistaken by developers. Tips about XML and JSON parsing will help you do integrations faster with the best selection of APIs. Security, governor limits, and testing is always something ignored. Following the tips about these issues will for sure make your life easier during AppExchange listing or production deployments. Lastly, debugging is something we all need to do sooner or later. Tips to debug via various resources makes sure all popular ways are covered.

In the next chapter about Visualforce, you will learn how to design Visualforce pages correctly with emphasis on key concepts such as re-using the native platform look and feel, security, and how to speed up page performance (view-state), and AJAX calls.

8
Writing Better Visualforce Code

Visualforce (**VF**) is the Force.com way to create custom screens. Developers from .NET and JEE (specially JSF) backgrounds will find a lot of similarity in custom tags, view states, and post backs. Just like Apex, it's important to understand that the VF pages execute in a multitenant environment. Thus, no one owns the server and can write pages that take minutes to process synchronous jobs. There is always a governor watching the resource consumption by page, and show can be stopped if the VF page consumes beyond the limits.

This chapter covers tips and tricks around key Visualforce areas such as:

- Differentiating facts about Visualforce architecture
- How to re-use the native look and feel in pages
- Avoiding too much copy/paste and reusing code
- Limiting view states
- Writing flexible pages via `FieldSets`
- Speeding up the Ajax calls
- Making the most out of global variables
- Avoiding common mistakes in JavaScript remoting
- Taking care of security in Visualforce pages
- Miscellaneous tips about charts, dynamic VF components, PDF rendering, and so on

Knowing the Visualforce architecture

In general, it's a good idea to first understand the VF architecture and order of execution before starting the coding. Having that foundation laid correctly is important. Here are few links for more information:

- **Visualforce architecture**: `http://www.salesforce.com/us/developer/docs/pages/Content/pages_intro_architecture.htm`
- **Order of execution in a Visualforce page**: `http://www.salesforce.com/us/developer/docs/pages/Content/pages_controller_lifecycle.htm`

With VF development, the most important piece of information to understand is caching behavior and session management, which is not similar to other languages/platforms such as Java, .NET, and PHP, and so on.

Let's compare this to how things work in Java (J2EE/JEE):

- **Caching class/page definitions**: Java servers cache all the class definitions in memory (RAM/Heap). Usually, on first request if the JSP page is not compiled/cached, it will be translated to a Java class definition and will be cached in Heap. On the next request for the page, the cached copy in heap will do the rendering.
- **Session management**: Java servers offer a variety of ways of session management and full control to alter it. Developers can tweak or replace a server's session management with their own as well. For example, bypassing cookies, using hidden fields, URL rewriting, and so on.
- **Session storage**: Developers can cache some data in a request-bound global variable, such as `Session` or `HttpSession`, and access it as required on every request.

Things work very differently in the Force.com platform, because each server is multitenant in nature and classes/pages of multiple customers/orgs live on the same server. Here are the differences:

- **Caching class/page definitions**: Because of the platform's multitenant nature, keeping definitions in memory is not possible. Compiled class/page definitions are stored in the database (metadata repository). To serve a page request, the compiled definitions are loaded into the memory for rendering the page.
- **Session management**: The platform takes care of session management and developers don't need to worry about it. As the platform complies with industry-leading security standards, and to protect customer data, Salesforce doesn't lets you tweak/replace the session management at all.

- **Session storage**: Again, because of the multitenant nature, there is no global variable such as `Session` available to developers, so any request-bound caching of data is not possible. Visualforce uses a scalable stateless mechanism called `ViewState`, which typically stores required request data for a flow.

Reusing the platform's native look and feel

There is a very basic and important **UX** (**User Experience**) aspect to consider while creating VF pages. I've seen developers going super fancy in styling, but that doesn't help on this platform; end users of VF pages are mostly the existing Salesforce customers (99 percent of the time). So it becomes unintuitive and a training issue if the VF page is not similar to the native look and feel.

An exception to this would be pages for public sites/portals, because end users are not the Salesforce customers, for example, developing a shopping portal on Force.com sites.

Most of the VF tags/components inherit the platform styling. Here are some tips and tricks about using key components to make your VF page look more like native pages.

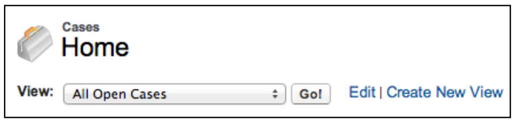
Starting the page design with native headers

The `<apex:sectionHeader>` component is mostly ignored by many Force.com developers. But it is the first step to start with a native look and feel. It adds a header section, which has the page title and subtitle. The following example shows how adding this tag starts your page with a native look and feel for the `Case` standard object:

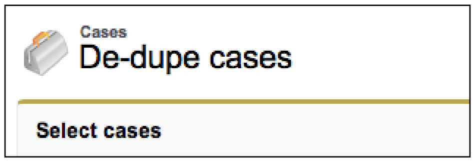
```
<apex:pagestandardController="Case">
<apex:sectionHeader title="Cases" subtitle="De-dupe cases"/>
<apex:pageBlock title="Select cases">
<!-- Your code goes here -->
</apex:pageBlock>
</apex:page>
```

The following screenshots show the VF page with the native cases tab styling:

- Native cases tab:



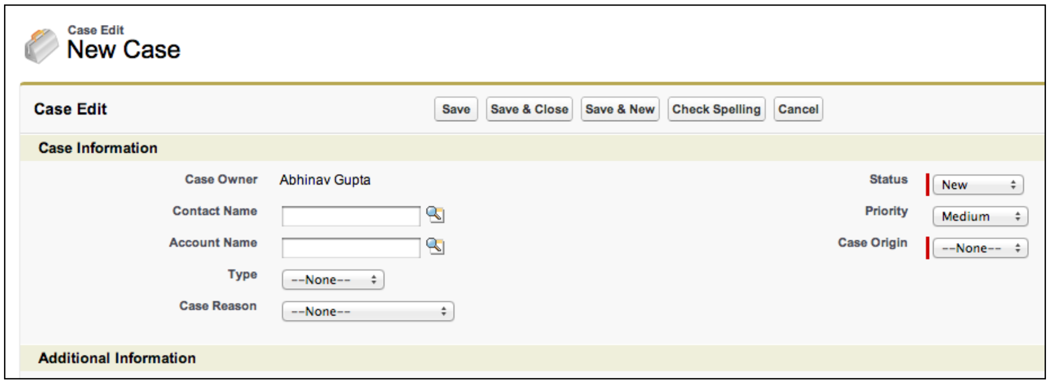
- Custom VF page:



Native detail sections/forms

Various components such as `<apex:pageBlock>`, `<apex:pageBlockSection>`, `<apex:pageBlockTable>`, and `<apex:inputField>` can help you in quickly laying out native looking detail sections. In most of the pages, these would be a second building block after the `<apex:sectionHeader>` component.

Here is how a native detail screen for **New Case** looks:



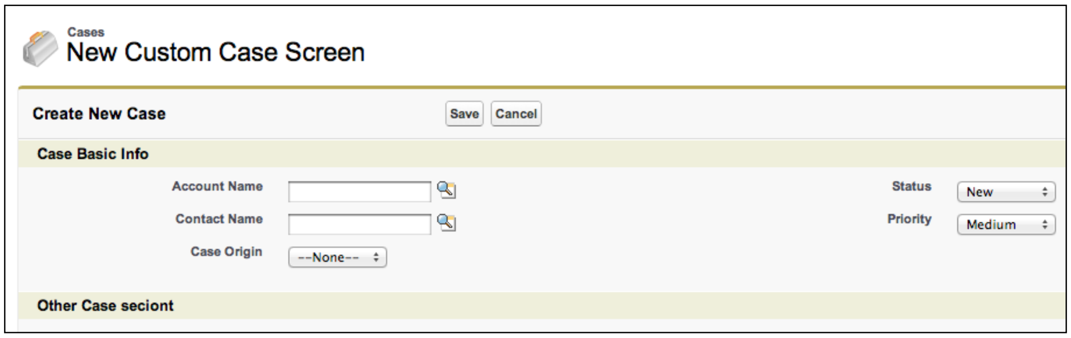
Case Owner	Abhinav Gupta	Status	New
Contact Name		Priority	Medium
Account Name		Case Origin	--None--
Type	--None--		
Case Reason	--None--		

Getting a similar look and feel in VF is easy, for example, refer to the following code snippet that creates a **New Case** screen in VF with a few lines of VF code:

```
<apex:pagestandardController="Case">
<apex:sectionHeader title="Cases" subtitle="New Custom Case Screen"/>
  <apex:form>
    <apex:pageBlock title="Create New Case" mode="edit">
      <apex:pageBlockSection title="Case Basic Info">
      <apex:inputField value="{!Case.AccountId}"/>
      <apex:inputField value="{!Case.Status}"/>
      <apex:inputField value="{!Case.ContactId}"/>
      <apex:inputField value="{!Case.priority}"/>
      <apex:inputField value="{!Case.Origin}"/>
      </apex:pageBlockSection>
        <apex:pageBlockSection title="Other Case section">
```

```
            <!-- Add more fields here -->
          </apex:pageBlockSection>
          <apex:pageBlockButtons>
            <apex:commandButton value="Save" action="{!save}"/>
            <apex:commandButton value="Cancel" action="{!cancel}"/>
          </apex:pageBlockButtons>
        </apex:pageBlock>
      </apex:form>
    </apex:page>
```

The preceding code snippet generates a **New Custom Case Screen** in VF, which is very similar in the look and feel to the preceding native one.



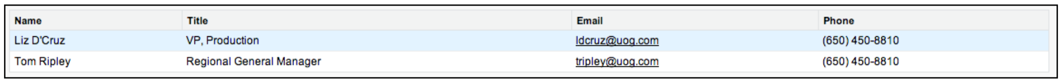
Using the `columns` attribute in the `<apex:pageBlockSection>` component helps change the layout from default two-column to a higher or lower value.

Styling tables to look like native grids

Similarly, components such as `<apex:pageBlockTable>` can be used to get grids/tables with the native look and feel. The best part is that this component works well with sObjects, so it automatically gets column headers as subject field labels (without any extra code), for example:

```
<apex:pageBlockTable value="{!account.Contacts}" var="con">
  <apex:column value="{!con.name}"/>
  <apex:column value="{!con.Title}"/>
  <apex:column value="{!con.Email}"/>
  <apex:column value="{!con.phone}"/>
</apex:pageBlockTable>
```

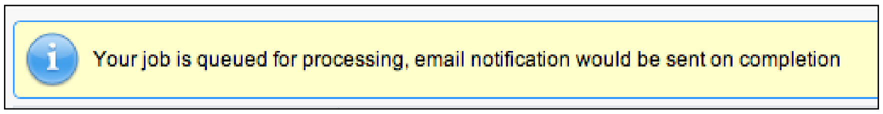
The preceding code gets the list of contacts of the accounts from the controller. Each element is stored in the `con` variable. The fields of contacts can then be accessed by the syntax `{!con.fieldName}`. Each contact is printed in a single row. On running, it renders a native looking grid as shown in the following screenshot:



Name	Title	Email	Phone
Liz D'Cruz	VP, Production	ldcruz@uog.com	(650) 450-8810
Tom Ripley	Regional General Manager	tripley@uog.com	(650) 450-8810

Printing messages in a native style

It's pretty simple to show messages of information, warning, or error type in VF with a native look and feel. Developers can print messages directly in a page using the `<apex:pageMessage/>` tag. The following screenshot shows a native looking informative message printed via VF code:



Your job is queued for processing, email notification would be sent on completion

The following single line of code gets you the preceding message:

```
<apex:pageMessage summary="Your job is queued for processing, email
notification would be sent on completion" severity="info" strength="3"
/>
```

If required, messages can be added via Apex and shown on the VF page as follows:

- **Controller**:

  ```
  public void onValidate() {
    if (event.Registeration_Deadline__c < System.now()){
      ApexPages.addMessage(new
      ApexPages.Message(ApexPages.Severity.ERROR,
      'Registration for this event is closed now.'));
    }
    ...
  }
  ```

- **VF page**: Needs to have the `<apex:pageMessages/>` component re-rendered

Always re-render the `<apex:pageMessages>` component on almost all VF AJAX calls. Doing this helps many times in trapping both user- and platform-generated messages gracefully.

Try using various severity levels in both Apex and Visualforce depending on situations such as `Confirm`, `Error`, `Fatal`, `Info`, and `Warning`.

Native (standard) versus custom controllers/ extensions

Standard controllers provide a rich set of platform functionalities to developers. They come in two flavors, standard controllers and standard list controllers. The former is good for working with single records and the latter works with a list of records. On a high level, they provide the following key functionalities:

- Security regarding the user's **FLS** (**field-level security**) and other permissions is taken care of automatically.
- The standard controller provides easy access to object fields, including parent and child relationships, without writing any code to query them. The standard controller detects the fields used in the page and queries them transparently. You can learn more about this at `http://www.salesforce.com/us/developer/docs/pages/Content/pages_controller_std_access_data.htm`.
- Standard CRUD actions (such as save, quick save, edit, delete) and list actions with pagination (next, previous, first, last) are available. Links from the Apex Developer's Guide gives a complete idea about the available methods:
    - `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_pages_standardcontroller.htm`
    - `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_pages_standardsetcontroller.htm`

It's a general rule of thumb to make the most out of standard controllers whenever possible. Here are a few code snippets from the VF Developer's Guide that show standard controller features:

- `http://www.salesforce.com/us/developer/docs/pages/Content/pages_controller_sosc_pagination.htm`
- `http://www.salesforce.com/us/developer/docs/pages/Content/pages_controller_sosc_edit_data.htm`

Apart from all the preceding features, sometimes it's required to go custom in some situations, such as:

- A complex page that performs DML on multiple object types simultaneously, for example, a page to create an account and multiple associated contacts in one go (a click on the **Save** button)
- Web service callouts are required during the page flow
- You need to send e-mails via Apex

Even in the preceding scenarios, Force.com developers should make the first attempt to get the solution by using custom controller extensions, because extensions retain most of the standard goodness and add new as required on top of it. For example, the following code snippet shows how stock price can be queried by using an extension, while retaining the native goodness:

```
<apex:pagestandardController="Account" extensions="DynamicBindingExt">
  <apex:pageMessage summary="Stock price for {!Account.Name} is
  {!stockPrice}" severity="info" strength="3" />
  <apex:detail subject="{!account.id}" />
</apex:page>

public with sharing class DynamicBindingExt {
  public Decimal stockPrice{get;set;}
  public DynamicBindingExt(ApexPages.StandardController controller) {
    Account acc = (Account)controller.getRecord();
    String recordName = acc.Name;
    // stockPrice = result from HTTP callout based on account name
  }
}
```

So, the key takeaway from here is, avoid resorting to a custom controller unless really required. It doesn't mean custom controllers are not good for any use, but the key is to use standard platform goodness as much as one can.

Reusing VF code

Code re-use on VF is important, especially in medium and large projects, where multiple developers might try to reinvent the wheel either by:

- Writing the same markup and logic again and again
- Copy-pasting the VF code across different pages

This practice creates a maintenance nightmare in the long run for sure. The good news is that the platform offers a couple of components that make code re-use easy.

Including other VF pages

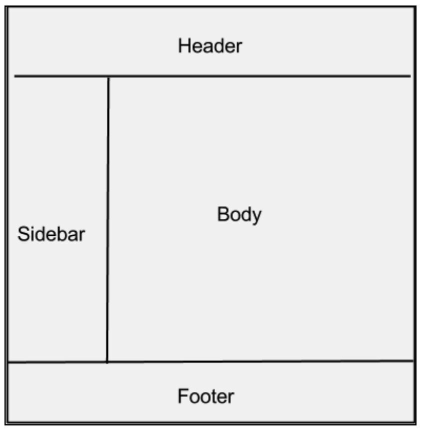
The `<apex:include>` component offers the simplest form of code re-use, where an existing VF page is inserted into another.

For code snippets and more details, please visit `http://www.salesforce.com/us/developer/docs/pages/Content/pages_compref_include.htm`.

[This component is good for simple code re-use requirements only.]

Defining templates or page layouts

The `<apex:composition>` component offers a sophisticated template-based approach for page content. It's particularly very useful when a couple of pages need to follow a common layout pattern, which includes a common header, sidebar, and footer with a variable body. The following diagram shows one sample layout:



[This component is best used in the Salesforce sites, where all the public pages need to adhere to a common layout. On creating/enabling sites in your org, the platform-generated pages such as `SiteTemplate.page`, give a good idea about the best use of this component.]

For code snippets and more details, please visit `http://www.salesforce.com/us/developer/docs/pages/Content/pages_compref_composition.htm`.

Defining your own components

Similar to standard components, developers can define their own components by using the `<apex:component>` tag. These components are not just a flat piece of code for inclusion, such as the `<apex:include>` tag, a component can be super dynamic and customizable to suit various inclusion scenarios. Here are some key features of components:

- They can have their own controllers, with the ability to query sObjects, make HTTP callouts, and make DML calls
- They can take attributes from the container page to customize behavior and appearance

For code snippets and more details, please visit `http://www.salesforce.com/us/developer/docs/pages/Content/pages_compref_component.htm`.

Limiting view states

As HTTP is a stateless protocol, all the state of page travels back and forth via a string called `ViewState`. It essentially keeps the following items:

- All non-transient attribute data associated with controllers/extensions
- Object graph that is reachable from a non-transient attribute in controllers/extensions
- Page's component structure and the associated state having values applied to those components
- Trivial data for VF runtime housekeeping

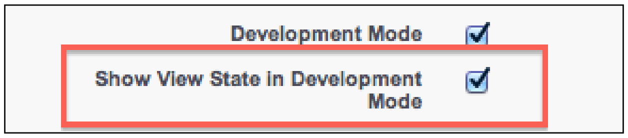
It's highly recommended that you know how to minimize view states, for the following two reasons:

- VF runtime imposes a limit of 135 KB on view states (as of Winter '13 release) and an exception is thrown if this limit is breached
- Bigger view states hurt the performance of Visualforce, because more data is travelling and processing over the wire

Here are some tips to minimize the view state.

Use the view state inspector

Apart from the other tips to minimize view state, it's always good to use the view state inspector, in case the page is performing slowly or hitting the limits. To enable the view state inspector, go to **Setup** | **Personal Setup** | **My Personal Information** | **Personal Information** | **Edit** (button). Here make sure that **Show View State in Development Mode** is checked, as indicated in the following screenshot:


Development Mode ✓
Show View State in Development Mode ✓

Using static when possible

Declare constants and variables using static variables in the controller, as static variables are not part of the view state. The good part about `static` is they are restored back to their original value on every request.

Here are some examples for using `static` in different ways to save the view state:

```
public with sharing class CustomContactController {
  // declared constant
  static final String RECORD_TYPE_NAME = 'My_Rec_Type';
  // one can do direct queries
  public static final Account [] accounts =
  [Select Id, Name from AccountWhere BillingCity like
  :ApexPages.currentPage().getParameters().get('billcity')];

  // getter setter for VF access are possible, logic to populate it
  in static block
  public static final Contact[] contacts {get;set;}

  static {
    // any complex processing can be done in static block
    // please note this block will be executed on every page refresh
    and ajax request
    String reqdPhone =
    ApexPages.currentPage().getParameters().get('phone');
    if (reqdPhone != '' && reqdPhone != null) {
      contacts = [Select Id, Name from Contact
      Where Phone like :reqdPhone];
    }
  }
}
```

Learn more about static variables at `http://www.tgerm.com/2010/09/visualforce-apex-static-instance.html`.

Trimming the view state by using transient variables

The `transient` keyword marks the instance variables, which shouldn't be transferred as part of the serialization process. Here is a precise definition of the serialization process from wikipedia (`http://en.wikipedia.org/wiki/Serialization`):

> *Serialization is a process of converting a data structure or object state into a format that can be stored (for example, in a file or memory buffer, or transmitted across a network connection link)*

This serialization can happen at a couple of locations in Apex or VF code, for example:

- View state, which serializes the controller/extension information to hidden HTML form fields
- Apex REST calls, where instance-level attributes are serialized/de-serialized from the JSON or XML string, learn more about this at `http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_rest_methods.htm`
- Other classes that are serializable, such as the `Batchable` or `Schedulable` interface

Here is an example of limiting the view state, the following code snippet marks the `User` list as `transient` as that info is required as read-only for once on the VF page:

```
public class CaseCustomController {
  public List<Case> casesToReassign {get;set;}
  /*
    marking this collection as transient, as we don't need the
    collection back on form submit or post back.
  */
  public transient List<User> availableSupportReps {get;set;}
}
```

Identifying a transient variable

To identify attributes as `transient` in a class, controller, or extension, check if certain attributes:

- Are only required at the time of page rendering, for example, a read-only list of records
- Are not submitted back during post backs (AJAX calls)
- Can be restored on post backs using other view state information such as record IDs

Avoid multiple forms

It is a recommended practice to avoid creating multiple HTML forms using the `<apex:form>` tags, as this ends up repeating the hidden view state variable for each form. The real problem starts on post backs, when all view states across all forms are submitted back to servers. For example:

```
<!-- Using multiple forms -->
<apex:page controller="CustomAccountController">

  <!-- form 1 -->
  <apex:form>
    <apex:inputText value="{!accountName}"
    <!-- Other account fields for search -->
    <apex:commandButton action="{!searchAccounts}" value=
    "Search Accounts"/>
  </apex:form>

  <!-- form 2 -->
  <apex:form>
    <apex:inputField value="{!acc.Name}" />
    <!-- More inputfields for Account -->
    <apex:commandButton action="{!quickCreateAccount}" value=
    "Quick Create Account"/>
  </apex:form>

</apex:page>
```

The solution to this problem is to remove multiple forms and demarcate them with action regions, as shown in the following code snippet:

```
<!-- Using action regions -->
<apex:page controller="CustomAccountController">

  <!-- single form -->
  <apex:form>
    <apex:actionRegion>
      <apex:inputText value="{!accountName}"
      <!-- Other account fields for search -->
      <apex:commandButton action="{!searchAccounts}" value=
      "Search Accounts"/>
    </apex:actionRegion>

    <apex:actionRegion>
      <apex:inputField value="{!acc.Name}" />
      <!-- More inputfields for Account -->
      <apex:commandButton action="{!quickCreateAccount}" value=
      "Quick Create Account"/>
    </apex:actionRegion>
  </apex:form>

</apex:page>
```

But with the Summer '12 release, the Single View State feature is GA. So even if you create multiple forms, they will share the single view state. So the problem is solved, but using action region is still a best practice to follow under such scenarios.

Query-required fields only

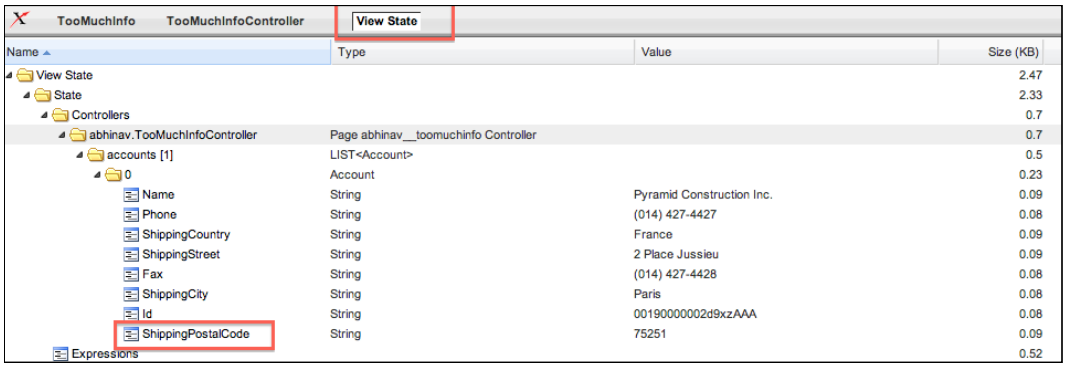
SOQL queries should fetch only those fields/columns which are required in business logic or presentation on page. For example, here is a VF page to show an editable grid of accounts with two columns, `Name` and `Description`:

```
<apex:pageBlockTable value="{!accounts}" var="acc">
  <apex:columnheaderValue="Name">
    <apex:inputField value="{!acc.Name}"/>
  </apex:column>
  <apex:columnheaderValue="Description">
    <apex:inputField value="{!acc.Description}"/>
  </apex:column>
</apex:pageBlockTable>
```

Controller for the same is doing all correct, but fetching too many columns that are not required such as phone, fax, and shipping detail fields, and so on.

```
public with sharing class TooMuchInfoController {
  public Account[] accounts {get;set;}
  public TooMuchInfoController() {
    accounts = [SELECT Name, Phone, Site, Description, Fax,
    ShippingCity, ShippingCountry, ShippingState, ShippingStreet,
    ShippingPostalCode FROM Account LIMIT 10];
  }
}
```

Doing this ends up in a bigger view state, which can be easily checked using the view state inspector, as indicated in the following screenshot:



Name	Type	Value	Size (KB)
View State			2.47
State			2.33
Controllers			0.7
abhinav.TooMuchInfoController	Page abhinav__toomuchinfo Controller		0.7
accounts [1]	LIST<Account>		0.5
0	Account		0.23
Name	String	Pyramid Construction Inc.	0.09
Phone	String	(014) 427-4427	0.08
ShippingCountry	String	France	0.09
ShippingStreet	String	2 Place Jussieu	0.09
Fax	String	(014) 427-4428	0.08
ShippingCity	String	Paris	0.08
Id	String	00190000002d9xzAAA	0.08
ShippingPostalCode	String	75251	0.09
Expressions			0.52

More tips and tricks on the view state

For learning more about the view state and how to minimize it, go through the excellent article from the Developer Force wiki at `http://wiki.developerforce.com/page/An_Introduction_to_Visualforce_View_State`.

Flexible pages using field sets

Customers often require the flexibility to change/add fields in VF pages, just like page layouts. This is possible to a certain extent with field sets, which are basically a set of fields decided by the point and click configurations. This becomes especially important when you are developing AppExchange apps, and depending on standard objects. In most cases, customers have their own custom fields created on standard objects, which they want to be visible/editable on the app's VF pages as well.

To know how to create field sets, please visit `https://ap1.salesforce.com/help/doc/en/fields_editing_field_sets.htm`.

To learn how to use field sets in Apex and VF code, please visit `http://www.salesforce.com/us/developer/docs/pages/Content/pages_dynamic_vf_field_sets.htm`.

Field sets offer the capability of marking certain fields as required via point and click. Apex or VF code can read this configuration for required actions.

```
<apex:repeat value="{!$ObjectType.Contact.FieldSets.
auditform}" var="f">
  <apex:inputField value=
  "{!Contact[f]}" required="{!f.required}"/>
</apex:repeat>
```

Speeding up Ajax calls

Ajax adds fun and responsiveness to any web page, but the fun goes away if the Ajax call takes a while to complete. This section shares some key tips so that developers can boost Ajax performance in their Visualforce pages.

Using the immediate attribute

This attribute is available on all VF- and Ajax-capable components such as `<actionFunction>`, `<actionPoller>`, `<actionSupport>`, `<commandLink>`, and `<commandButton>`. This attribute is by default set to `false`, which means all validation rules must be processed before an AJAX call. Sometimes, it's not necessary to care about doing any validations, such as:

- Clicking on the **Cancel** or **Reset** buttons, which wipes almost all forms of data
- Clicking on the **Delete** button on the record grid, which just takes the ID of the record to be deleted in `<apex:param>` and refreshes the grid

A huge performance boost can be achieved if the `immediate` attribute is set to `true` under the preceding scenarios.

Re-rendering required components only

New developers sometimes ignore the `reRender` attribute, which is available in all VF- and AJAX-capable components. This attribute tells us which sections/components of page should be refreshed from new information available after an AJAX call. Leaving this attribute refreshes the complete page, which for sure takes more time.

```
<apex:pageMessages id="messages"/>
<!-- … other markup and components -->
  <apex:pageBlockSection id="contactFields">
    <apex:repeat value="{!$ObjectType.Contact.FieldSets.auditform}"
    var="f">
```

```
        <apex:inputField value="{!Contact[f]}"
        required="{!f.required}"/>
      </apex:repeat>
    </apex:pageBlockSection>
  <!-- Re-render only the contactFields and messages sesion -->
  <apex:commandButton value="Save" action="{!save}"
  reRender="contactFields, messages"/>
```

Using the `reRender` attribute also safeguards you from some VF bugs. Components such as `<apex:commandButton/>` don't pick the `<apex:param/>` inputs correctly, if the `reRender` attribute is not specified. For more details, please visit `http://success.salesforce.com/ideaView?id=08730000000YcV8AAK`.

Demarcating using action regions

AJAX is about partial page refreshes, but sometimes it's good to control the number of components and amount of data being processed with every AJAX request. `<apex:actionRegion>` helps in demarcating that boundary so that a limited number of components participate in various processing steps. This not only speeds up the AJAX performance, but also helps in some cases where an AJAX call has to bypass some validations.

To learn more about the action region, please check the articles available at:

- `http://www.salesforce.com/us/developer/docs/pages/Content/pages_compref_actionRegion.htm`
- `http://www.tgerm.com/2010/09/visualforce-actionregion-deep-dive.html`

Global variables and functions

Though this topic is mentioned in the Appendices section of the VF Developer's Guide, it is very important to understand as a developer.

Both global variables and functions are used via the following expression syntax:

```
{!$VAR or Function}
```

Consider the examples discussed in this section.

The `$Action global` variable can be used to link to a new account page:

```
<apex:outputLink value="{!URLFOR($Action.Account.New)}">
    Create New Account
</apex:outputLink>
```

For a multilingual app using custom labels, we can refer to those labels in a page, for example, a custom label named `error_msg_invoice` could be accessed in a page as follows:

```
{!$Label.error_msg_invoice }
```

A safe hyperlink to another VF page named `AccountCreationWizard` can be created as follows:

```
<apex:outputLink value="{!$Page.AccountCreationWizard }">
    Start Account Creation
</apex:outputLink>
```

To access field labels in a page, one can use `$ObjectType` as follows:

```
{!$ObjectType.Account.Fields.Name.Label}
```

The current time can be printed using the `NOW()` function:

```
Current Time: {!NOW()}
```

For a complete reference of global variables and functions, please visit `http://www.salesforce.com/us/developer/docs/pages/Content/pages_variables.htm`.

JavaScript remoting

JavaScript remoting offers super fast AJAX calls to methods in Apex controllers via JavaScript. If you don't know about JavaScript remoting, here is a recommended prerequisite reading: `http://www.salesforce.com/us/developer/docs/pages/Content/pages_js_remoting.htm`

Here are a few key implementation tips to consider while using JS remoting.

Public versus global – using the right access modifier

During the initial launch of JS remoting, it was mandatory to make Apex controllers `global`, but with recent releases, this limitation is pulled off, and Apex controllers can be `public` as well. It's strongly recommended to keep minimal `global` Apex code in org, specially when developing apps for managed packages, because global classes/methods are not maintenance friendly and can't be renamed or pulled off in newer releases.

Making the most out of the method arguments and return types

JavaScript remoting supports both primitive and complex types in both arguments and return values. This opens possibilities to serialize sophisticated information back and forth in the form of JSON between page and controller.

Here is a quoted text from the official guide:

> *Your method can take Apex primitives, collections, typed and generic sObjects, and user-defined Apex classes and interfaces as arguments. Generic sObjects must have an ID or sobjectType value to identify actual type. Interface parameters must have an apexType to identify actual type.*
>
> *Your method can return Apex primitives, sObjects, collections, user-defined Apex classes and enums, SaveResult, UpsertResult, DeleteResult, SelectOption, or PageReference.*

Handling namespace prefixes in managed packages

There are two approaches to call remote Apex controllers via JS remoting.

Approach 1

Refer to the following code snippet:

```
[namespace.]controller.method([parameters...,]  callbackFunction,
[configuration] );
// namespace :abhinav
// controller :GoogleChartsController
// method :loadOpps
// parameter: accountName = document.getElementById('acctSearch').
value;
abhinav.GoogleChartsController.loadOpps(accountName,
function(result, event){
alert('Total Records:' + result.length);
    }, {escape:true});
```

This approach should be used when working in an unmanaged environment, that is, without namespace prefixes. A good example of that would be pages developed as part of org customization done in sandboxes for one Salesforce org.

Approach 2

Refer to the following code snippet:

```
Visualforce.remoting.Manager.invokeAction('fully_qualified_remote_
action',       invocation_parameters );
// fully qualified remote action: GoogleChartsController
// invocation parameter: accountName = document.
getElementById('acctSearch').value;
Visualforce.remoting.Manager.invokeAction(
     '{!$RemoteAction.GoogleChartsController.loadOpps}',
accountName,
function(result, event){
alert('Total Records:' + result.length);
     },
     {escape: true}
);
```

This approach works well in all scenarios as it resolves namespaces correctly if working across both managed and unmanaged.

Taking care of security compliance in pages

When working with Apex and Visualforce it's pretty easy to go wrong and breach security. If you are developing/listing an app on AppExchange, the security review process makes sure your app complies with the guidelines. But it's good to take care of security, if you're doing Force.com customization for a single org.

For all Force.com developers, this security guideline is a highly recommended reading: `http://wiki.developerforce.com/page/Secure_Coding_Guideline`

Encode/escape

It's mostly safe to encode/escape the stuff getting printed on a page. Most of the Visualforce components, such as `<apex:outputField>`, `<apex:outputText>`, and so on take care of escaping by default. But in a few cases, it's good to encode the text server side, for example, the following code prints account ID on a page:

```
/apex/MyPage?Id={!$CurrentPage.parameters.Id}
```

It will break in case user has passed an attack string in ID, as shown in the following code line:

```
/apex/MyPage?Id=<script>alert('XSS');</script>
```

The simple fix to such situations is to use one of the various encoding functions such as `JSENCODE`, `HTMLENCODE`, `JSINHTMLENCODE`, and `URLENCODE` depending on the situation. The preceding code could be fixed by using the following snippet:

```
/apex/MyPage?Id={!JSENCODE($CurrentPage.parameters.Id)}
```

To learn more about the escaping function, please check these links:

- `http://www.salesforce.com/us/developer/docs/pages/Content/pages_variables_functions.htm`
- `http://wiki.developerforce.com/page/Secure_Coding_Cross_Site_Scripting#Apex_and_Visualforce_Applications`

Enforcing CRUD and FLS

The Object (CRUD) and **field-level security** (**FLS**) settings come from profiles. It's used to restrict access on object types and individual fields based on different profiles' permissions.

In most scenarios, the platform and VF runtime will transparently take care of CRUD/FLS enforcements transparently, for example:

- Merge fields, that is, `{!object.field}`, are checked for required permissions during page rendering
- Tags such as `<apex:inputField>` and `<apex:outputField>` work based on the preceding merge field concept, the `inputField` tag doesn't render or renders as read-only, and the `outputField` tag doesn't print on missing permissions

Apart from this, developers need to take care of security in various scenarios, where they are bypassing native features, for example:

- Copying object field values in the `controller` attributes or methods, and later printing them on the VF page:

```
<apex:page controller="CustomContactController">
  <apex:outputText value="{!FullName}" />
</apex:page>

public with sharing class CustomContactController {
  public String getFullName() {
    Contact con = [SELECT FirstName, LastName
                  FROM Contact
                  WHERE Id =:contactId];
                  returncon.FirstName + ' ' + con.LastName;
  }
}
```

The preceding code snippet can be fixed either in Apex or VF code. Both approaches are shown as follows (please opt for one of them only):

- **Fixed VF page**:

```
<apex:page controller="CustomContactController">
  <apex:outputText value="{!FullName}"rendered="
  {!AND($ObjectType.Contact.fields.FirstName.Accessible,
  $ObjectType.Contact.fields.LastName.Accessible)}"/>
</apex:page>
```

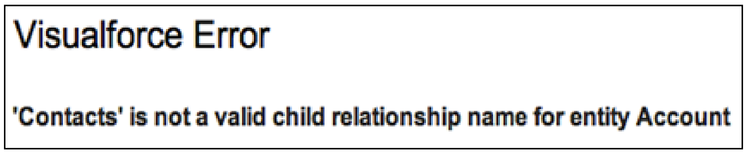
- **Fixed controller**:

```
public with sharing class CustomContactController {
public String getFullName() {
  if
  (!Schema.sObjectType.Contact.fields.
  FirstName.isAccessible()
  ||
  !Schema.sObjectType.Contact.fields.LastName.
isAccessible()
  ){
    /*
      either return blank or throw an error here,
      based on biz requirements
    */
    return '';
  }

  Contact con = [SELECT FirstName, LastName
                FROM Contact
                WHERE Id =:contactId];
                returncon.FirstName + ' ' + con.LastName;
  }
}
```

- Custom DML operations and access in other components such as related lists. The following code snippet checks whether the `Contact` object is accessible via the current profile:

```
<apex:relatedList list="Contacts" rendered="{!$ObjectType.Contact.
accessible}"/>
```

If the preceding check is missing in the `relatedList` component, it would crash the page for the following error in case the profile lacks the permission:



Visualforce Error

'Contacts' is not a valid child relationship name for entity Account

Similarly, if custom DML is performed via command buttons, checks can be added as follows:

```
<apex:commandButton value="Update"  rendered="{!$ObjectType.
Account.Updateable}" action="{!customAccountUpdate}"/>
```

Learn more about enforcing CRUD/FLS in VF and Apex at:

- `http://wiki.developerforce.com/page/Enforcing_CRUD_and_FLS`
- `http://wiki.developerforce.com/page/Testing_CRUD_and_FLS_Enforcement`

Miscellaneous tips

As the name suggests, this section has tips associated with miscellaneous areas, such as querying data, rendering pages as PDF or via dynamic components, or using native charts.

Querying a million rows

By default, an Apex code can only query up to 50,000 rows. This might look limiting in some situations. To overcome this, you can use the `readOnly` attribute on the `<apex:page />` component. This attribute boosts the row limit to 1 million rows, but it restricts the page's ability to do any DML operations. Still, this makes sense as you can use this attribute in pages which are generating read-only fancy reports, charts, and so on.

Rendering a page as a PDF

VF runtime gives the option to render a page as a PDF by setting `renderAs` to `pdf` in the Apex page tag, as follows:

```
<apex:pagerenderAs="pdf" />
```

Here are some links to a bunch of good tricks for printing sophisticated pages, such as adding page numbers or images in footers, controlling the styling of PDFs, and so on:

- Creating professional PDF documents with CSS and Visualforce: `http://wiki.developerforce.com/page/Creating_Professional_PDF_Documents_with_CSS_and_Visualforce`
- Header and footer tweaks: `http://forceguru.blogspot.in/2010/12/header-footer-in-pdf.html`

Dynamic VF components

Sometimes some complex component structure can't be drawn out directly, using Visualforce code, that is, some components of a page need to be programmatically crafted out of some logic in Apex, which can't be achieved anyway in VF code. In those situations, it's a good idea to use dynamic Visualforce components. For example, the Apex code to create `apex:outputText` would be as follows:

```
Component.Apex.OutputTextopText opText = new Component.Apex.
OutputText();
opText.value = account.BillingCity; // Billing City based on
previously loaded Account record
```

Similarly Apex equivalents are available for almost all VF components. A very good example of a realistic situation to created related lists is available in the VF Developer's Guide at `http://www.salesforce.com/us/developer/docs/pages/Content/pages_dynamic_vf_components_sample.htm`.

Charts

When it comes to presenting information as charts in Visualforce, one can either use third-party APIs such as Google Charts or native Visualforce charting, which is GA in the Winter '13 release. Unless the client is inclined towards a third-party charting solution, it's recommended to use Visualforce charting, because:

- You need to make sure the license and terms of use are compatible with your app's business model.

- You might need to pay an extra cost and be bound to usage and load limitations imposed by the third-party API. No extra cost has to be paid to use Visualforce charting.
- Your app's pages can be down or crash during maintenance windows of third-party servers. Visualforce charting complies with platform maintenance windows, so the app as a whole performs well.
- Mostly least plumbing code is required to connect Visualforce charting with your app, this code snippet illustrates the simplicity to draw a chart without any complex code:

```
<apex:pagestandardController="Opportunity" recordSetVar="opps">
  <apex:chart data="{!opps}" width="600" height="400">
    <apex:axis type="Category" position="left" fields=
    "Name" title="Opportunities"/>
    <apex:axis type="Numeric" position=
    "bottom" fields="Amount" title="Amount"/>
    <apex:barSeries orientation="horizontal" axis=
    "bottom"  xField="Name" yField="Amount"/>
  </apex:chart>
</apex:page>
```

Summary

This chapter took you through various Visualforce areas, with a variety of tips and tricks starting from understanding the VF architecture, getting the native look and feel in your pages, and making good use of the available global variables. Code re-use is very important in almost all projects. This chapter covered tips to best re-use the code via various possible approaches. Big emphasis was given on improving page performance. We started with tricks about minimizing the view state, and then advanced to speeding up the Ajax calls and tuning JavaScript remoting to avoid common pitfalls. Security is something that is often ignored, but this chapter shares some good tips to help you write secure pages. Lastly, miscellaneous tips were shared about a variety of topics such as charting, dynamic VF components, and PDF rendering.

Index