Praise for

*Development with the Force.com Platform*

“Jason Ouellette’s book is useful right out of the gate—an incredibly practical deep dive into building business applications on the Force.com Platform. Whether you’re learning about database integration in Apex code or developing a custom user interface with Visualforce, this is a must-read for anyone who wants to build applications on the Salesforce.com Cloud.”

—Howard A. Brown, Founder and CEO, DemandResults

“A great resource for experienced programmers with wide-ranging coverage of intermediate/advanced topics with clear and well-explained code samples.”

—David Cheng, Principal, Clarity TechWorks

“Jason covers all the bases, from understanding the concepts of the Force.com platform all the way to advanced development. He’s packed in tons of code examples, instructions, considerations, and ideas to inspire and challenge for months to come.”

—Jeff Grosse, Senior Business Consultant, Blue Cross Blue Shield of Minnesota

“Jason has created a no-nonsense, easy-to-follow guide for Salesforce development. It shows how to get the best out of the platform using the internal tools, techniques, and API that are available to enhance any application based around Salesforce.”

—Brendan Lally, CTO for Startups, StartITup.net

“Jason’s book captures many insights and hard lessons Appirio gathered through years of Force.Com development and makes them easily accessible to beginners and advanced developers of business applications. A must-have for any company considering building on or porting apps to the Force.Com platform.”

—Narinder Singh, Co-founder and Head of R&D, Appirio
“This book gives developers, managers, and entrepreneurs an extensive technical overview of the Force.com platform and provides key insights only found through practical hands-on experience. This should be required reading for any serious developer of business applications today.”

—Jim Thompson, CEO of Rogue IT, Developer of Chargent and GreatVines for Force.com
Development with the Force.com Platform
This page intentionally left blank
To Tracey, for rolling with me
This page intentionally left blank
# Table of Contents

Foreword xvi  
Preface xix  
Acknowledgments xxi  
About the Author xxii  

## Chapter 1  Introducing Force.com  1  
Force.com in the Cloud Computing Landscape 1  
Platform as a Service (PaaS) 2  
Force.com as a Platform 4  
Force.com Services 7  
Inside a Force.com Project 9  
Project Selection 9  
Team Selection 11  
Lifecycle 12  
Tools and Resources 15  
Sample Application: Services Manager 17  
Background 17  
User Roles 18  
Development Plan 18  
Summary 19  

## Chapter 2  Database Essentials  21  
Overview of Force.com's Database 21  
Objects 21  
Fields 23  
Relationships 25  
Query Language 25  
Data Integration 28  
Working with Custom Objects 31  
Force.com Developer Edition 31  
Tools for Custom Objects 31  
Object Creation 33  
Field Creation 36  
Entering and Browsing Data 38
<table>
<thead>
<tr>
<th>Chapter 3 Database Security</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of Database Security</td>
<td>63</td>
</tr>
<tr>
<td>Object-Level Security</td>
<td>65</td>
</tr>
<tr>
<td>Profiles</td>
<td>66</td>
</tr>
<tr>
<td>Field-Level Security</td>
<td>67</td>
</tr>
<tr>
<td>Record-Level Security</td>
<td>69</td>
</tr>
<tr>
<td>Record Ownership</td>
<td>69</td>
</tr>
<tr>
<td>User Groups</td>
<td>70</td>
</tr>
<tr>
<td>Sharing Model</td>
<td>70</td>
</tr>
<tr>
<td>Sample Application: Securing Data</td>
<td>73</td>
</tr>
<tr>
<td>Designing the Security Model</td>
<td>74</td>
</tr>
<tr>
<td>Implementing the Security Model</td>
<td>78</td>
</tr>
<tr>
<td>Testing the Security Model</td>
<td>83</td>
</tr>
<tr>
<td>Summary</td>
<td>88</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 4 Additional Database Features</th>
<th>89</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependent Fields</td>
<td>90</td>
</tr>
<tr>
<td>Record Types</td>
<td>90</td>
</tr>
<tr>
<td>Defining Record Types</td>
<td>91</td>
</tr>
<tr>
<td>Securing Record Types</td>
<td>92</td>
</tr>
<tr>
<td>Using Record Types</td>
<td>93</td>
</tr>
<tr>
<td>Roll-Up Summary Fields</td>
<td>95</td>
</tr>
<tr>
<td>Field History Tracking</td>
<td>97</td>
</tr>
<tr>
<td>Tags</td>
<td>98</td>
</tr>
<tr>
<td>Enabling Tags</td>
<td>99</td>
</tr>
<tr>
<td>Using Tags</td>
<td>99</td>
</tr>
<tr>
<td>Force.com Connect Offline</td>
<td>100</td>
</tr>
<tr>
<td>Administration of Force.com Connect Offline</td>
<td>100</td>
</tr>
<tr>
<td>Using Force.com Connect Offline</td>
<td>102</td>
</tr>
</tbody>
</table>
Chapter 5 Business Logic 111

Introduction to Apex 112
Introducing the Force.com IDE 113
Installation 113
Force.com Perspective 113
Force.com Projects 114
Problems View 115
Schema Explorer 115
Apex Test Runner View 116
Execute Anonymous View 116
Apex Language Basics 116
Variables 117
Operators 121
Arrays and Collections 122
Control Logic 124
Understanding Governor Limits 129
Database Integration in Apex 130
Database Records as Objects 130
Database Queries 132
Persisting Database Records 137
Database Triggers 139
Database Security in Apex 142
Object-Oriented Apex 143
Encapsulation 143
Information Hiding 147
Modularity 148
Inheritance 149
Polymorphism 150
Debugging and Testing 151
Debugging 151
Testing 154
Custom Controllers 203
Controller Extensions 206
View Components 207
  View Component Basics 208
  Data Components 210
  Action Components 213
  Primitive Components 214
  Force.com-Styled Components 215
  Force.com User Interface Components 218
Visualforce and the Native User Interface 222
  Standard Pages 222
  Standard Buttons 224
  Page Layouts 225
  Custom Buttons and Links 226
  Custom Tabs 227
Visualforce in Production 227
  Security 228
  Error Handling 230
  Governor Limits 232
  Unit Tests 232
Sample Application: Skills Matrix 233
  Basic Implementation 234
  Full Implementation 235
  Implementation Walkthrough 236
Summary 242

Chapter 8 Advanced User Interfaces 245
  Asynchronous Actions 245
    Partial Page Refresh 246
    Action as JavaScript Function 247
    Action as Timed Event 248
    Action as JavaScript Event 249
    Indicating Action Status 250
  Modular Visualforce 252
  Static Resources 252
  Inclusion 253
  Composition 253
Foreword

In a famous (if possibly mythical) conversation, it’s said that Steve Jobs recruited John Sculley away from PepsiCo by asking one question: “Do you want to spend the rest of your life selling sugared water, or do you want to change the world?” Creating a new application platform is one of the tiny handfuls of ways that a small number of people can truly change the world, sometimes for decades to follow. Mastering a new platform is one of the most important investments that a business professional can make in adding massive leverage to other business and technical skills.

Radical platform improvements eliminate wasteful friction and complexity from the systems that we use to get things done. Breakthrough platform improvements liberate the talent of people who didn’t previously see IT as a medium for creative innovation. Cloud computing platforms go one step farther: They let anyone offer the benefit of their new view of a problem, or their new insight on performing a useful task, to anyone with an Internet connection in any part of the world.

That’s the vision that’s driven the creation of the Force.com cloud platform and that Jason Ouellette illuminates in this timely and valuable guide.

Few people are so well grounded in the subject at hand. Jason’s insights come from working with Appirio’s customers to solve real-world business problems at the vastly improved speed—and with the superior return on investment—that the cloud computing model should be expected to provide.

It would be essentially impossible for anyone to claim longer experience with cloud application development in general, let alone Force.com in particular—since some of the technologies that Jason describes, clearly and effectively, do things that were thought to be conceptually impossible until Force.com provided the first existence proofs in just the past few years. For example, before the advent of Apex code, many would have said that custom business logic necessarily implied infiltration of customer-specific changes into the code base of a packaged enterprise software product. Cost, complexity, and delayed exploitation of software upgrades were assumed to be unavoidable consequences.

Jason shares with developers his informed understanding of Apex Code as a multi-tenant programming language, surrounded by a natively cloud-based developer experience. These are among the most important concepts for a developer to grasp, if the goal is to use the cloud to its full potential instead of merely relocating the traditional developer experience—warts and all—to the other end of the wire.
Thinking back over the last few generations of application development, it seems as if each major breakthrough has moved the center of development toward the most important problem to be solved…whatever that problem might be at any given time.

- When the cost and complexity of hardware were the biggest IT problems, COBOL- and FORTRAN-era development was centered on maximizing computational efficiency. Ease of use often suffered.
- When ease of end-user learning became the biggest IT problem, Visual Basic and other RAD tools were centered on crafting an accessible user interface and wiring up the application function behind it. Application portfolio coherence and consistency often suffered.
- Today, hardware is cheap, and the user interface conventions of the Internet are widely understood. The hardest problems in business computing today are those of connection, collaboration, and coherence of the data that should be carefully controlled even as it’s consistently and conveniently shared.

Times like these demand a development environment in which connection is implicit, collaboration is straightforward, and coherence of data becomes the path of least resistance rather than requiring continual (and often futile) effort to achieve. The standards-based, Internet-accessible cloud is that environment: The Force.com platform is the first and most fully realized demonstration of the possibilities that thus arise.

It is, therefore, a great pleasure to recommend Jason Ouellette as your trusted guide to this new world. This is where the next generation of successful developers will find their opportunities for professional contribution and personal achievement. Enjoy the flight.

Craig Weissman, CTO, Salesforce.com
This page intentionally left blank
I wrote this book to help developers discover Force.com as a viable, even superior tool for building business applications.

I’m always surprised at how many developers I meet who aren’t aware of Force.com as a platform. They know of Salesforce, but only that it’s a CRM. Those who have heard of Force.com are surprised when I describe what Appirio and other companies are building with it. “I didn’t know you could do that with Force.com” is a common reaction, even to the simplest of things such as creating custom database tables.

I hope that this book is effective in introducing business application developers to what Force.com offers. This is a combination of its features as a development platform and the benefits of it being “in the cloud,” delivered over the Internet as a service rather than installed on your own servers. I believe you’ll find, as I did, that Force.com can save you significant time and effort for many classes of applications.

**Key Features of This Book**

This book covers areas of Force.com relevant to developing applications in a corporate environment. It takes a hands-on approach, providing code examples and encouraging experimentation. It includes sections on the Force.com database, Apex programming language, Visualforce user interface technology, integration to other systems, and supporting features such as workflow and analytics. SFA, CRM, customer support, and other pre-built applications from Salesforce are not discussed, but general Force.com platform skills are helpful for working in these areas as well. The book does not discuss cloud computing in general terms. It also avoids comparing Force.com with other technologies, platforms, or languages. Emphasis is placed on understanding Force.com on its own unique terms rather than as a database, application server, or cloud computing platform.

Although Force.com is a premium service sold by Salesforce, all the material in this book was developed using a free Force.com Developer Edition account. Additionally, every feature described in this book is available in the free edition.

Throughout the text you will see sidebar boxes labeled Note, Tip, or Caution. Notes explain interesting or important points that can help you understand key concepts and techniques. Tips are little pieces of information that will help you in real-world situations, and often offer shortcuts to make a task easier or faster. Cautions provide information about detrimental performance issues or dangerous errors. Pay careful attention to Cautions.
**Target Audience for This Book**

This book is intended for application developers who use Java, C#.NET, PHP, or other high-level languages to build Web and rich-client applications for end users. It assumes knowledge of relational database design and queries, Web application development using HTML and JavaScript, and exposure to Web services.

**Code Examples for This Book**

The code listings in this book are also available on the book’s Web site: http://www.informit.com/title/9780321647733. They are also available as a Force.com package, freely available on Force.com AppExchange: http://sites.force.com/appexchange/listingDetail?listingId=a0N30000001SS3rEAG. The package can be installed directly into your own Force.com organization.
Acknowledgments

There are many people to thank for this book.

- **Mark Taub**: Mark is the Editor-in-Chief at Pearson. At Dreamforce 2008, Mark attended a presentation of mine about using Google Data APIs with Force.com. Despite this, he approached me with an idea on a book for Force.com development.

- **My coworkers at Appirio**: Many thanks go out to the Notorious P.S.E. crew of Narinder Singh, Ryan Nichols, Todd Bursey, Titash Bardhan, James Eitzmann, Bill Mers, Iein Valdez, Dayal Gaitonde, and Marlin Scott for supporting me during the writing process.

- **Trina MacDonald**: Trina is an Acquisitions Editor at Pearson. In our weekly meetings, she has guided me through the book-writing process and kept me on track and on time with the content. I can’t imagine this process running much more smoothly, and I have her to thank.

- **David Cheng, Brendan Lally, Colin Loretz**: The technical reviewers for this book have consistently provided valuable and timely feedback.

- **Paul Kopacki**: Paul is the VP of Developer Relations and Technical Enablement at Salesforce. He supported the book from the start. He validated my approach and helped me differentiate the book from the existing content available from Salesforce.

- **Ron Hess**: Ron is a Developer Evangelist at Salesforce. He’s developed many open-source libraries for Force.com, including my favorite, XmlDom. He was responsive to any question I had about the Force.com platform.

- **Songlin Qiu**: Songlin was my technical editor at Pearson. She is incredibly fast and thorough in reviewing draft chapters. She tirelessly flagged style usage mistakes that I continued to make even after 11 chapters of her polite corrections.

- **Olivia Basegio**: Olivia is the Editorial Assistant at Pearson. She worked behind the scenes to make the publishing process seem easy.

- **Tracey**: My wife and teammate in impossible missions, Tracey and I have undertaken larger and scarier projects than this book and steamed right through them. As always, she fed me and kept the wheels from flying off. I couldn’t have done it without her.
About the Author

Jason Ouellette has been working with Force.com since 2004. He developed three of the ten most popular applications on AppExchange, the official Force.com application marketplace, including the #1 most installed application, Appirio Calendar Sync for Salesforce and Google Apps. He is Chief Architect for Appirio, a leading Force.com Independent Software Vendor and Salesforce Consulting partner. He has been inventing cutting-edge enterprise software for more than 13 years. Prior to joining Appirio, he served as a director of R&D for application products at Composite Software, where he led development of data services for Siebel, SAP, and salesforce.com. At webMethods, he helped architect the industry’s first XML-based B2B server.

He lives with his wife and two geriatric cats in San Francisco, California.
Introducing Force.com

This chapter introduces the concepts, terminology, and technology components of the Force.com platform and its context in the broader Platform as a Service (PaaS) landscape. The goal is to provide context for exploring Force.com within a corporate software development organization. If any of the following sentences describe you, this chapter is intended to help.

- You have read about cloud computing or PaaS and want to learn how Force.com compares to other technologies.
- You want to get started with Force.com but need to select a suitable first project.
- You have a project in mind to build on Force.com and want to learn how your existing development skills and process can be leveraged.

This chapter consists of three sections, listed below.

- **Force.com in the Cloud Computing Landscape**: Learn about PaaS and Force.com’s unique features as a PaaS solution.
- **Inside a Force.com Project**: Examine how application development with Force.com differs from other technologies in terms of project selection, technical roles, and tools.
- **Sample Application**: A sample business application is referenced throughout this book to provide a concrete basis for discussing technical problems and their solutions. In this chapter the sample application’s requirements and use-cases are outlined, as well as a development plan, mapped to chapters of the book.

**Force.com in the Cloud Computing Landscape**

Phrases like “cloud computing” and “Platform as a Service” have many meanings put forth by many vendors. This section provides definitions of the terms to serve as a basis for understanding Force.com and comparing it with other products in the market. With
Chapter 1  Introducing Force.com

In this background, you can make the best choice for your projects, whether that is Force.com, another PaaS product, or your own in-house infrastructure.

**Platform as a Service (PaaS)**

The platform is infrastructure for the development of software applications. The functionality of a platform’s infrastructure differs widely across platform vendors, so this section focuses on a handful of the most established vendors. The suffix “as a Service” (aaS) means that the platform exists “in the cloud,” accessible to customers via the Internet. There are many variations on this acronym, including SaaS (Software as a Service), DaaS (Development as a Service), and so forth.

PaaS is a category within the umbrella of cloud computing. “Cloud computing” is a phrase to describe the movement of computing resources away from physical data centers or servers in a closet in your company and into the network, where they can be provisioned, accessed, and deprovisioned instantly. You plug a lamp into an electrical socket to use the electrons in your region’s power grid. It is usually not necessary to run a diesel generator in your basement. You trust that the power company is going to provide that service, and you pay the company as you use the service.

Cloud computing as a general concept spans every conceivable configuration of infrastructure, well outside the scope of this book. The potential benefits are reduced complexity and cost versus a traditional approach. The traditional approach is to invest in infrastructure by acquiring new infrastructure assets and staff or redeploying or optimizing existing investments. Cloud computing provides an alternative.

Many companies provide PaaS products. The following subsections introduce the mainstream PaaS products and include brief descriptions of their functionality. Consult the Web sites of each company for further information.

**Amazon Web Services**

Amazon Web Services refers to a family of cloud computing products. The most relevant to PaaS is Elastic Compute Cloud (EC2). EC2 is a general-purpose computing platform. You can provision virtual instances of Windows or Linux machines at will, loading them with your own custom operating-system image or one prebuilt by Amazon or the community. These instances run until you shut them down, and you are billed for usage of resources such as CPU, disk, and network.

A raw machine with an OS on it is a great start, but to build a business application requires you to install, manage access to, maintain, monitor, patch and upgrade, back up, plan to scale, and generally care and feed in perpetuity an application platform on the EC2 instance. If your organization has the skills to build on .NET, J2EE, LAMP, or other application stacks, plus the OS, database administration, and IT operations experience, EC2’s virtual servers in the cloud could be a strong alternative to running your own servers in-house.

Amazon provides various other products that compliment EC2. These include Simple Queue Service (SQS) for publish-and-subscribe-style integration between applications,
Simple DB for managing schemaless data, and Simple Storage Service (S3), a content repository.

Microsoft Azure Services Platform
At the time of writing this book, Azure is not yet commercially available. Microsoft’s entrance into the PaaS world will likely offer some unique value, particularly for companies seeking to leverage the cost savings of cloud computing but preserve their existing investments in .NET, SQL Server, SharePoint, and other Microsoft products. Azure is marketed as a blend of on-premise software and services in the cloud. It consists of two parts. The first part is Windows Azure, a new operating system that can utilize Microsoft’s data centers for general computation and storage. The second part encompasses three categories of cloud services: .NET Services, SQL Services, and SharePoint Services. These services map to existing Microsoft products for computing, database, and collaboration. The intent is presumably to enable Microsoft’s existing development community to pick and choose how their applications are partitioned between local and hosted resources without costly rewrites or redeployment. Pricing is not yet available, but Microsoft says it will charge for resource consumption, defined as some combination of CPU, network bandwidth, storage, and number of transactions.

Google App Engine
App Engine is a platform designed for hosting Web applications. App Engine is like having an unlimited number of EC2 instances working for you, preconfigured with a distributed data store and Python or Java-based application server, but without the IT operations effort required by EC2. App Engine includes tools for managing the data store, monitoring your site and its resource consumption, and debugging and logging.

App Engine is free for up to 500MB of storage and five million page views per month. Applications requiring more storage or bandwidth can purchase it by setting a maximum daily dollar amount they’re willing to spend, divided into five buckets: CPU time, bandwidth in, bandwidth out, storage, and email.

Force.com
Force.com is targeted toward corporate application developers and independent software vendors. Unlike the other PaaS offerings, it does not expose developers directly to its own infrastructure. Developers do not provision CPU time, disk, or instances of running operating systems. Instead, Force.com provides a custom application platform centered around the relational database, one resembling an application server stack you might be familiar with from working with .NET, J2EE, or LAMP.

Although it integrates with other technologies using open standards such as SOAP and REST, the programming languages and metadata representations used to build applications are proprietary to Force.com. This is unique among the PaaS products but not unreasonable when examined in depth. Force.com operates at a significantly higher level of
abstraction than the other PaaS products, promising dramatically higher productivity to developers in return for their investment and trust in a single-vendor solution.

Force.com is free for developers. Production applications are priced primarily by storage used and number of unique users.

**Facebook**

Facebook is a Web site for connecting with your friends, but it also provides developers with ways to build their own socially aware applications. These applications leverage the Facebook service to create new ways for users to interact while online. The Facebook platform is also accessible to applications not built inside Facebook, exposing the “social graph” (the network of relationships between users) where permitted.

Much of the value of Facebook as a platform stems from its large user base and consistent yet extensible user experience. It is a set of services for adding social context to applications. Unlike Force.com and App Engine, for example, Facebook has no facility to host custom applications.

**Force.com as a Platform**

Force.com is different from other PaaS solutions in its focus on business applications. Force.com is a part of Salesforce.com, which started as a SaaS Customer Relationship Management (CRM) vendor. But Force.com is unrelated to CRM. It provides the infrastructure commonly needed for any business application, customizable for the unique requirements of each business through a combination of code and configuration. This infrastructure is delivered to you as a service on the Internet.

Since you are reading this book, you have probably developed a few business applications in your time. Consider the features you implemented and reimplemented in multiple applications, the unglamorous plumbing, wiring, and foundation work. Some examples are security, user identity, logging, profiling, integration, data storage, transactions, workflow, and reporting. This infrastructure is essential to your applications but expensive to develop and maintain. Business application developers do not code their own relational database kernels, windowing systems, or operating systems. This is basic infrastructure, acquired from software vendors or the open-source community and then configured to meet user requirements. What if you could do the same for your application infrastructure? This is the premise of the Force.com.

The following subsections list differentiating architectural features of Force.com with brief descriptions.

**Multitenancy**

Multitenancy is an abstract concept, an implementation detail of Force.com, but one with tangible benefits for developers. Figure 1-1 shows a conceptual view of multitenancy. Customers access shared infrastructure, with metadata and data stored in the same logical database.

The multitenant architecture of Force.com consists of the following features.
- **Shared infrastructure**: Every customer (or tenant) of Force.com shares the same infrastructure. You are assigned a logical environment within the Force.com infrastructure.

  At first some might be uncomfortable with the thought of handing their data to a third-party where it is co-mingled with that of competitors. Salesforce’s whitepaper on its multitenant technology includes the technical details of how it works and why your data is safe from loss or spontaneous appearance to unauthorized parties.

- **Single version**: There is only one version of the Force.com platform in production. The same platform is used to deliver applications of all sizes and shapes, used by 1 to 100,000 users, running everything from dog-grooming businesses to the Japanese national post office.

- **Continuous, zero-cost improvements**: When Force.com is upgraded to include new features or bug fixes, the upgrade is enabled in every customer’s logical environment with zero to minimal effort required.

  Salesforce can roll out new releases with confidence because it maintains a single version of its infrastructure and can achieve broad test coverage by leveraging tests, code, and configurations from their production environment. You, the customer, are helping maintain and improve Force.com in a systematic, measurable way as a side effect of simply using it. This deep feedback loop between the Force.com and its users is something impractical to achieve with on-premise software.

**Relational Database**

The heart of Force.com is the relational database provided as a service. The relational database is the most well-understood and widely used way to store and manage business data.
data. Business applications typically require reporting, transactional integrity, summarization, and structured search, and implementing those on nonrelational data stores requires significant effort. Force.com provides a relational database to each tenant, one that is tightly integrated with every other feature of the platform. There are no Oracle licenses to purchase, no tablespaces to configure, no JDBC drivers to install, no ORM to wrangle, no DDL to write, no queries to optimize, and no replication and backup strategies to implement. Force.com takes care of all of this for you.

Application Services
Force.com provides many of the common services needed for modern business application development. These are the services you might have built or integrated repeatedly in your past development projects. They include logging, transaction processing, validation, workflow, email, integration, testing, reporting, and user interface.

These services are highly customizable with and without writing code. Although each service can be valued as an individual unit of functionality, there is tremendous value from their unification. All the features of Force.com are designed, built, and maintained by a single responsible party, Salesforce. Salesforce provides documentation for these features as well as support staff on-call, training and certification classes, and accountability to its customers for keeping things running smoothly. This is in contrast to many software projects that end up as a patchwork of open-source, best-of-breed tools and libraries glued together by you, the developer, asked to do more with fewer people, shorter timelines, and cheaper, often unsupported tools.

Declarative Metadata
Almost every customization configured or coded within Force.com is readily available as simple XML with a documented schema. At any point in time, you can ask Force.com for this metadata via a set of Web services. The metadata can be used to configure an identical environment or integrate with a source control system. It is also helpful for troubleshooting, allowing you to visually compare the state of two environments. Although there are a few features of Force.com not available in this declarative metadata form, Salesforce’s stated product direction is to provide full coverage.

Programming Language
Force.com has its own programming language, called Apex. It allows developers to script interactions with other platform features, including the user interface. Its syntax is a blend of Java and database stored procedure languages like T/SQL and can be written using a Web browser or a plug-in to the Eclipse IDE.

Other platforms take a different approach. Google’s App Engine simultaneously restricts and extends existing languages such as Python so that they play nicely in a PaaS sandbox. There are obvious benefits, such as leveraging the development community, ease of migration, and skills preservation. One way to understand Apex is as a domain-specific language. Force.com is not a general-purpose computing platform to run any Java or C# program you want to run. Apex is kept intentionally minimalistic, designed with only the
needs of Force.com developers in mind, built within the controlled environment of Salesforce R&D. Although it won’t solve every programming problem, Apex’s specialized nature leads to some advantages in learning curve, code conciseness, ease of refactoring, and ongoing maintenance costs.

**Force.com Services**

Force.com can be divided into four major services: database, business logic, user interface, and integration. Technically there are many more services provided by Force.com, but these are the high-level categories that are most relevant to new Force.com developers.

**Database**

Force.com is built around a relational database. It allows the definition of custom tables containing up to 500 fields. Fields contain strongly typed data using any of the standard data types, plus rich types such as currency amounts, picklists, and phone numbers. Fields can contain validation rules to keep data clean before it is committed and formulas to derive values like cells in a spreadsheet. Field history tracking provides an audit log of changes to chosen fields.

Custom tables can be related to each other, allowing the definition of complex data schemas. Tables, rows, and columns can be configured with security constraints. Data and metadata is protected against accidental deletion through a “recycling bin” metaphor. The database schema is often modifiable instantly, without manual migration. Data is imported from files or other sources with free tools, and APIs are provided for custom data loading solutions.

Data is queried via a SQL-like language called SOQL (Salesforce Object Query Language). Full-text search is available through SOSL (Salesforce Object Search Language).

**Business Logic**

Apex is the language used to implement business logic on Force.com. It allows code to be structured into classes and interfaces, and it supports object-oriented behaviors. It has strongly typed collection objects and arrays modeled after Java.

Data binding is a first-class concept in Apex, with the database schema automatically imported as language constructs. Data manipulation statements, trigger semantics, and transaction boundaries are also part of the language.

The philosophy of test-driven development is hard-wired into the Force.com platform. Methods are annotated as tests and run from a provided test harness or test API calls. Test methods are automatically instrumented by Force.com and output timing information for performance tuning. Force.com prevents code from being deployed into production that does not have adequate unit test coverage.

**User Interface**

Force.com provides two approaches for the development of user interfaces: Page Layouts and Visualforce. Page Layouts are inferred from the data model, including validation rules,
and then customized using a WYSIWYG editor. Page Layouts feature the standard Salesforce look-and-feel. For many applications, Page Layouts can deliver some or all of the user interface with no development effort.

Visualforce allows developers to build custom user interfaces. It consists of a series of XML markup tags called components with their own namespace. As with JSP, ASP.NET, Velocity, and other template processing technologies, the components serve as containers to structure data returned by the Controller, a class written in Apex. To the user, the resulting Web pages might look nothing like Salesforce, or adopt its standard look-and-feel. There are Visualforce components to express the many types and styles of UIs, including basic entry forms, lists, multistep wizards, Ajax, Flex, mobile applications, and content management systems. Developers can create their own components to reuse across applications.

User interfaces in Visualforce are public, private, or some blend of the two. Private user interfaces require a user to log in before gaining access. Public user interfaces, called Sites, can be made available to anonymous users on the Internet.

### Integration

In the world of integration, more options are usually better, and standards support is essential. Force.com supports a wide array of integration technologies, almost all of them based on industry-standard protocols and message formats. You can integrate other technologies with Force.com using the standard recipe of configuration plus code. Here are some examples.

- **Apex Web Services** allows control of data, metadata, and process from any platform supporting SOAP over HTTP, including JavaScript. This makes it possible to write composite applications that combine Force.com with technology from other vendors in many interesting and powerful ways. Force.com’s Web services API is in its 15th version, and Salesforce supports all 16 versions simultaneously.

- **Business logic** developed in Apex can be exposed as a Web service, accessible with or without a Force.com user identity. Force.com generates the WSDL from your Apex code. Additionally, Force.com converts WSDL to Apex bindings to allow access to external Web services from within the platform.

- **You can create virtual email inboxes on Force.com and write code to process the incoming email. Sending email from Force.com is also supported.**

- **Force.com provides an API for making HTTP requests, including support for client-side certificates, SSL, proxies, and HTTP authentication.** With this you can integrate with Web-based resources, such as Representational State Transfer (REST) or JSON services.

- **Salesforce-to-Salesforce (S2S)** is a publish-and-subscribe model of data sharing between multiple Force.com environments. If the company you need to integrate with already uses Force.com and the data is supported by S2S, integration becomes
a relatively simple configuration exercise. There is no code or message formats to maintain. Your data is transported within the Force.com environment from one tenant to another.

If your requirements dictate a higher-level approach to integration, integration software vendors like Cast Iron Systems and Informatica offer adapters to Force.com to read and write data and orchestrate complex transactions spanning disparate systems.

**Inside a Force.com Project**

This section discusses what makes a Force.com project different from a typical corporate in-house software development effort, starting with project selection. Learn some tips for selecting a project in Force.com's sweet spot. Then examine how traditional technical roles translate to development activities in a Force.com project and how technologies within Force.com impact your product development lifecycle. Lastly, get acquainted with the tools and resources available to make your project a success.

**Project Selection**

Some projects are better suited to implementation on Force.com than others. It is possible to run into natural limits of the PaaS approach or battle against the abstraction provided by the platform. Always strive to pursue projects that play into Force.com strengths. There are no absolute rules for determining this, but projects with the following characteristics tend to work well with Force.com:

- **The project is data-centered, requiring the storage and retrieval of structured data.**
  Structured data is the most important point. Implementing a YouTube-like application on Force.com is not the best idea, since it primarily works with unstructured data in the form of video streams. Force.com supports binary data, so a video-sharing Web site is certainly possible to build. But handling large amounts of binary data is not a focus or core competency of Force.com. A hotel reservation system is an example of a more natural fit.

- **The user interface is composed primarily of wizards, grids, forms, reports.**
  Force.com does not restrict you to these user interface patterns. You can implement any type of user interface, including “rich” clients that run using Flash in the browser, and even full desktop applications that integrate with Force.com via its Apex Web Services API. But to capture the most benefit from the platform, stick with structured, data-driven user interfaces that use standard Web technologies such as HTML, CSS, and JavaScript.

- **The underlying business processes involve email, spreadsheets, and hierarchies of people who participate in a distributed, asynchronous workflow.**
  Standard Force.com features such as workflow, approvals, and email services add a lot of value to these applications. They can be configured by business analysts or controlled in-depth by developers in Apex code.
The rules around data sharing and security are fine-grained and based on organizational roles and user identity.

User identity management and security are deep subjects and typically require high effort to implement in a custom system. With Force.com they are standard, highly configurable components that you can leverage without coding. You can then spend more time thinking though the “who can see what” scenarios rather than coding the infrastructure to make them possible.

The project requires integration with other systems.

Force.com is built from the ground up to interoperate with other systems at all its layers: data, business logic, and user interface. The infrastructure is taken care of, so you can focus on the integration design. Exchange a million rows of data between your SQL Server database and Force.com. Call your Apex services from a legacy J2EE application or vice versa. Add an event to a Google calendar from within your Visualforce user interface. These scenarios and more are fully supported by the platform.

The project manipulates data incrementally, driven by user actions rather than a calendar.

Force.com is a shared resource. Simultaneously there are other customers of varying sizes using the same infrastructure. This requires Force.com to carefully monitor and fairly distribute the computing resources so that all customers can accomplish their goals with a high quality of service. If one customer's application on Force.com was allowed to consume a disproportionate share of resources, other customers' applications would suffer resource starvation. The limitations in place, called governors, prevent too much memory, CPU, disk, or network bandwidth from being concentrated in the hands of any one customer. The platform strongly enforces these governor limits, so the best Force.com applications involve computing tasks that can be split into small units of work.

The data volume is limited, below a few million records per table.

Data volume is important to think about with any system: How large is my data going to grow and at what rate? Force.com consists of a logical single transactional database. There is no analytical data store. Applications that require access to large volumes of data, such as data warehousing and analytics, cannot be built on Force.com. Other software vendors provide solutions to this area, but all involve copying data from Force.com to their own products.

Force.com is not an all-or-nothing proposition. If your project does not fit within these guidelines, you might still want to explore Force.com but in conjunction with other PaaS solutions such as Amazon EC2. Thanks to Force.com's integration capabilities, EC2 and Force.com can be used together as a composite solution, EC2 augmenting Force.com where general-purpose computing is needed.
Team Selection

The best people to staff on Force.com projects might already work at your company. Projects do not require brand-new teams staffed with Force.com experts. With the majority of the platform based in mature technology such as relational databases and Web development, adapting existing teams can be a straightforward task.

Here are some examples of traditional software development roles and how they can contribute to a Force.com project:

- **Business Analyst**
  Substantial Force.com applications can be built entirely by configuration, no computer science background or coding skills required. Salesforce refers to this as “clicks, not code.” Business analysts who are proficient with Microsoft Excel and its macro language, or small-scale databases like Microsoft Access and FileMaker Pro, can get hands-on with the Force.com data model, validation rules, workflows, approval rules, and page layouts.

- **Data Modeler**
  A data model forms the core of a Force.com application. Data modelers can use their existing Entity-Relationship tools and techniques to design the data layer, with some deltas to account for Force.com behavior. Rather than scripts of DDL statements, their work output is Force.com’s metadata XML or manual configuration of the data objects. Data modelers can also design reports and report types, which define data domains available to business users to build their own reports.

- **Database Administrator**
  Many traditional DBA tasks are obsolete in Force.com because there is no physical database to build, monitor, and tune. But a DBA still has plenty of work to do in planning and implementing the Force.com object model. There are objects to define or permissions to configure, and the challenges of data migration are still as relevant in Force.com as in any database-backed system.

- **Database Developer**
  The design of Force.com’s programming language, Apex, has clearly been inspired by stored procedure languages like T-SQL and PL/SQL. Existing database developers can adapt their skills to writing Apex code, particularly when it requires detailed work on the data like triggers.

- **Object-Oriented Analysis and Design Specialist**
  Force.com includes an object-oriented language, and persistent data is represented as objects. With all of these objects floating around, people with skills in traditional techniques like Unified Modeling Language (UML) are valuable to have on your project team. Larger applications benefit from a well-designed object model, and as in any language, designing before writing Apex code can be a real timesaver.
Chapter 1  Introducing Force.com

- **User Interface Designer**
  Force.com supports modern Web standards for creating usable, flexible, and maintainable UIs. UI designers can help by building screen mock-ups, page layouts, and the static portions of Visualforce pages to serve as templates.

- **Web Developer**
  Developers who have built Web applications can quickly learn enough Apex and Visualforce and build similar applications on Force.com, typically with much less effort. Skills in HTML, CSS, JavaScript, or Adobe Flex are needed to build custom Force.com user interfaces.

- **4GL Developer**
  Developers proficient in fourth-generation languages such as Java, C#.NET, and PHP have no problem picking up Apex code. It has the same core syntax as Java, minus the Java-specific libraries.

- **Integration Specialist**
  Force.com is a producer and consumer of Web services and supports REST as well as any integration strategy based on HTTP. An integration expert can design the interaction between systems, define the remote operations, and implement them using Force.com or a dedicated integration product.

- **Quality Assurance Engineer**
  Testing is a critical part of any software project, and on Force.com testing is mandatory before code is deployed to production. A QA engineer can write unit tests in Apex and test plans for security and integration testing. Standard tools like Selenium can be used to automate UI testing.

- **Operations Specialist**
  Although there are no servers or operating systems to manage, larger deployments of Force.com can involve integration with on-premise systems. Single Sign On (SSO) integration and data migration are two common examples. Operations experts can help in this area, as well as with application deployment and Force.com administration tasks such as user maintenance.

**Lifecycle**

The software development lifecycle of a Force.com project is much like an on-premise Web application development project, but with less toil. There are many moving parts in J2EE, .NET, or LAMP projects. Most require a jumble of frameworks to be integrated and configured properly before one line of code relevant to your project is written. In fairly integrated environments like .NET on the Microsoft platform, there are fewer frameworks to be integrated but still plenty of infrastructure code to be written.

This section describes areas of Force.com functionality designed to streamline the development lifecycle and focus your time on the value-added activities related to your application. There are implicit costs and benefits in each of these areas. On the cost side,
there is usually a loss of control and flexibility versus technologies with less abstraction. It is up to you to evaluate these features and judge whether they constitute costs or benefits for your project.

**Integrated Logical Database**

Relational databases are still the default choice for business applications, despite the availability of alternatives like XML and object-oriented databases. The relational model maps well onto business entities, data integrity is easily enforceable, and implementations scale to hold massive amounts of data while providing efficient recall, composition, and modification.

For business applications coded in an object-oriented language, accessing relational databases introduces an impedance mismatch. Databases organize data in terms of schemas, tables, and columns. Programs organize data and logic into objects, methods, and fields. There are many ways to juggle data between the two, none of them ideal. To make matters more complicated, there are many layers of protocol and message needed to transport queries, resultsets, and transactions between the program and the database.

In Force.com, the database tables are called objects. They are somewhat confusingly named because they do not exhibit entirely object-oriented behavior. The name comes from the fact that they are logical entities that act as tables when being defined, loaded with data, queried, updated, and reported on, but are surfaced to programs as first-class objects. There is no mismatch between the way data is represented in code and the way it’s represented in the database. Your code remains consistent and concise whether you are working with in-memory instances of your custom-defined Apex classes or objects from the database. This also enables compile-time validation of programs, including queries and data manipulation statements, to ensure that they adhere to the database schema. This one seemingly simple feature eliminates a whole class of defects that were previously discovered only through unit tests or in production by unfortunate users.

The logical aspect of the database is also significant. Developers have no direct access to the physical databases running in Salesforce’s data centers. The physical data model is a proprietary meta-model optimized for multitenant applications, with layers of caches and fault tolerance, spanning thousands of servers in multiple data centers. When you create an object in Force.com, no corresponding Oracle database table is created. The metadata describing your new table is stored and indexed by a series of physical tables, becoming a unified, tenant-specific vocabulary baked into the platform’s higher-level features. The synergy of integrated, metadata-aware functionality makes Force.com much more than the sum of its features.

**Metadata-Derived User Interface**

As described previously, the definition of your objects becomes the vocabulary for other features. Nowhere is this more evident than in the standard Force.com user interface, commonly referred to as the “native” UI. This is the style pioneered by the Salesforce
CRM: lots of tables, topped with fat bars of color with icons of dollar signs and telescopes, and a row of tabs for navigation. Lots of page refreshes too.

It is worth getting to know the capabilities of native UI even if you have reservations about its appearance or usability. To some it is an artifact of the Precambrian era of Web applications. To others it is a clean-cut business application, consistent and safe. Either way, as a developer you cannot afford to ignore it. The native UI is where many configuration tasks are performed, for features not yet visible to Eclipse and other tools.

If your project’s user interface design is amenable to the native UI, you can build screens almost as fast as users can describe their requirements. Rapid application prototyping is an excellent addition or alternative to static screen mock-ups. Page layouts are descriptions of which fields appear on a page in the native UI. They are automatically created when you define an object and configured with a simple drag-and-drop layout tool.

**Simplified Configuration Management**
Configuration management is very different from what you might be accustomed to from on-premise development. Setting up a development environment is trivial with Force.com. You can provision a new development environment in a few clicks and deploy your code to it using the familiar Eclipse IDE.

When added to your Eclipse IDE or file system, Force.com code and metadata are ready to be committed to an existing source control system. Custom Ant tasks are available to automate your deployments. Sandboxes can be provisioned for testing against real-world volumes of data and users. They are automatically refreshed from snapshots of production data per your request. Force.com’s packaging feature allows you to partition your code into logical units of functionality, making it easier to manage and share with others at your company or in the larger community.

**Integrated Unit Testing**
The ability to write and execute unit tests is a native part of the Apex language and Force.com development environment. Typically a test framework is an optional component that you need to integrate into your development and build process. With the facility to test aligned closely with code, writing and executing tests becomes a natural part of the development lifecycle rather than an afterthought.

In fact, unit tests are required by Force.com to deploy code into production. This applies to all Apex code in the system: user interface logic, triggers, and general business logic. To achieve the necessary 75% test coverage often requires as much if not more code than the actual Apex classes.

To make sure you don’t code yourself into a corner without test coverage, a great time to write tests is while you code. Many development methodologies advocate test-driven development, and writing tests as you code has benefits well beyond simply meeting the minimum requirements for production deployment in Force.com. For example, a comprehensive library of tests adds guardrails to refactoring and maintenance tasks, steering you away from destabilizing changes.
Integrated Model-View-Controller (MVC) Pattern

The goal of the MVC pattern is maintainable user interface code. It dictates the separation of data, visual elements that represent data and actions to the user, and logic that mediates between the two. If these three areas are allowed to collide and the codebase grows large enough, the cost to fix bugs and add features becomes prohibitive.

Visualforce adopts MVC by design. For example, its view components do not allow the expression of business logic and vice versa. Like other best practices made mandatory by the platform, this can be inconvenient when you just want to do something quick and dirty. But it is there to help. After all, quick-and-dirty demos have an uncanny tendency to morph into production applications.

Integrated Interoperability

Force.com provides Web services support to your applications without code. You can designate an Apex method as a Web service. WSDL is automatically generated to reflect the method signature. Your logic is now accessible to any program that is capable of calling a Web service, given valid credentials for an authorized user in your organization. You can also restrict access by IP address or open up your service to guests.

As in other languages, Apex provides you with a WSDL-to-Apex tool. This tool generates Apex stubs from WSDL, enabling you to integrate with SOAP-enabled business processes existing outside of Force.com. Lower-level Apex libraries are also available for raw HTTP and XML processing.

End of Life

Retiring a production application requires a few clicks from the system administrator. Users can also be quickly removed or repurposed for other applications. Applications can be readily consolidated because they share the same infrastructure. For example, you might keep an old user interface online while a new one is being run in parallel, both writing to the same set of objects. Although these things are possible with other technologies, Force.com removes a sizable chunk of infrastructure complexity, preserving more intellectual bandwidth to devote to tackling the hard problems specific to your business.

Tools and Resources

Force.com has a rich developer ecosystem. There are discussion groups for reaching out to the development community on specific subjects, a source-code repository for open-source projects, a Web site called AppExchange where you can browse for free and paid extensions to the platform, services companies to help you plan and implement your larger projects, and Ideas, a site for posting your ideas for enhancing the platform.

The following subsections list some tools and resources that exist to make your projects successful.
Developer Force (http://developer.force.com)
Developer Force is a rich source of information on Force.com. It contains documentation, tutorials, e-books written by Salesforce, a blog, and a Wiki with links to many more resources inside and outside of Salesforce.

Developer Discussion Boards (http://community.salesforce.com)
This is a public discussion forum for the Force.com development community. It is divided into a dozen separate boards by technology area. Users post their questions and problems, gripes, and kudos. Other users in the community contribute answers and solutions, including Salesforce employees. The boards are a great way to build a reputation as a Force.com expert and keep current on the latest activity around the platform.

Ideas (http://ideas.salesforce.com)
If you have a suggestion for improving Force.com or any Salesforce product, visit the Ideas site and post it. Other users in the community can vote for it. If your idea is popular enough, it might be added to the next release of Force.com. Incidentally, Ideas is a reusable component of Force.com, so you can build your own customized idea-sharing sites.

Code Share (http://developer.force.com/codeshare)
Code Share is a directory of open-source code contributions from the Force.com community, with links to the actual source code hosted on Google Code. Salesforce employees have contributed many projects here. Two notable ones are the Facebook Toolkit, a library for integrating with Facebook, and XmlDom, an XML parsing library modeled after one in Java.

Platform Documentation
Salesforce provides documentation through online, context-sensitive help within the Web user interface, as well as HTML and PDF versions of its reference manuals. All documentation can be found at Developer Force.

AppExchange (www.appexchange.com)
AppExchange is a directory of ready-to-install applications developed on Force.com. The applications consist of metadata, such as Visualforce pages and Apex code, deployable into your Force.com environment. Users can rate applications from one to five stars and write reviews. There are many free applications written by Salesforce employees to illustrate new platform features. Commercial applications are also available for trial and purchase. AppExchange is how ISVs distribute their Force.com applications to customers.

Dreamforce
Salesforce has a series of user conferences every year called Dreamforce. San Francisco hosts the largest Dreamforce venue, with thousands attending to participate in training sessions, booths, product demos, keynote speeches, breakout sessions, executive briefings, and, of course, the parties. Dreamforce is a fun way to stay up to date with the technology.
**Systems Integrators**
For deployments including significant numbers of users, integration with other enterprise systems, or complex data migrations, consider contracting the services of a systems integrator. There are systems integrators who have competency with Force.com, SFA, SSS, and other Salesforce products. They include pure-play systems integrators such as Appirio or Model Metrics, as well as general consultancies like Accenture.

**Technical Support**
When you encounter undocumented or incorrect behavior in the system, submit a bug report. If the issue can be described simply, like a cryptic error message, search for it in the discussion groups. In many cases someone else has already run into the same problem before you, posted about it, and attracted the attention of Salesforce employees. If not, the ability to log and track Force.com platform support cases is available in Force.com’s Web user interface.

**Sample Application: Services Manager**
Every following chapter in this book contributes to the construction of a sample application called Services Manager. Services Manager is designed for businesses that bill for their employees’ time. These businesses need accurate accounting of when and where employees are staffed, numbers of hours worked, skills of the employees, project expenses, amounts billed to customers, and so forth. This section describes these features in preparation for later discussions of their design and code.

The goal is not to build a fully functional application for operating a professional services business, but to provide a logically related set of working code samples to accompany the technical concepts covered in this book.

**Background**
Imagine you own a professional services business. The services your company provides could be architecture, graphic design, software, law, or anything with the following characteristics:

- High cost, highly skilled employees
- Complex projects lasting a week or more
- Customers billed at an hourly rate and invoiced monthly
- High cost of acquiring new customers

Your profit comes from the difference between the billing rate and the internal cost of resources. This is typically small, so your process must be streamlined, repeatable, and scalable. To increase profit you must hire more resources and win more customer projects.
Chapter 1  Introducing Force.com

User Roles

The users of the Services Manager application span many roles in the organization. The roles are covered in the following subsections, with a summary of their responsibilities and how they use Services Manager.

Services Sales Representative
Sales reps work with customers to identify project needs and manage the relationship with the customer. Reps use the Sales Force Automation (SFA) product from Salesforce to manage their sales process. In general, they do not use Services Manager directly, but start the process by winning the contract.

Staffing Coordinator
Staffing coordinators manage and schedule resources for projects. When the opportunity is closed, they are notified via email. They then create a project using Services Manager and staff it by matching the availability and skills of resources against the scheduling and skill requirements of the project.

Project Manager
Project managers are responsible for success of projects on a daily basis. They direct and prioritize project activities. They use Services Manager to manage the detailed weekly schedules of their consultants and monitor the health and progress of their projects.

Consultant
The consultant is engaged directly with the customer and is responsible for the project deliverables. In Service Manager, he or she logs time spent on the project, indicates the completion of project milestones, and submits expenses.

Accounts Receivable
Accounts receivable is responsible for invoicing and collecting customers based on work that has been delivered. At the end of each billing cycle, they use Services Manager to generate invoices for customers.

Services Vice President
The VP is responsible for the services P&L and success of the team. Services Manager provides the VP with reports on utilization and other metrics for assessing the team’s overall performance.

Development Plan
The Services Manager sample application is developed incrementally throughout this book, each chapter building on the previous. Every chapter covers a set of technical concepts followed by the relevant Services Manager requirements, design, and implementation.
The goal is to expose you to the abstract technology and then make it practical by getting your hands dirty on the sample application.

The following list names the remaining chapters in this book, with brief descriptions of the features of Services Manager to be covered.

- Chapter 2, "Database Essentials": Design and create the database and import data.
- Chapter 4, "Additional Database Features": Define fields for reporting and make a subset of data accessible offline.
- Chapter 5, "Business Logic": Build triggers to validate data and unit test them.
- Chapter 6, "Advanced Business Logic": Write services to generate email notifications based on user activity.
- Chapter 7, "User Interfaces": Construct a custom user interface for tracking the skills of consultants.
- Chapter 8, "Advanced User Interfaces": Enhance the skills user interface with Ajax.
- Chapter 9, "Integration": Calculate and transmit corporate performance metrics to a fictional industry benchmarking organization.
- Chapter 10, "Advanced Integration": Develop a Java program to update Force.com with information from a human resources database.
- Chapter 11, "Additional Platform Features": Build a custom dashboard component to visualize the geographic distribution of consultants on projects.

Summary

This chapter has introduced you to Force.com, explained how it differs from other PaaS technologies and what infrastructure it’s designed to replace, and given guidelines for its use on your projects. Here are a few thoughts to take away from this chapter.

- Force.com is a PaaS uniquely designed to make business applications easy to build, maintain, and deliver. It consists of database, business logic, user interface, and integration services, all of them interoperable and interdependent, accessible through configuration or code.
- The most suitable applications for implementation on Force.com operate primarily on structured data. Traditional software development roles are still relevant in the Force.com world, particularly Web and client/server developers. Data modeling takes on a new importance with the platform, as data objects are tightly integrated with the rest of the technology stack, and unit testing is mandatory.
- Services Manager is the sample application built on throughout this book. It’s designed to serve companies in the professional services space, those selling projects to customers and billing them for the time of its highly skilled employees.
This page intentionally left blank
## Index

### Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4GL developers</td>
<td>12, 148</td>
</tr>
<tr>
<td>() operator</td>
<td>122</td>
</tr>
<tr>
<td>+ (addition operator)</td>
<td>121</td>
</tr>
<tr>
<td>&amp; (AND operator)</td>
<td>121</td>
</tr>
<tr>
<td>&amp;&amp; (AND operator)</td>
<td>122</td>
</tr>
<tr>
<td>- (arithmetic negation operator)</td>
<td>121</td>
</tr>
<tr>
<td>= (assignment operator)</td>
<td>121</td>
</tr>
<tr>
<td>!= (comparison operator)</td>
<td>121</td>
</tr>
<tr>
<td>&gt; (comparison operator)</td>
<td>121</td>
</tr>
<tr>
<td>&gt;= (comparison operator)</td>
<td>121</td>
</tr>
<tr>
<td>&lt; (comparison operator)</td>
<td>121</td>
</tr>
<tr>
<td>&lt;= (comparison operator)</td>
<td>121</td>
</tr>
<tr>
<td>== (comparison operator)</td>
<td>121</td>
</tr>
<tr>
<td>/ (division operator)</td>
<td>121</td>
</tr>
<tr>
<td>=== (equality operator)</td>
<td>122</td>
</tr>
<tr>
<td>? (if/then/else operator)</td>
<td>122</td>
</tr>
<tr>
<td>!== (inequality operator)</td>
<td>122</td>
</tr>
<tr>
<td>! (logical negation operator)</td>
<td>121</td>
</tr>
<tr>
<td>* (multiplication operator)</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>(OR operator)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;&gt; (signed shift right operator)</td>
<td>121</td>
</tr>
<tr>
<td>&lt;&lt; (signed shift left operator)</td>
<td>121</td>
</tr>
<tr>
<td>+ (string concatenation operator)</td>
<td>121</td>
</tr>
<tr>
<td>&gt;&gt;&gt; (unsigned shift right operator)</td>
<td>121</td>
</tr>
<tr>
<td>+++ (unsigned shift right operator)</td>
<td>121</td>
</tr>
<tr>
<td>++ (unary increment operator)</td>
<td>121</td>
</tr>
<tr>
<td>-- (unary decrement operator)</td>
<td>121</td>
</tr>
<tr>
<td>^ (XOR operator)</td>
<td>121</td>
</tr>
</tbody>
</table>

### Abstract methods

abstract methods, 149-150

### Accept button (Visualforce pages)

Accept button (Visualforce pages), 225
access modifiers
Apex classes, 147
methods, 148
accessibility fields, 64, 68, 80
accounts
ID population, 57
receivable, Services Manager application, 18
ACM (Advanced Currency Management), 353-354
action methods, custom controllers, 204-205
actionFunction component, 247
actionPoller component, 248
actions, Visualforce
components, 213-214
JavaScript events, 249
JavaScript functions, 247
partial page refreshes, 246-247
status, 250-251
timed events, 248
actionStatus component, 250-251
facets, 250
with JavaScript, 251
actionSupport component, 249, 271
active workflow rules, 334
addition (+) operator, 121
administration
Connect Offline, 100-102
permissions, 66
Adobe Flex, 258
disadvantages, 258
Flex Builder, 259
FlexDemo project, 260
Internet Explorer support, 263-264
MXML code, 260-261
static resource, 261
Visualforce page, 262
toolkit, 259
Advanced Currency Management (ACM), 353-354
Ajax, Visualforce support
actions
status, 250-251
as JavaScript functions, 247
JavaScript events, 249
partial page refreshes, 246-247
timed events, 248
Amazon Web Services, 2-3
analytics
dashboards, 345
reports, 343
creating, 343-344
custom types, 344
running, 344
snapshots, 346-347
AND operator (&), 121
AND operator (&&), 122
anonymous benchmarking (Services Manager application), 293, 296-298
anti-joins (SOQL), 166
Apex, 6, 112
arrays, 122-123
Batch, 128
classes, 143
access modifiers, 147
constructors, 145-146
information hiding, 147-148
inheritance, 149-150
initializers, 146-147
inner, 147
methods, 144
properties, 145
user-defined, 143
variables, 144
code
development, 113
listing, 29
collections, 122
arrays, 123
Lists, 122-123
Maps, 124
Sets, 124
control logic, 124
asynchronous execution, 127-128
conditional statements, 125
exception statements, 126-127
loops, 125
recursion, 127
custom Web services, 320
  administrative rights, 320
calling, 322-323
code example, 322
definition, 321
governor limits, 321
database integration, 130
deleting/undeleting records, 139
inserting records, 137
persisting records, 137-139
queries, 132. See also SOQL
records as objects, 130-132
relationships, 131
security, 142
triggers, 139-141
updating records, 138
upserting records, 138
debugging, 151
debug method, 154
Finest log level, 154
logs, monitoring, 152
dynamic, 187
database queries, 188-189
schema metadata, 189-191
capsulation, 143
future methods, 128
generating from WSDL, 289-290
governor limits, 129
HTTP integration, 290-293
inheritance, 149-150
interfaces, 148-149
managed sharing, 174-176
  objects, 175-176
  rules, 176-180
operators, 121-122
overview, 112
polymorphism, 150-151
receiving email, 186
  class, creating, 186
  configuring, 185
  inbound email processing, 187
  personalizing, 185
record submissions for approval, 342
scope, 112
sending email, 181
  attachments, 184
  blind carbon copies, 184
  carbon copies, 184
  Documents, 184
  mass emails, 183
  methods, 184-185
  organization-wide email address
    unique identifiers, 185
  replying, 184
  sender display names, 184
  signatures, 184
  single messages, 181-182
  templates, 182
  tracking activity, 185
  transactional behavior, 184
SOQL queries, integrating, 135
SOSL, 168-169
testing, 154
governor limits, 154
running, 155
test methods, writing, 154
Test Runner view
  (Force.com IDE), 116
transaction processing
  DML database methods, 170-171
  record locking, 173-174
  savepoints, 171-173
variables, 117-120
  case sensitivity, 117
  constants, 118
  data types, 117-119
  dates to strings conversions, 120
  declaring, 117
  enums, 118
  names, 117
  rounding numbers, 119
  string to date conversions, 120
Web services, calling, 289-290

APIs

  enabled permissions, 310
Enterprise, 314
  creating records, 317-318
deleting/undeleting records, 319
limits, 320
record bulk modifications, 319
record retrieval, 315–316
updating records, 318
faults, 314
Metadata, 323
creating objects in Java, 324–325
file-based services, 323
object–based services, 324
overview, 323–324
Web services, 304–306
App Builder Tools
custom object metadata, 32
Services Manager data model, creating
application definition, 50
assignment custom objects, 53
project custom objects, 51–52
relationships, 54–55
resource custom objects, 52
skill custom objects, 53
timecard custom objects, 54
App Engine (Google), 3
AppExchange Web site, 16
applications
errors, 314
Hello World, 116, 199–201
logos, 50
retiring, 15
services, 6
Services Manager. See Services Manager application
Approval History related lists, 337
approval processes, 337
Approval History related list, 337
configuring, 340
creating, 340
defining, 337
diagram, 341
field update action, 340
ProcessInstance objects, 341
ProcessInstanceHistory object, 341
record submissions, 342
requests, 337
retrieving, 341
timecard record example, 337
viewing approved records, 337
architecture
security, 63
Visualforce, 196
architecture of Force.com
application services, 6
declarative metadata, 6
multitenancy, 4–5
programming language, 6
relational databases, 5
arithmetic negation operator (-), 121
arrays
Apex, 122
creating, 123
mixed array and list syntax, 123
assert method, 154
assignment operator (=), 121
assignments
custom objects, 53
logical data model, 45
asynchronous actions. See Ajax
asynchronous execution (Apex), 127–128
asyncMethod method, 128
atomic data types, 117
attachments (email), 184
attributes
Apex classes, 145
controller, 209
opt_allOrNone, 170
reRender, 246
standardController, 209
view components, 209
authentication
delegated WSDL, downloading, 360
users, 267
auto number fields, 38
automatic properties, 145
Azure, 3
Batch Apex, 128
batch processing, database triggers, 140
batch size (queries), 316
BenchmarkWS class, 297
blind carbon copies (email), 184
blob data types, 118
Boolean data types, 117
break keyword, 126
Briefcase configuration
  (Connect Offline), 100
browsing data, 39
bulk modifications, records, 319
business
  analysts, 11
  logic services, 7
  units
    collaboration, testing, 86-87
    security, 75, 78
buttons
  custom objects, 35
  customizing, 226
  standard, 224-225
C
  logging in, 312
  query batch sizes, 316
  records
    creating, 318
    retrieval, 315
  Web services clients, 309-310
carbon copies (email), 184
case sensitivity (Apex variables), 117
chaining constructors, 146
child relationship metadata, 190
child-to-parent relationships, 135
classes
  Apex, 143
    access modifiers, 147
    constructors, 145-146
    information hiding, 147-148
    inheritance, 149-150
    initializers, 146-147
    inner, 147
    methods, 144
    properties, 145
    user-defined, 143
    variables, 144
  BenchmarkWS, 297
Crypto, 291
CustomWS, 322
EncodingUtil, 291
exception, 126
Http, 290
HttpRequest, 290
HttpResponse, 290
TimecardManager
  creating, 156
  unit testing, 158
YahooGeocode, 293
clients
  logical data model, 42
  Web services, generating, 306-310
Clone button (Visualforce pages), 225
cloud computing
  defined, 2
  PaaS, 2
    Amazon Web Services, 2-3
    Facebook, 4
    Force.com, 3
    Google App Engine, 3
    Microsoft Azure, 3
Code Share, 16
collections (Apex), 122
  Arrays, 123
  Lists, 122-123
  Maps, 124
  Sets, 124
column to field mapping, 58
commandButton component, 213
commandLink component, 213
communication errors, 231
CompareSkillsComponent component, 271
classification operators, 121
composition (Visualforce), 253-254
conditional statements (Apex), 125
configuring
  analytic snapshots, 346
  approval processes, 340
  Connect Offline
    Briefcase configuration, 100
data sets, defining, 101-102
currency exchange rates, 350
custom labels, 349
dashboards, 345
delegated SSO, 361
development lifecycles, 14
federated SSO, 355–356
fields
  accessibility, 80
  history tracking, 97
inbound email processing, 185
object permissions on profiles, 66
organization-wide defaults, 71, 80
outbound messaging, 274–276
delivery status, 276
message definition, 276
workflow rule, 275–276
record types for profiles, 93
shared objects (S2S), 281, 284
selecting fields to publish, 283
selecting objects to publish, 283
subscriptions, 284
sharing rules, 82
Visualforce security, 230

Connect Offline, 100
administration, 100–102
configuring
  Briefcase configuration, 100
  data sets, defining, 101–102
  conflict resolution, 103
desktop client, 102
detail record example, 102
logging in, 102
sending changes back to
  Force.com, 102
  Services Manager application staffing, 107–108
connections (S2S), 280–281
constants, 118, 144
constructors (Apex classes), 145–146
  chaining, 146
  declaring, 145
consultants
  profiles, testing, 85
  Services Manager application, 18
contact ID population, 57

containsKey method, 124
continue keyword, 126
control logic (Apex), 124
  asynchronous execution, 127–128
  conditional statements, 125
  exception statements, 126–127
  loops, 125
  recursion, 127
controller attribute, 209
controllers
  ProjectMap Visualforce page, 366–367
  Services Manager Skills Matrix, 269
  Skills Matrix (Services Manager application), 236–242, 269
Visualforce, 196, 201
  custom, 203–205
  extensions, 206
  Services Manager application, 295–301
  standard, 201–203
converting
data types, 118–119
dates to strings, 120
strings to dates, 120
cost variable, 145
creating
  analytic snapshots, 346
  approval processes, 340
arrays, 123
  custom objects, 33
  buttons/links, 35
  object definitions, 33–35
  page layouts, 36
  search layouts, 36
  standard fields, 35
  triggers, 35
  validation rules, 35
  custom tabs, 39
databases
  Force.com data model, 47, 50
  logical data model, 41–46
date, datetime, time datatypes, 120
  fields, 36–37, 340
  lists, 122
profiles, 78-79
Project Map dashboard, 362
records, 39, 317-318
relationships, 54-55
reports, 343-344
SAML assertions, 357
static resources, 252
tags, 99
test methods, 154
TimecardManager class, 156
users, 84
Crypto class, 291
CSV files, exporting, 56
currencies (multiple), 350-353
ACM, 353-354
dated currency rates, retrieving, 354
exchange rates, configuring, 350
records, setting, 350
SOQL conversions, 352
support cases, logging, 350
viewing, 352
Visualforce, 352
CurrencyIsoCode field, 350
Currently Assigned Formula field, 53
custom objects
assignment, 53
creating, 33
buttons/links, 35
object definitions, 33-35
page layouts, 36
search layouts, 36
standard fields, 35
triggers, 35
validation rules, 35
projects, 51-52
records, creating, 39
resource, 52
skill, 53
timecard, 54
tools, 31-32
Views, 41
CustomWS class, 322
customers, logical data model, 42
customizing
controllers, 203-205
action methods, 204-205
data exposure, 203
fields, 23
labels, 349
profiles, 66
report types, 344
tabs, 39
Visualforce
components, 255-256
pages, 226-227
Web services, 320
administrative rights, 320
calling, 322-323
code example, 322
definition, 321
governor limits, 321
Dashboards, configuring, 345
dashboards (Services Manager Project Map), 362
controller, 366-367
creating, 362
defining, 367
GoogleMultiMap component, 364-365
Visualforce page, 366
data
browsing, 39
custom object tools, 32
entering, 39
entities
assignments, 45
client, 42
customer, 42
project, 43
resources, 44
skills, 46
timecards, 46
exposing, 203
importing, 55
preparations, 56-57
process, 58-59
verification, 59-60
integrating, 28
  object-relational mapping, 29
user interfaces, generating, 30
  Web services API, 29
  XML metadata, 30
modelers, 11
relationships, 25
sets, defining, 101-102
types
  Apex variables, 117-119
  rich, 24
Visualforce components, 210
  metadata-aware, 210
  primitive, 211
  repeating, 212
volume, 10
Data Loader, importing data, 55
  preparations, 56-57
  process, 58-59
  verification, 59-60
Data Manipulation Language. See DML
databases
  administrators, 11
  Apex integration, 130
    database records as objects, 130-132
    queries, 132. See also SOQL
    relationships, 131
  creating
    Force.com data model, 47, 50
    logical data model, 41-46
developers, 11
fields, 23
  auto number, 38
  creating, 36-37
  custom, 23
  formula, 24, 38
  history tracking, 24
  logical, 23
  relationship, 37-38
  rich data types, 24
  roll-up summary, 38
  standard, 23
  unique identifiers, 23
  validation rules, 23
integrated logical, 13
objects, 13, 21
  database tuning, 22
  fields, 21
  logical, 22
  operational tasks, 22
  undelete functionality, 22
queries
  Apex, 132. See also SOQL
  languages, 25-28
records
  deleting, 139
  inserting, 137
  persisting, 137-139
  undeleting, 139
  updating, 138
  upserting, 138
relational, 5
relationships, 25
security
  Apex, 142
  funnel, 64-65
  object-level. See object-level
  security
  overview, 63-65
  record-level. See record-level
  security
services, 7
  triggers, 139
    batch processing, 140
    defining, 139-140
    error handling, 141
    governor limits, 141
tuning, 22
dataList component, 212
dataList view component, 208
dataTable component, 212
date data types, 117
dated currency rates, retrieving, 354
dates, 120
datetime data types, 117, 120
DE (Developer Edition), 31
debug method, 154
debugging
Apex, 151
  debug method, 154
  Finest log level, 154
  logs, monitoring, 152
  workflow rules, 335
decimal data types, 118
declarative metadata, 6
declaring
Apex variables, 117
  constructors, 145
  future methods, 128
  inner classes, 147
  interfaces, 148
  methods, 144
  variables, 144
delegated administration sharing
  reasons, 73
delegated SSO, 359-361
  configuring, 361
  errors, 362
  profiles, 361
  sample code, 360-361
  WSDL authentication, downloading, 360
Delete button (Visualforce pages), 225
Delete statements, 139
DeleteResult object, 319
deleting
  database records, 139
  records, 91, 319
delivery status (outbound messages), 276
dependent fields, 90
  alternatives, 90
  picklist values, 90
  Services Manager application skill types, 104
dependent picklists, 104
DER (Distinguished Encoding Rules), 358
DescribeFieldResult object, 190
desktop client (Connect Offline), 102
detail component, 220
detail records (Connect Offline), 102
details page (Force.com sites), 262
developer discussion boards, 16
Developer Edition (DE), 31
Developer Force Web site, 16, 31
development. See also listings
  Apex code, 113
  lifecycles, 12
  application retirement, 15
  configuration management, 14
  integrated logical databases, 13
  interoperability, 15
  metadata-derived user interfaces, 13-14
  MVC pattern, 15
  unit testing integration, 14
  Services Manager application, 18
  Visualforce, 198-199
Distinguished Encoding Rules (DER), 358
division operator (/), 121
DML (Data Manipulation Language), 170
  database methods, 170-171
  database records, persisting, 137-139
  deleting, 139
  Insert statement, 137
  undeleting, 139
  Update statement, 138
  Upsert statement, 138
DmlException class, 126
do while loops, 125
documents, emailing, 184
domain names, 265
double data types, 118
Dreamforce, 16
dynamic Apex, 187
  database queries, 188-189
  schema metadata, 189
  child relationship, 190
  field, 190
  object, 189
  picklist, 191
  record type, 191
 Enterprises API, 314
 limits, 320
 records
 bulk modifications, 319
 creating, 317–318
 deleting/undeleting, 319
 retrieval, 315–316
 updating, 318
 Web services, 304
 entities
 assignments, 45
 client, 42
 customer, 42
 project, 43
 resources, 44
 skills, 46
 timecards, 46
 entry points (governor limits), 129
 enums (Apex variables), 118
 equality operator (===), 122
 errors
 communication, 231
 data type conversions, 119
 handling
 database triggers, 141
 delegated SSO, 362
 Visualforce, 230–231
 Web services, 314
 events
 JavaScript, 249–251
 timed, 248
 Excel
 Connector, 33
 formula for populating
 account IDs, 57
 formula for populating contact IDs, 57
 exception statements (Apex), 126–127
 exchange rates, configuring, 350
 Execute Anonymous view
 (Force.com IDE), 116
 Executive VP profiles, 86
 exporting CSV files, 56
extending Visualforce, 257
Adobe Flex
  disadvantages, 258
  Flex Builder, 259
  FlexDemo project, 260-264
toolkit, 259
JavaScript libraries, 257-258
extensions (controllers), 206
external sharing related lists, 286

F
Facebook, 4
facets (actionStatus component), 250
feature opt-ins, 39
federated SSO, 354-359
  configuring, 355-356
  SAML assertions, 356-357
    creating, 357
    PEM conversions, 358
    signing/encoding, 358
    testing, 359
fields, 23
  accessibility, 64, 80
  auto number, 38
  creating, 36-37
  CurrencyIsoCode, 350
custom, 23
custom objects, 35
dependent, 90
  alternatives, 90
  picklist values, 90
  Services Manager application skill types, 104
federated SSO, 355
filters, 101
formula, 24, 38
  Currently Assigned, 53
  Total Hours, 54
  Years of Experience, 52
history tracking, 24, 97-98
logical, 23
metadata, 190
objects, 21
relationship, 37-38
rich data types, 24
roll-up summary, 38, 95-97
  Services Manager application
    project reporting, 104, 107
    summary calculation, 96
security, 67-69
sharing objects, 175
SOSL specifications, 168
standard, 23
unique identifiers, 23
update actions, creating, 340
validation rules, 23
visibility, 93
files
  based services, 323
  CSV, exporting, 56
filters
  field, 101
  record ownership, 101
  SOQL records, 133-134
final keyword, 144
Finest log level, 154
Flex (Adobe)
  disadvantages, 258
  Flex Builder, 259
  FlexDemo project, 260
    Internet Explorer support, 263-264
    MXML code, 260-261
    static resource, 261
    Visualforce page, 262
toolkit, 259
FlexDemo project, 260
  Internet Explorer support, 263-264
  MXML code, 260-261
  static resource, 261
  Visualforce page, 262
Force.com
  architecture
    application services, 6
    declarative metadata, 6
    multitenancy, 4-5
    programming language, 6
    relational databases, 5
Connect Offline, 100-103
IDE, 113
  custom object metadata, 32
debug logs, 152
installing, 113
perspective, 113
projects, 114
Scheme Explorer, 115
Services Manager application
timecard validation, 156
views, 115-116
Visualforce development, 199
integration solutions, 273
calling Web services from Apex
code, 289-290
HTTP, 290-293
outbound messaging, 274-279
S2S, 279-288
outbound requests, controlling, 289
overview, 3-4
services, 7
  business logic, 7
database, 7
  integration, 8-9
  user interface, 7
sites, 264
  details page, 267
domain names, 265
enabling, 265
Login Settings page, 267
main page, 266
pages, adding, 266
security, 266
  user authentication, 267
versions, 5
formatting datetime data types, 120
formula fields, 24, 38
  Currently Assigned, 53
  Total Hours, 54
  Years of Experience, 52
forwarding records, 286
funnel of security, 64-65
future methods, 128

G–H

generating user interfaces, 30
getCost method, 145
getDescribe method, 190
global access modifier, 148
Google App Engine, 3
GoogleMultiMap component, 364-365
Governor Limits, 28
  Anonymous Block, 129
  Apex, 129
custom Web services, 321
database triggers, 141
database triggers, 141
easy points, 129
incoming email messages, 186
resource types, 129
test methods, 154
Visualforce, 232
Group object query, 178
group operator (), 122
groups
  search, 168
  user, 70
handleInboundEmail method, 185
handling
events
  database triggers, 141
delegated SSO, 362
  Visualforce, 230-231
Web services, 314
exceptions, 126
Hello World application, 116, 199-201
history tracking, 24
HTTP, Force.com integration, 290-293
Http class, 290
HttpRequest class, 290
HttpResponse class, 290
I
ID data types, 118-119
IDE (Force.com), 113
  Apex Test Runner view, 116
custom object metadata, 32
debug logs, 152
Execute Anonymous view, 116
installing, 113
perspective, 113
Problems view, 115
projects, 114
Schema Explorer, 115
Services Manager application timecard validation, 156
Visualforce development, 199
Ideas Web site, 16
if/then/else operator (?), 122
iframe component, 214
images, 50, 214
implementing Services Manager application, 326
Java example, 326-330
security, 78
field accessibility, configuring, 80
organization-wide defaults, configuring, 80
profiles, creating, 78-79
role hierarchy, 81
sharing rules, 82
Skills Matrix
basic, 234
controllers, 236-242
full, 235
Visualforce page, 238-240
implicit joins, 26
Import Wizard, 33
importing
data, 55
preparations, 56-57
process, 58-59
verification, 59-60
relationships, 56
inactive workflow rules, 334
InboundEmailHandler interface, 185
inbound email processing, 187
class, creating, 186
configuring, 185
personalizing, 185
include component, 253
includeScript component, 214
inclusion (Visualforce), 253
inequality operator (!==), 122
information hiding (Apex), 147-148
inheritance (Apex), 149-150
initializers (Apex classes), 146-147
inner classes (Apex), 147
inner joins (SOQL), 162-163
inputCheckbox component, 211
inputField component, 210
inputFile component, 211
inputHidden component, 211
inputSecret component, 211
inputText component, 211
inputTextArea component, 212
Insert statements, 137
inserting database records, 137
installing Force.com IDE, 113
instance initializers, 146
instanceof keyword, 151
integers, 151
integration
calling Web services from Apex code, 289-290
data, 28
object-relational mapping, 29
user interfaces, generating, 30
Web services API, 29
XML metadata, 30
HTTP, 290-293
interoperability, 15
logical databases, 13
MVC, 15
outbound messaging, 274
configuring, 274-276
limits, 274
Web service, creating, 276-279
S2S, 279
connecting, 280-281
record sharing, 284-288
shared objects, configuring, 281, 284
services, 8-9
Services Manager application, 326
Java integration implementation sample, 327–330
JSON file format sample, 330
specialists, 12
unit testing, 14
Web services, 304
APIs, 304–306
C# clients, generating, 309–310
clients, generating, 306, 310
Enterprise API. See Enterprise API error handling, 314
Java clients, generating, 307–308
logging in, 310–312
Metadata API, 323–324
overview, 304
security, 305
Services Manager application, 296–298
SOAP data types, 313
versions, 305

interfaces
Apex, 148–149
InboundEmailHandler, 185
native user
custom buttons/links, 226
custom tabs, 227
page layout, 225–226
standard buttons, 224–225
standard pages, 222–224
Visualforce development, 198

international organizations
multilingual support, 348–349
multiple currencies, 350–353
ACM, 353–354
dated currency rates, retrieving, 354
exchange rates, configuring, 350
records, setting, 350
SOQL conversions, 352
support cases, logging, 350
viewing, 352
Visualforce, 352

interoperability, 15
IP whitelisting, 311
isDefaultRecordTypeMapping object, 191

J–K

Java
API SimpleDateFormat pattern, 120
logging in, 312
objects, creating, 324–325
query batch sizes, 316
records, 315–317
Services Manager integration implementation, 327–330
Web services clients, 307–308

JavaScript, Visualforce
events, 249–251
functions, 247
libraries, 257–258

job function security, 75

joins (SOQL)
anti-joins, 166
inner, 162–163
outer, 162
semi-joins, 164–166

JSON file format, 330
Jump Start Wizard, 340

keyset method, 124

keywords
abstract, 149
final, 144
instanceof, 151
loops, 126
static, 144
super, 149
this, 145
virtual, 149
with sharing, 142

L

labels, custom, 349

languages
Apex. See Apex
DML
database methods, 170–171
database records, persisting, 137–139
SAML assertions, 356-357
  creating, 357
PEM conversions, 358
signing/encoding, 358
testing, 359
SOQL, 25-28
  anti-joins, 166
  Apex database queries, 132
approvals, retrieving, 341
currency conversions, 352
dated currency rates,
  retrieving, 354
dynamic queries, 188
Governor Limits, 28
implicit joins, 26
inner joins, 162-163
multiple object queries, 134-135
multi-select picklists, 166
nested resultsets, 26
no functions allowed, 27
object relationships, 135-136
outer joins, 162
queries in Apex, 135
query results, sorting, 134
record limits, 134
records, filtering, 133-134
records, retrieval, 315-316
sample query, 26
semi-joins, 164-166
statements, 132
SOSL, 25, 28
  Apex, 168-169
  field specifications, 168
  overview, 167-168
  queries, 167, 189
  record limits, 168
  search groups, 168
WSDL
  Apex, generating, 289-290
delegated authentication,
  downloading, 360
outbound messaging, 276-279
layout, Visualforce pages, 225-226
listing profiles, 67

**lifecycles (development), 12**
  application retirement, 15
  configuration management, 14
  integrated logical databases, 13
  interoperability, 15
  metadata-derived user interfaces,
    13-14
  MVC pattern, 15
  unit testing integration, 14

**limits**
  future methods, 128
  outbound messaging, 274
  records
    Connect Offline configuration, 102
    SOSL, 168
  savepoints, 171
  sending email (Apex), 181
  SOQL records, 134
  Web services APIs, 305-306

**links, customizing, 35, 226**
list loops, 126
list standard page, 222
listings
  Apex, 29
    abstract methods, 150
    arrays, creating, 123
    automatic properties, 145
    conditional statements, 125
    constructor chaining, 146
    constructor declaration, 145
    database records, creating, 130
    database relationships, 131
data type conversions, 118-119
datetime formatting, 120
DML database method, 171
datatime handling, 127
future methods, declaring, 128
inner class declaration, 147
instance initializer, 146
instanceof keyword, 151
interface declaration, 148
lists, creating, 122
Maps, 124
method declarations, 144
method overloading, 150
mixed array and list syntax, 123
nested lists, 123
properties, 145
read-only fields, 131
read-only/write-only properties, 145
receiving email, 186
record locking, 174
recursion with unsupported depth, 127
rounding numbers, 120
savepoints, 172
sending email with SingleEmailMessage, 181
sending email with template, 182
sending mass email, 183
Sets, 124
sharing rules, 179
simple statement, 117
subclass, 149
test method, 154
user-defined classes, 143
variable declarations, 144
variables with traditional accessor methods, 145
child relationship metadata, 190
CSV import file, 57
custom Web services, 322
database records
deleting, 139
inserting, 137
undeleting, 139
updating, 138
upserting, 138
database triggers
batch processing, 141
defining, 140
date, datetime, time datatypes,
creating, 120
delegated SSO, 360-361
enums, defining, 118
Excel formulas for populating IDs, 57
federated SSO
creating SAML assertions, 357
OpenSAML initialization, 357
PEM conversions, 358
signing/encoding SAML assertions, 358
testing SAML assertions, 359
field metadata, 190
formula fields, 24
Currently Assigned, 53
Total Hours, 54
Years of Experience, 52
Hello World application, 116
logging in, 312
Metadata API, object creation, 324
object metadata, 189
PHP implementation, 278
picklist metadata, 191
records
creating, 317-318
retrieval, 315-316
submissions for approval, 342
type metadata, 191
REST services
calling, 291
invoking, 291
testing, 293
Services Manager application controller sample code, 299-301
e-mail notification triggers on Timecard object, 192
GoogleMultiMap component, 364-365
Java integration implementation, 327-330
JSON file format, 330
ProjectMap, 366-367
Skills Matrix, 269-271
Skills Matrix actionSupport component, 271
Skills Matrix controller, 236-238
Skills Matrix Visualforce page, 239-240
Skills Matrix YUI overlay support, 270
TimecardManager class, 156
unit tests, 158, 240
validateTimecard trigger, 156
Visualforce page sample code, 301-302
Web service integration, testing, 297-298
skill validation rule, 53
SOQL
  anti-joins, 166
  Apex, 168
  approvals, retrieving, 341
  child-to-parent relationship, 135
  currency conversions, 352
  dated currency rates, retrieving, 354
  dynamic queries, 188-189
  Group object query, 178
  inner joins, 162
  multi-select picklist, 167
  outer join, 162
  parent-to-child query, filter on child, 164
  parent-to-child relationships, 135
  Project Share object query, 177
  queries, 26-28
  queries in Apex, 136
  query results, sorting, 134
  record filtering, 133
  record limits, 134
  relationship queries, 27
  semi-joins, 164
  statements, 132
SQL relationship queries, 26
validation rule example, 23
Visualforce
  action status, 250
  actionFunction component, 247
  actionPoller component, 249
  actionStatus component, 250-251
  actionSupport component, 249
  commandButton component, 213
  component reference in JavaScript, 258
  controller extensions, 207
  custom component, 256
  custom component to render Google Map, 255
  custom controller action methods, 205
  custom controller data exposure, 204
  error handling, 232
  Flex object embedding, 264
  FlexDemo Internet Explorer support, 263-264
  FlexDemo MXML code, 260-261
  FlexDemo controller extension method, 263
  FlexDemo Visualforce page, 262
  Force.com-styled components, 215
  Hello World, 199-200
  include component, 253
  inputField component, 210
  outputField component, 210
  partial page refreshes, 246
  record-level security, 229
  repeat component, 213
  sample controller and page, 220-222
  standard controller with multiple records, 202
  standard controller with single record, 201
  templates, 254
  unit tests, 233
  view component sample, 208
XML metadata, 30
lists, 122-123
  creating, 122
  mixed array and list syntax, 123
  nesting, 123
listViews component, 218
locking records, 173-174
logging in, 31, 310
  API enabled permissions, 310
  Connect Offline, 102
  IP whitelisting, 311
  login Web service, 311
  security tokens, 311
  troubleshooting, 312
logical data models (Services Manager application), 41-46
  assignments, 45
  clients, 42
  customers, 42
  projects, 43
  resources, 44
  skills, 46
  timecards, 46

logical negation operator (!), 121

Login Settings page (Force.com sites), 267

log Web service, 311

logs
  debug, 152
  finest log level, 154

long data types, 118

Lookup relationships, 38, 49

loops (Apex), 125

metadata
  API, 323
    creating objects in Java, 324-325
    file-based services, 323
    object-based services, 324
    overview, 323-324
  aware components, 210
  custom object tools, 32
  declarative, 6
  derived user interfaces, 13-14
  schema, 189
    child relationship, 190
    field, 190
    object, 189
    picklist, 191
    record type, 191
  XML, integrating, 30

methods
  abstract, 150
  access modifiers, 148
  action, 204-205
  Apex
    classes, 144
    email objects, 184-185
    assert, 154
    asyncMethod, 128
    containsKey, 124
    debug, 154
    DML database, 170-171
    future, 128
    getCost, 145
    getDescribe, 190
    handleInboundEmail, 185
    keySet, 124
    overloading, 150
    overriding, 149
    remove, 124
    rollback, 171
    sendEmail, 184
    setBccSender, 184
    setCcAddresses, 184
    setCost, 145
    setDocumentAttachments, 184
    setFileAttachments, 184
    setOrgWideEmailAddressId, 185
setReplyTo, 184
setSaveAsActivity, 183-185
setSavepoint, 171
setSenderDisplayName, 184
setUseSignature, 184
Skills Matrix testing, 242
test, 154
testAsUser, 242
testNoResourceForUser, 242
testNoResourceSelected, 242
testNoSkills, 242
testSave, 242
testWithSkills, 242
valueOf, 119
Microsoft Azure, 3
Model-View Controller (MVC), 15
modular Visualforce, 252
  composition, 253-254
  custom components, 255-256
  inclusion, 253
  static resources, 252-253
modularity, Apex interfaces, 148-149
monitoring
debug logs, 152
workflow queues, 335
multilingual support, 348
  custom labels, 349
  Translation Workbench, 348
multiple currencies, 350-353
  ACM, 353-354
  dated currency rates, retrieving, 354
  exchange rates, configuring, 350
  records, setting, 350
  SOQL conversions, 352
  support cases, logging, 350
  viewing, 352
  Visualforce, 352
multiplication operator (\(*\)), 121
multi-select picklists, 166
multitenancy of Force.com, 4-5
MVC (Model-View Controller), 15

N

names
  analytic snapshots, 346
  Apex variables, 117
  view components, 208
native user interfaces
  browsing data, 39
  data entry, 39
  opting-in, 39
  Visualforce
    custom buttons/links, 226
    custom tabs, 227
    development, 198
    page layout, 225-226
    standard buttons, 224-225
    standard pages, 222-224
nesting
  lists, 123
  result sets, 26
New button (Visualforce pages), 225
New Custom Field Wizard, 36-37
new features, opting-in, 39
notification email triggers, 192-193
NullPointerException class, 126
numbers
  data type conversions, 118
  rounding, 119

O

object-oriented programming
  analysis and design specialists, 11
  Apex
    classes, 143-147
    encapsulation, 143
    information hiding, 147-148
    inheritance, 149-150
    interfaces, 148-149
    polymorphism, 150-151
Object-Relational Mapping (ORM), 29
objects, 13, 21
  Connect Offline configuration, 101
  creating in Java, 324-325
custom
  assignment, defining, 53
  creating, 33-36
  projects, defining, 51-52
  records, creating, 39
  resource, defining, 52
  skill, defining, 53
  timecard, defining, 54
  tools, 31-32
  Views, 41
database
  records as, 130-132
  tuning, 22
data types, 118
DescribeFieldResult, 190
fields, 21
  accessibility, configuring, 80
  dependent, 90
  filters, 101
  history tracking, 97-98
  roll-up summary, 95-97
  security, 67-69
Group, 178
isDefaultRecordTypeMapping, 191
logical, 22
metadata, 189
operational tasks, 22
permissions, 64-66
ProcessInstance, 341
ProcessInstanceHistory, 341
Project Share, 177
relationships, 135-136
security, 63-65
  fields, 67-69
  profiles, 66-67
  Visualforce, 228-229
sending email (Apex)
  MassEmailMessage, 183
  methods, 184-185
  SingleEmailMessage, 181-182
services, 324
shared S2S, configuring, 281-284
  selecting fields to publish, 283
  selecting objects to publish, 283
  subscriptions, 284
sharing, 175-176
  standard, 47
  Timecard, 192-193
  undelete support, 22
offline access. See Connect Offline
Open Perspective dialog box, 113
OpenSAML toolkit
  federated SSO SAML assertions, 356
  initializing, 357
operations specialists, 12
operators (Apex), 121-122
opt_allOrNone parameter, 170
opting-in for features, 39
OR operator (|), 121
OR operator (||), 122
organization-wide defaults
  records, 71-72
  Services Manager application, 80
organization-wide email address unique identifiers, 185
ORM (Object-Relational Mapping), 29
outbound messaging, 274
  configuring, 274-276
  delivery status, 276
  workflow rule, 275-276
  limits, 274
  message definition, 276
  Web service, creating, 276-279
outbound requests, controlling, 289
outer joins (SOQL), 162
outputField component, 210
outputLabel component, 211
outputLink component, 214
outputPanel component, 214
outputText component, 214
overloading methods, 150
overriding
  methods, 149
  standard buttons, 225
  standard pages, 224
ownership
  filters, 101
  records, 69-70
PaaS (Platform as a Service), 2
Amazon Web Services, 2-3
Facebook, 4
Force.com, 3
Google App Engine, 3
Microsoft Azure, 3
pageBlockButtons component, 295
pageBlockSectionItem components, 295
pageMessages component, 231, 294
pages
  adding, Force.com sites, 266
  definitions, Visualforce view components, 209
  page layouts, 7
    custom objects, 36
    record types, 93
Visualforce, 197
  action status, 250-251
  actions as JavaScript functions, 247
  composition, 253-254
  custom buttons/links, 226
  custom components, 255-256
  custom tabs, 227
  inclusion, 253
  JavaScript events, 249
  layout, 225-226
  partial page refreshes, 246-247
  security, 229
Services Manager application, 294-295, 301-302
Skills Matrix (Services Manager), 238-240
  standard buttons, 224-225
  standard pages, 222-224
  static resources, 252-253
  timed events, 248
parent-to-child relationships (SOQL), 135
partial page refreshes (Visualforce pages), 246-247
Partner API (Web services), 305
PEM (Privacy Enhanced Mail), 358

permissions
  administrative, 66
  API enabled, 310
  objects, 64-66
persistence database records, 137-139
personal tags, 99
perspective (Force.com IDE), 113
PHP implementation, 278
picklists
  dependent, 104
  metadata, 191
  multi-select, 166
  values (record types), 92
Platform as a Service. See PaaS
platform documentation resources, 16
polymorphism (Apex), 150-151
populating account/contact IDs formulas, 57
primitive components, 211, 214-215
Privacy Enhanced Mail (PEM), 358
private access modifier, 148
Problems view (Force.com IDE), 115
procedural sharing reasons, 73
processing
  inbound email, 185-187
transactions
  DML database methods, 170-171
  record locking, 173-174
  savepoints, 171-173
ProcessInstance object, 341
ProcessInstanceHistory object, 341
profiles, 66
  administrative permissions, 66
  Consultant, testing, 85
  creating, 78-79
  custom, 66
  defined, 63
  delegated SSO, 361
  Executive VP, testing, 86
  field-level security, 68
  licensing, 67
  managing, 66
  object permissions, 66
Staffing Coordinator, 85
standard, 66
programming languages
Apex. See Apex
DML
database methods, 170-171
database records, persisting, 137-139
SAML assertions, 356-357
creating, 357
PEM conversions, 358
signing/encoding, 358
testing, 359
SOQL, 132
anti-joins, 166
Apex database queries, 132
approvals, retrieving, 341
currency conversions, 352
dated currency rates,
retrieving, 354
dynamic queries, 188
Governor Limits, 28
implicit joins, 26
inner joins, 162-163
multiple object queries, 134-135
multi-select picklists, 166
nested resultsets, 26
no functions allowed, 27
object relationships, 135-136
outer joins, 162
queries in Apex, 135
query results, sorting, 134
record limits, 134
records, retrieval, 315-316
records, filtering, 133-134
sample query, 26
semi-joins, 164-166
statements, 132
SOSL
Apex, 168-169
field specifications, 168
overview, 167-168
queries, 167, 189
record limits, 168
search groups, 168
WSDL
Apex, generating, 289-290
deleed authentication,
downloading, 360
outbound messaging, 276-279
Project Map dashboard, 362
controller, 366-367
creating, 362
defining, 367
GoogleMultiMap component,
364-365
Visualforce page, 366
projects
custom objects, defining, 51-52
FlexDemo, 260
Internet Explorer support, 263-264
MXML code, 260-261
static resource, 261
Visualforce page, 262
Force.com IDE, 114
lifecycles, 12
application retirement, 15
configuration management, 14
integrated logical databases, 13
interoperability, 15
metadata-derived user interfaces,
13-14
MVC pattern, 15
unit testing integration, 14
logical data model, 43
selecting, 9-10
Services Manager application
managers, 18
reporting, 104, 107
team selection, 11-12
Project Share object query, 177
properties. See attributes
protected access modifier, 148
public access modifier, 147-148
public groups, 70
public tags, 99
quality assurance engineers, 12
queries  
batch size, 316  
database (Apex), 132. See also SOQL  
dynamic database Apex, 188-189  
Group object, 178  
Project Share object, 177  
SOQL  
   dynamic, 188  
   multiple objects, 134-135  
   results, sorting, 134  
SOSL, 167, 189
query languages  
SOQL, 25  
   anti-joins, 166  
   Apex database queries, 132  
   approvals, retrieving, 341  
   currency conversions, 352  
   dated currency rates, 354  
   dynamic queries, 188  
   Governor Limits, 28  
   implicit joins, 26  
   inner joins, 162-163  
   multiple object queries, 134-135  
   multi-select picklists, 166  
   nested resultsets, 26  
   no functions allowed, 27  
   object relationships, 135-136  
   outer joins, 162  
   queries in Apex, 135  
   query results, sorting, 134  
   record limits, 134  
   records, retrieval, 315-316  
   records, filtering, 133-134  
   sample query, 26  
   semi-joins, 164-166  
   statements, 132  
SOSL, 28  
   Apex, 168-169  
   field specifications, 168  
   overview, 167-168
queries, 167, 189  
record limits, 168  
search groups, 168
queues (workflow, monitoring), 335
raising exceptions, 126
read-only properties, 145
receiving email (Apex), 186  
   class, creating, 186  
   configuring, 185  
   inbound email processing, 187  
   personalizing, 185
record-level security, 64, 69  
   record ownership, 69-70  
   sharing model, 70-73  
      organization–wide defaults, 71-72  
      sharing reasons, 72-73  
user groups, 70
records  
   approvals  
      submissions, 342  
      viewing, 337  
   bulk modifications, 319  
   creating, 39, 317-318  
   database  
      deleting, 139  
      inserting, 137  
      as objects, 130-132  
      persisting, 137-139  
      undeleting, 139  
      updating, 138  
      upserting, 138  
   deleting/undeleting, 319  
   detail, 102  
   forwarding, 286  
   limits  
      Connect Offline configuration, 102  
      SOSL, 168  
      locking, 173-174  
      ownership, 69-70, 101  
   retrieving, 315-316  
   security, 229
sharing, 284-288
SOQL, 133-134
types, 90-91
  changing, 95
  configuring for profiles, 93
  default, 95
  defining, 91-92
  deleting, 91
  field visibility, 93
  Master, 91
  metadata, 191
  new record creation, 93
  page layout assignment, 93
  picklist values, 92
  security, 92-93
  updating, 318
recursion (Apex), 127
registration, 31
relatedList component, 218
relational databases, 5
relationships, 25
  creating, 54-55
  database, 131
  definitions, 25
  fields, 37-38
  importing, 56
  integrity enforcement, 25
  objects, 135-136
  Services Manager application, 48-50
remove method, 124
repeating components, 212-213
replying (email), 184
reports, 343
  creating, 343-344
  custom types, 344
  running, 344
requests, approvals, 337
reRender attribute, 246
resources
  AppExchange, 16
  Code Share Web site, 16
  custom objects, 52
  developer discussion boards, 16
  Developer Force, 16
  Dreamforce, 16
  Ideas Web site, 16
  logical data model, 44
  platform documentation, 16
  systems integrators, 17
  technical support, 17
  types, 129
REST services, 291
  calling, 291
  invoking, 291-292
  testing, 293
restoring savepoints, 171
restrictions (Apex managed sharing), 175
resultsets, nesting, 26
retiring applications, 15
retrieving
  approvals, 341
  dated currency rates, 354
  records, 315-316
rich data types, 24
roles
  Services Manager application, 81
  user groups, 70
rollback method, 171
roll-up summary fields, 38, 95-97
  Services Manager application project reporting, 104, 107
  summary calculation, 96
rounding numbers, 119
rules
  Apex sharing, 176-180
  validation, 23
  workflow, 334
    active, 334
    debugging, 335
    inactive, 334
    outbound messaging, 275-276
    queue, monitoring, 335
    time-based, 334
running
  analytic snapshots, 346
  reports, 344
S

S2S (Salesforce-to-Salesforce), 279
  connecting, 280-281
  record sharing, 284-288
  shared objects, configuring, 281-284
  selecting fields to publish, 283
  selecting objects to publish, 283
  subscriptions, 284
Salesforce.com, 4
  Force.com application services, 6
  multitenancy, 5
Salesforce Object Query Language. See SOQL
sales representatives (Services Manager application), 18
SAML (Security Assertion Markup Language)
  assertions, 356-357
    creating, 357
    PEM conversions, 358
    signing/encoding, 358
    testing, 359
savepoints, transaction processing, 171-173
Schema Explorer
  Apex database relationships, 131
  Force.com IDE, 115
  SOQL queries, 132
schema metadata (dynamic Apex), 189
  child relationship, 190
  field, 190
  object, 189
  picklist, 191
  record type, 191
scope (Apex), 112
search groups (SOSL), 168
search layouts (custom objects), 36
searching tags, 99
sectionHeader component, 294
Secure Sockets Layer (SSL), 305
security
  architecture, 63
  databases, 63-65, 142
  Force.com sites, 266
  funnel, 64-65
  object-level, 63-65
    fields, 67-69
    profiles, 66-67
  record types, 92-93
  record-level, 64, 69
    record ownership, 69-70
    sharing model, 70-73
    user groups, 70
Services Manager application, 73
  business units, 75-78
  designing, 74-75
  field accessibility, configuring, 80
  implementing, 78
  job functions, 75
  organization-wide defaults, configuring, 80
  profiles, creating, 78-79
  role hierarchy, 81
  sharing rules, 82
  testing, 83-87
  tokens, logging in, 311
Visualforce, 228
  object-level, 228-229
  page-level, 229
  record-level, 229
Web services, 305
Security Assertion Markup Language. See SAML
selectCheckboxes component, 212
selecting
  projects, 9-10
  teams, 11-12
selectList component, 212
selectRadio component, 212
semi-joins (SOQL), 164-166
sendEmail method, 184
sender display names (email), 184
sending email (Apex), 181
  attachments, 184
  blind carbon copies, 184
  carbon copies, 184
  Documents, 184
  mass emails, 183
  methods, 184-185
organization-wide email address
unique identifiers, 185
replying, 184
sender display names, 184
signatures, 184
single messages, 181-182
templates, 182
tracking activity, 185
transactional behavior, 184

services
application, 6
business logic, 7
database, 7
Force.com, 7
integration, 8-9
user interface, 7-8

Services Manager application, 17
anonymous benchmarking, 293, 296-298
background, 17
Connect Offline for staffing, 107-108
custom objects list, 55
data model, creating
application definition, 50
assignment custom objects, 53
project custom objects, 51-52
relationships, 54-55
resource custom objects, 52
skill custom objects, 53
timecard custom objects, 54
dependent fields for skill types, 104
development plan, 18
e-mail notifications, 192-193
Force.com data model, 47, 50
relationships, 48-50
standard objects, 47
implementation strategy, 326
importing data, 55
preparations, 56-57
process, 58-59
verification, 59-60
integration scenario, 326
Java integration implementation
sample, 327-330

JSON file format, 330
logical data model, 41-46
assignments, 45
clients, 42
customers, 42
projects, 43
resources, 44
skills, 46
timecards, 46
Project Map dashboard, 362
controller, 366-367
creating, 362
defining, 367
GoogleMultiMap component, 364-365
Visualforce page, 366
roll-up summary fields for project reporting, 104, 107
security, 73
business units, 75-78
designing, 74-75
field accessibility, configuring, 80
implementing, 78
job functions, 75
organization-wide defaults, configuring, 80
profiles, creating, 78-79
role hierarchy, 81
sharing rules, 82
testing, 83-87
Skills Matrix, 233-234, 268
actionSupport, 271
basic implementation, 234
CompareSkillsComponent, 271
controller, 236-238, 269
controller tests, 240-242
custom component, 269
full implementation, 235
refreshing, 271
Visualforce page, 238-240
YUI overlay support, 270
timecard validation, 155
Force.com IDE setup, 156
triggers, 156-157
unit testing, 157-159
user roles, 18
utilization, 293
controller sample code, 299-301
Visualforce controller design, 295-296
Visualforce page design, 294-295
Visualforce page sample code, 301-302

setBccSender method, 184
setCcAddresses method, 184
setCost method, 145
setDocumentAttachments method, 184
setFileAttachments method, 184
set iteration for loops, 126
setOrgWideEmailAddressId method, 185
setReplyTo method, 184
Sets, 124
setSaveAsActivity method, 183-185
setSavepoint method, 171
setSenderDisplayName method, 184
setUseSignature method, 184
sharing, 64
managed Apex, 174-176
objects, 175-176
rules, 176-180
objects (S2S), 281, 284
selecting fields to publish, 283
selecting objects to publish, 283
subscriptions, 284
record-level security, 70, 73
organization-wide defaults, 71-72
sharing reasons, 72-73
records (S2S), 284-288
rules, 73, 82
signatures (email), 184
signed shift left operator (<<), 121
signed shift right operator (>>, 121
signing SAML assertions, 358
Simple Queue Service (SQS), 2
SimpleDateFormat Java API pattern, 120
single sign-on. See SSO
SingleEmailMessage object, 181-182

sites (Force.com), 264
details page, 267
domain names, 265
enabling, 265
Login Settings page, 267
main page, 266
pages, adding, 266
security, 266
user authentication, 267

skills
 custom objects, 53
 logical data model, 46
types, dependent fields, 104
validation rule listing, 53

Skills Matrix (Services Manager application), 233-234, 268
actionSupport, 271
basic implementation, 234
CompareSkillsComponent, 271
controller, 236-238, 269
controller tests, 240-242
custom component, 269
full implementation, 235
refreshing, 271
Visualforce page, 238-240
YUI overlay support, 270

snapshots (analytic), 346-347

SOAP data types, 313

SOQL (Salesforce Object Query Language), 25, 132
Apex database queries, 132
approvals, retrieving, 341
currency conversions, 352
dated currency rates, retrieving, 354
Governor Limits, 28
joins
 anti-joins, 166
 implicit, 26
inner, 162-163
outer, 162
semi-joins, 164-166

multi-select picklists, 166
nested resultsets, 26
no functions allowed, 27
object relationships, 135-136
queries
  Apex integration, 135
dynamic, 188
multiple objects, 134-135
results, sorting, 134
records
  filtering, 133-134
  limits, 134
  retrieval, 315-316
sample query, 26
statements, 132
SOSL (Salesforce Object Search Language), 25, 28
  Apex, 168-169
  field specifications, 168
  overview, 167-168
  queries, 167, 189
  record limits, 168
  search groups, 168
SQS (Simple Queue Service), 2
SSL (Secure Sockets Layer), 305
SSO (single sign-on), 354
delegated, 359-361
  configuring, 361
  errors, 362
  profiles, 361
  sample code, 360-361
  WSDL authentication, downloading, 360
federated, 354-359
  configuring, 355-356
  SAML assertions, 356-359
staffing
  coordinators
    Services Manager application, 18
testing, 85
    Services Manager application, 107-108
standardController attribute, 209
standards
  buttons (Visualforce pages), 224-225
  controllers, 197, 201
  multiple records, 202-203
  single records, 201-202
fields, 23
objects (Services Manager application), 47
pages, 222-224
profiles, 66
Standard Setup Wizard, 340
start/stop facets (actionStatus component), 250
statements
  control (Apex), 125
  Delete, 139
  exception (Apex), 126-127
  Insert, 137
  SOQL, 132
  Undelete, 139
  Update, 138
  Upsert, 138
static initializers, 147
static keyword, 144
static resources (Visualforce), 252-253
strings, 117
  concatenation operator (+), 121
  conversions, 119-120
stylesheet component, 214
subclasses (Apex), 149
submitting records for approval, 342
subscriptions (S2S shared objects), 284
subtraction operator (-), 121
super keyword, 149
syntax (Visualforce view components), 208-209
  attributes, 209
  bodies, 209
  names, 208
system exceptions, 314
systems integrators, 17
tabs
  custom
    creating, 39
    Visualforce pages, 227
standard page, 222
user interfaces, 39
tags, 98-99
teams, selecting, 11-12
technical support, 17
templates
e-mail (Apex), 182
Visualforce pages, 253-254
testAsUser method, 242
testing
Apex, 154
governor limits, 154
running, 155
test methods, writing, 154
methods, 154
REST services, 293
SAML assertions, 359
Services Manager security, 83
additional users, creating, 84
business unit collaboration, 86-87
consultant profiles, 85
data preparations, 84-85
Executive VP profiles, 86
Staffing Coordinator profiles, 85
unit testing, 14
Services Manager application
timecard validation, 157-159
Skills Matrix (Services Manager), 240-242
Visualforce, 232-233
testNoResourceForUser method, 242
testNoResourceSelected method, 242
testNoSkills method, 242
testSave method, 242
testWithSkills method, 242
this keyword, 145
time-based workflow rules, 334
TimecardManager class
creating, 156
unit testing, 158
timecards
custom objects, 54
e-mail notification triggers, 192-193
logical data model, 46
validation (Services Manager application), 155
Force.com IDE setup, 156
triggers, 156-157
unit testing, 157-159
time data types, 117, 120
timed events, 248
TLS (Transport Layer Security), 305
tools
Adobe Flex toolkit, 259
App Builder
custom object metadata, 32
Services Manager data model, creating, 50-54
custom objects, 31-32
Data Loader, importing data, 55-60
preparations, 56-57
process, 58-59
verification, 59-60
OpenSAML toolkit
federated SSO SAML assertions, 356
initializing, 357
Visualforce development, 198-199
Total Hours formula field, 54
tracking
e-mail activity (Apex), 185
field history, 97-98
traditional for loops, 126
transactions, processing
DML database methods, 170-171
record locking, 173-174
savepoints, 171-173
Translation Workbench, 348
Transport Layer Security (TLS), 305
triggers
custom objects, 35
database, 139
batch processing, 140
defining, 139-140
error handling, 141
governor limits, 141
upserting database records, 138

users
  authentication, 267
  creating, 84
  defined classes (Apex), 143
  groups, 70
  interfaces
    designers, 12
    generating, 30
    metadata-derived, 13-14
    native. See native user interfaces
    services, 7-8
    tabs, 39
  profiles, 66
    administrative permissions, 66
    Consultant, testing, 85
    creating, 78-79
    custom, 66
    defined, 63
    Executive VP, testing, 86
    field-level security, 68
    licensing, 67
    managing, 66
    object permissions, 66
    Staffing Coordinator, testing, 85
    standard, 66

Services Manager application, 18

utilization (Services Manager application), 293
  controller sample code, 299-301
  Visualforce
    controller design, 295-296
    page design, 294-295
    page sample code, 301-302

validation rules
  custom objects, 35
  fields, 23
  skills, 53

valueOf method, 119
variables
access modifiers, 148
Apex, 117-120
case sensitivity, 117
classes, 144-145
constants, 118
data types, 117-119
dates to strings conversions, 120
declaring, 117
enums, 118
names, 117
rounding numbers, 119
string to date conversions, 120
cost, 145
verifying data imports, 59-60
versions
Force.com, 5
Web services, 305
Vice President (Services Manager application), 18
view components (Visualforce), 207
action components, 213
attributes, 209
bodies, 209
data components, 210
metadata-aware, 210
primitive, 211
repeating, 212
Force.com-styled, 215-217
names, 208
overview, 208
page definitions, 209
primitive components, 214-215
sample controller and page, 220-222
syntax, 208-209
user interface, 218-220
visibility, 210
view standard page, 222
viewing, 39
approved records, 337
currencies, 352
custom objects, 41
debug logs, 152

Force.com IDE
Apex Test Runner, 116
Execute Anonymous, 116
Problems, 115
shared records, 286
virtual keyword, 149
visibility (Visualforce view components), 210
Visualforce, 8, 196
action components, 213-214
Ajax
action status, 250-251
actions as JavaScript functions, 247
JavaScript events, 249
partial page refreshes, 246-247
timed events, 248
architecture, 196
controllers, 196, 201
custom, 203-205
extensions, 206
standard, 201-203
data components, 210
metadata-aware, 210
primitive, 211
repeating, 212
development process, 198
development tools, 198-199
error handling, 230
error communication, 231
uncaught exceptions, 230-231
extending, 257
Adobe Flex. See Adobe Flex JavaScript libraries, 257-258
Force.com sites, 264
details page, 267
domain names, 265
enabling, 265
Login Settings page, 267
main page, 266
pages, adding, 266
security, 266
user authentication, 267
Force.com-styled components, 215-217
GoogleMultiMap component, 364–365
governor limits, 232
Hello World example, 199–201
modular, 252
composition, 253–254
custom components, 255–256
inclusion, 253
static resources, 252–253
multiple currencies support, 352
native user interface, 198
overview, 196
pages, 197
buttons, 224–226
layout, 225–226
links, 226
Skills Matrix (Services Manager application), 238–240
standard, 222–224
tabs, 227
primitive components, 214–215
ProjectMap page, 366
security, 228–229
Services Manager application
controller design, 295–296
page design, 294–295
unit tests, 232–233
user interface components, 218
detail, 220
enhancedList, 218
listViews, 218
relatedList, 218
view components, 207
attributes, 209
bodies, 209
names, 208
overview, 208
page definitions, 209
sample controller and page, 220–222
syntax, 208–209
visibility, 210
volume (data), 10

Web developers, 12
Web services, 304

APIs, 304
data integration, 29
Developer’s Guide Web site, 305
limits, 305–306
calling from Apex code, 289–290
clients, generating, 306, 310
C#, 309–310
Java, 307–308
custom, 320
administrative rights, 320
calling, 322–323
code example, 322
definition, 321
governor limits, 321
Enterprise API, 314
creating records, 317–318
deleting/undeleting records, 319
limits, 320
record bulk modifications, 319
record retrieval, 315–316
updating records, 318
error handling, 314
integration, 296–298
logging in, 310
API enabled permissions, 310
IP whitelisting, 311
login Web service, 311
security tokens, 311
troubleshooting, 312
Metadata API, 323
creating objects in Java, 324–325
file-based services, 323
object-based services, 324
overview, 323–324
outbound messaging, creating, 276–279
overview, 304
security, 305
SOAP data types, 313
versions, 305
Web sites
Adobe Flex toolkit, 259
AppExchange, 16
Code Share, 16
DE, 31
developer discussion boards, 16
Developer Force, 16, 31
Eclipse, 113
Excel Connector download, 33
Flex Builder, 260
Ideas, 16
SimpleDateFormat Java API
pattern, 120
Web Services API Developer’s
Guide, 305

while loops, 125
with sharing keyword, 142
with/without sharing security modes, 229

wizards
Import, 33
Jump Start, 340
New Custom Field, 36-37
Standard Setup, 340

workflow rules
active, 334
debugging, 335
inactive, 334
outbound messaging, configuring,
275-276
queue, monitoring, 335
time-based, 334

write locks, 173
write-only properties, 145

WSDL (Web Service Definition Language)
Apex, generating, 289-290
delegated authentication,
downloading, 360
outbound messaging, 276-279

X–Z

XML metadata, integrating, 30
XOR operator (^), 121
YahooGeocode class, 293
Years of Experience Formula field, 52

zero-cost improvements, 5