1

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

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.

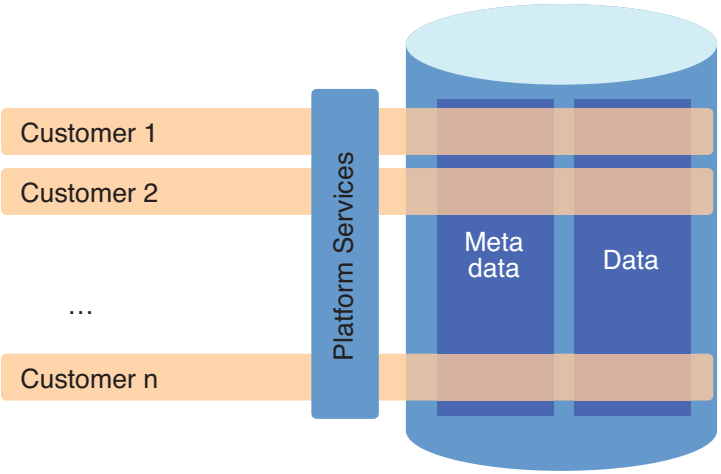

Figure 1-1 Multitenant architecture

- **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. 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.

- **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.

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 3, "Database Security": Define users, roles, and profiles. Configure sharing rules.
- 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