salesforce.com
Success. Not Software.

# Force.com: A Comprehensive Look at the World's Premier Cloud-Computing Platform

## Contents

## Executive Overview: Cloud Computing and Force.com

History has shown that every so often, incremental advances in technology and changes in business models create major paradigm shifts in the way that software applications are designed, built, and delivered to end users. In the 1980s, the invention of personal computers, computer networking, and graphical user interfaces gave rise to the adoption of client/server applications that ran on expensive, inflexible, character-mode mainframe applications. Today, we're witnessing another advance: Powerful mobile computing devices, reliable broadband Internet access, service-oriented architectures, and the high cost of managing dedicated on-premises applications are driving the transition away from traditional software toward the delivery of decomposable, managed, shared, on-demand, Web-based services.

### Cloud Computing

*Cloud computing, software as a service (SaaS)*, and *on-demand software* are related terms that generally refer to hardware, software applications, and services that are available for immediate use because they execute in the *cloud* (the Internet). Cloud computing may also be thought of as *utility-based computing* because, similar to power and water utilities, users pay only for the resources they use on a month-to-month basis.

Cloud computing is gaining popularity among businesses of all sizes. This model is beginning to replace the traditional on-premises model of delivering software applications because, by comparison, cloud computing delivers unprecedented levels of ease, productivity, and success. With cloud computing, organizations can simply use readily available applications and services to focus on getting their work done. They're no longer saddled with the burdens and high costs of managing data centers, hardware, and software. Just as power companies relieve homeowners from having to maintain personal power generators for electricity, cloud-based solutions relieve companies from having to maintain dedicated computer systems and staff to provide their business applications.

### Cloud-Computing Platforms

Every paradigm shift in software brings a fresh set of challenges for companies that build or consume applications. Cloud computing is no different. From an operational point of view, cloud computing presents several unique requirements for an application provider:

- **Operations/Availability** – Application providers are responsible not only for building the application, but for hosting and maintaining it so that it remains available for users.
- **Deployment** – For an application to be considered "on-demand," it must have automated mechanisms that let users sign up, log in, and start work immediately.
- **Elasticity** – A cloud-based application must be elastic, automatically scaling its consumption of computing resources to the demands placed on it at any given time.

Shared, cloud-based applications must also address some unique technical requirements:

- **Customization** – An application must let each organization customize its data model, interface, and business logic to fit its needs.
- **Security** – An application must have bulletproof, configurable security mechanisms that let an organization secure data among different types of users and organizations.
- **Upgradeability** – An application's code base must be easy to upgrade and patch, without breaking organization-specific customizations and configurations.
- **Integration** – An application must be able to combine selected data and functionality via industry-standard protocols, both with other cloud applications and with traditional on-premises applications.
- **Device Independence** – An application must work on various devices, including desktop computers and mobile devices, so users can be productive wherever and however they work.

The plethora of unique challenges for delivering cloud-based applications has given rise to a related paradigm shift, namely *cloud-computing platforms*. A hosted (cloud-computing) application platform is an Internet-based software development and deployment environment managed by

the platform provider, thus relieving customers of the operational burdens of application delivery. Cloud application platforms are more than just hosting solutions, however. They offer a new type of application framework that addresses all the demanding operational and technical requirements of delivering applications and services in the cloud.

### Comparing Clouds: Infrastructure Clouds and Cloud Platforms

Raw-computing clouds are *machine-centric services* that provide on-demand infrastructure services (called *infrastructure as a service*, or *IaaS*) for application deployment. Such clouds focus on providing the computing power and storage capacity needed to execute virtual servers that comprise various application components. Providers that choose to deploy applications using infrastructure clouds are responsible for monitoring, administering, and maintaining their application instances, just as they would within their data centers.

A cloud platform, also known as *platform as a service* (or *PaaS*), is an *application-centric approach* that abstracts the concept of servers. By using cloud application platforms, service providers can focus on core application development from day one and to deploy an application with the push of a button. At no time does an application provider need to worry about service availability, load balancing, scalability, system backups, operating system patches, security, and similar infrastructure concerns—all these responsibilities shift to the platform provider.

## Force.com

*Force.com* is cloud computing for the enterprise. Using the Force.com platform, private enterprises and commercial software providers alike can quickly create and run custom business applications over the Internet, without the need for up-front software and hardware expenditures, configuration, and maintenance. In addition to supporting the sales, marketing, and support applications for which salesforce.com is best known, Force.com makes the core technologies behind Salesforce CRM available for developing enterprise-class applications that serve a wide range of business needs and customer interactions. And because the design of Force.com meets all the unique requirements of cloud applications, service providers can easily deliver trusted, secure, configurable, and customizable services that can support multiple devices and are easy to upgrade and integrate with other applications.

## Force.com Applications

The Force.com platform is a rich development environment designed to enable the database applications at the center of most corporate application development projects. With its comprehensive stack of database, integration, logic, and user interface (UI) capabilities, Force.com can be used in various application scenarios that were, until recently, the exclusive domain of traditional client/server and application server database tools such as Visual Basic, .NET, and Java. These scenarios span the full range of business use cases, including Intranet-style applications such as employee directory and time-off management applications; departmental or group applications for recruiting, bug tracking, and asset management; and applications that extend Salesforce CRM, such as professional services project management.

To begin to understand the power of the Force.com platform, it's useful to look at some examples of what Force.com customers have already accomplished. For example, one of the world's largest media companies, which needed a database application to manage the scheduling of some of its assets, evaluated both Force.com and .NET. Although the team budgeted 3,000 hours for creating and deploying this application in .NET, they completed the application in fewer than 100 hours on Force.com. What's more, the application developed with Force.com included advanced features such as globalization and international currency conversion, which the .NET implementation design lacked.

The continued success of Electronic Arts (EA), the world's leading interactive entertainment software company, depends on being able to recruit and retain talented game developers. EA chose Force.com to create a recruiting application to manage its mission-critical talent acquisition activities. Three weeks after it began the design, EA rolled out an award-winning system—a success that led EA to create ten additional applications. One of these applications, a vendor management system, was estimated to require 9–12 months to develop with traditional means. With Force.com, EA deployed the application in just 6 weeks.

These customers are not alone. Salesforce.com customers have created more than 100,000 custom applications on the Force.com platform. And these applications don't just operate within a Salesforce CRM environment. By using the Force.com Web services API to integrate these applications with existing systems, customers are executing more than 100 million Force.com API transactions per day.

## Force.com Development vs. Traditional Development

A recent independent analysis of Force.com highlights how much more efficiently businesses can build and deploy applications in the Force.com cloud than with traditional means. Galorath Inc. conducted a 20-month study of the Force.com platform to calibrate its SEER cost estimation tool for the budgeting of Force.com projects. This work led to these conclusions about Force.com compared to traditional Java-based application development platforms:

- Requirements definition time is 25 percent less due to rapid prototyping and the updating of Force.com applications.
- The testing effort is at least 10 percent less due to extensive re-use of proven code.
- Development productivity of new code is five times greater.
- Overall project cost is 30 to 40 percent less.

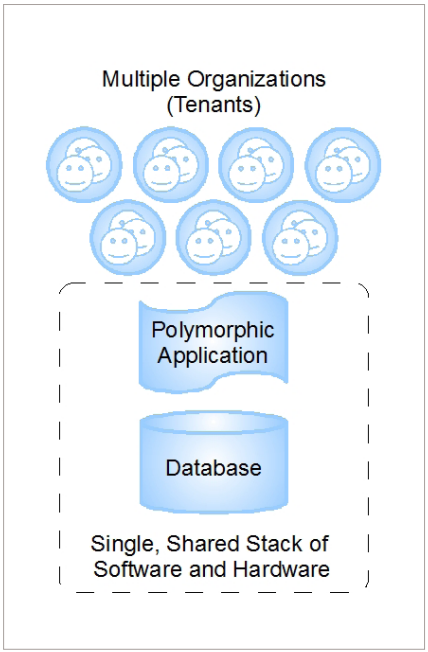
## Force.com Enabling Technologies

The capabilities and success of the Force.com platform are largely tied to two key enabling technologies: multitenancy and metadata.

### Multitenancy

The origins of today's emerging utility-computing model stem from the failure of the application service provider (ASP) model of the 1990s. An ASP delivered a traditional, on-premises application from its data center to a remote customer over a network, managing all the application's operational aspects. The customer was responsible for purchasing and maintaining a conventional perpetual software license, and the ASP charged a fee for managing the entire process. But the cost inefficiencies of deploying and maintaining a dedicated hardware and software deployment for each customer (or tenant) doomed the ASP model to failure. The desire for a utility-based computing model did not die with ASPs, however; instead, it sparked the innovation that resulted in a different way of building applications with less expense and overhead that could serve the needs of many customers.

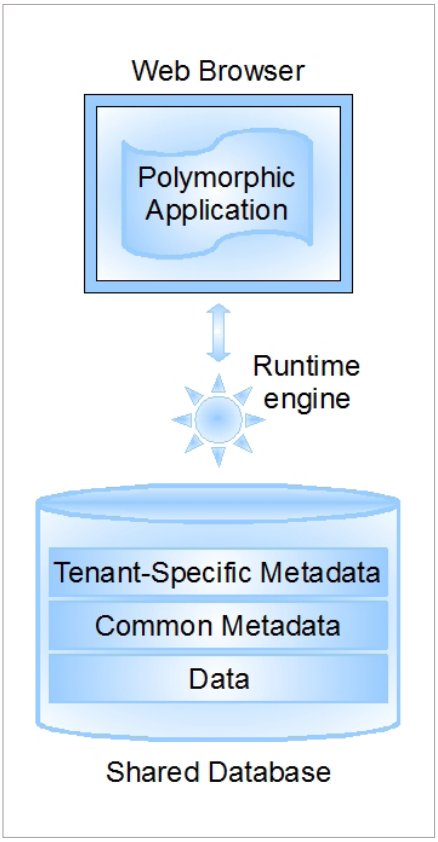
Modern cloud-based applications are much more cost-efficient than their single-tenant counterparts, thanks to a new application development approach that's at the heart of the Force.com platform—*multitenancy*. A multitenant application can serve the needs of multiple organizations, or tenants, by sharing a single physical instance and version of the application. Tenants using a multitenant service operate in virtual isolation from one another; Organizations can use and customize an application as though they each has its own instance, yet each organization's data and customizations remain secure and insulated from the activity of all other tenants. The single-application instance effectively morphs at runtime for any particular tenant at any given time.




Multitenant business applications that rely on Force.com, such as the extremely successful Salesforce CRM application, are similar to consumer applications such as Google Mail that also run a single code base and depend on an infrastructure shared by all users. It is this multitenant architecture that makes possible the quick deployment, low risk, and rapid innovation for which salesforce.com has become known.

**Metadata**

Multitenancy is practical only when it can support applications that are reliable, customizable, upgradeable, secure, and fast. But how can just one instance of a multitenant application let each tenant create custom extensions to standard data objects and new custom data objects? How will tenant-specific data be kept secure in a shared database so one tenant can't see another tenant's data? How can one tenant customize the application's interface and business logic in real time without affecting the functionality or availability of the application for all other tenants? How can the core application's code base be patched or upgraded without breaking tenant-specific customizations? And how will the application's response time scale as tens of thousands of tenants subscribe to the service? It's difficult (if not impossible) to create a statically compiled application executable that can meet these and other unique challenges of multitenancy. Inherently, a multitenant application must be dynamic—or polymorphic—in nature to fulfill the requirements of individual tenants and their users.



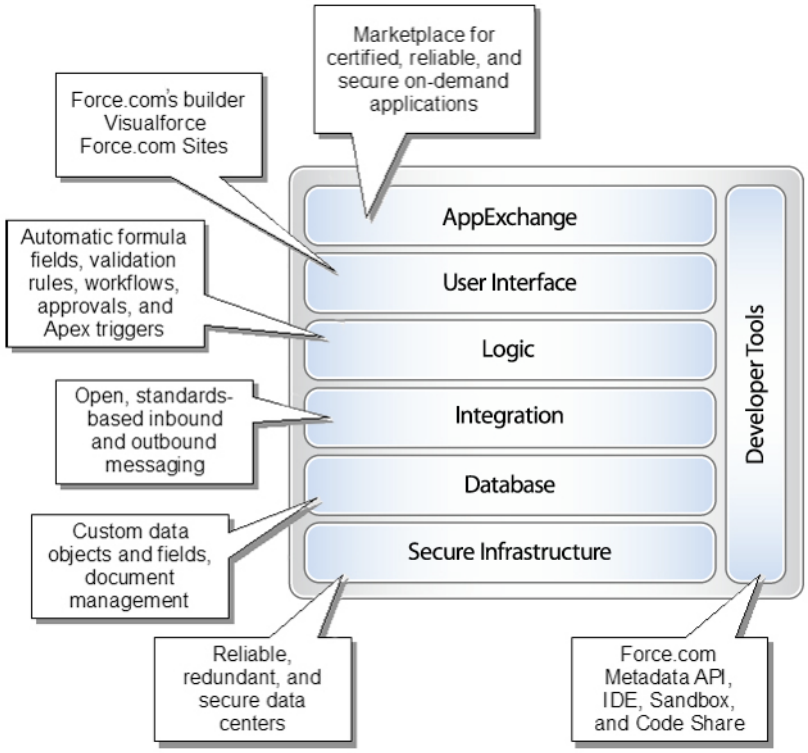

To meet these challenges, multitenant application designs have evolved to use a runtime engine that generates application components from *metadata*—data about the application itself. In Force.com's metadata-driven architecture, there's a clear separation of the compiled runtime engine (kernel), application data, the metadata that describes the base functionality of an application, and the metadata that corresponds to each tenant's customizations. These distinct boundaries make it possible to independently update the system kernel, modify the core application, or customize tenant-specific components all in real time, with no risk of one update affecting the other components.

In Force.com, everything exposed to developers and application users is internally represented as metadata. Forms, reports, work flows, user access privileges, tenant-specific customizations and business logic, and even the definitions of underlying data objects and indexes are all abstract constructs that exist merely as metadata. For example, when a developer builds a new custom application and defines a custom table, lays out a data entry form, or writes some procedural code, Force.com does not create a table in a database or compile any code. Instead, Force.com simply stores metadata that the platform's engine uses to generate the "virtual" application components at runtime. When someone wants to modify or customize the application, all that's required is a simple, non-blocking update to the corresponding metadata. When a user works with the application, Force.com's runtime application generator uses metadata to render the application components in the interface.

Force.com's metadata architecture is scalable. Because metadata is a key ingredient of Force.com applications, the platform's runtime engine must optimize access to metadata; otherwise, frequent metadata access would hinder platform scalability. To address this potential bottleneck, Force.com uses optimized metadata caches to maintain the most recently used metadata in memory, thus avoiding performance-sapping disk I/O and code recompilations and improving application response times.

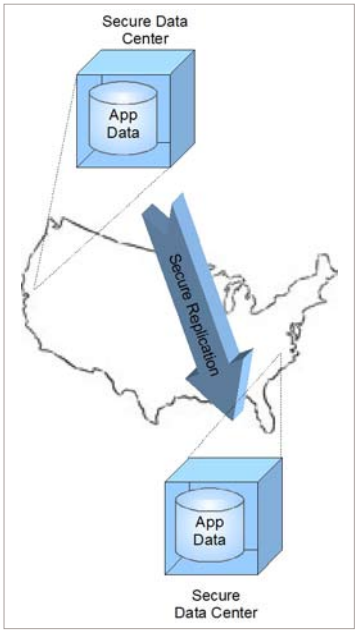
## Force.com Stack Overview

To understand the capabilities of the Force.com platform, this section provides an overview of the "Force.com stack"—the layers of technologies and services that make up the platform.




### Service Delivery: Global, Trusted, Secure Infrastructure

The Force.com service delivery infrastructure provides the foundation for the most challenging cloud-based requirements. This infrastructure, which consists of advanced and highly managed data center and security technologies, currently powers more than 200 million transactions a day and supports more than 1.5 million subscribers. It is this same infrastructure that delivers all cloud applications developed and deployed by customers as well as the Salesforce CRM applications. To see the availability, volume, and performance achieved by the Force.com infrastructure in near real time, visit http://trust.salesforce.com/.




### Database as a Service: Custom Objects and Fields

The Force.com database builds on the foundation of the Force.com infrastructure to provide much of the platform's development power. Here you can create metadata that describes an application's custom database objects such as tables and fields.

You can also declare metadata to establish relationships between data objects that are automatically implemented as master lookups and lists of related detail objects in Force.com applications. To ensure data integrity, you can declare data validation rules and use formulas to logically derive new data values. You can even audit database changes with the click of a mouse. The Force.com database provides all these capabilities without the usual requirements for database maintenance and overhead. There's no backup, no tuning, and no stream of upgrades for you to manage because salesforce.com takes care of these tasks.

**Integration as a Service: Open, Standards-Based Integration Solutions**

Your cloud applications may have to fit into an existing environment that includes a variety of data sources and applications. The Force.com platform provides the resources for integrating those applications into your current environment to access data in other systems, create mashups that combine data from multiple sources, or include external systems in your business processes.

At the core of these integration capabilities is the Force.com Web Services API, which provides easy access to all the data stored in a Force.com application through an open, standards-based SOAP Web service. This API is used by numerous environments, ranging from traditional development tools such as .NET, Java, and PHP to middleware and integration solutions such as BizTalk, Informatica, and Tibco. In addition, salesforce.com and third parties have used the API to create prepackaged connectors to many applications, including SAP R/3, Oracle Financials, and others. There are even prebuilt integration solutions that let organizations "connect the clouds" by integrating Force.com applications with other cloud-based offerings such as Amazon Web Services, Facebook, and Google Apps.

**Logic as a Service: Automated Workflows, Approval Processes, and Apex Triggers**

The Force.com platform makes it easy to automate a company's unique business processes and requirements. The workflow engine provides common, reusable process components such as task creation, record assignment, time-based actions, and even event-based system integration. With Force.com, you can easily incorporate these components into your application's logic.

For even greater flexibility, you can use Apex Code, salesforce.com's programming language, to extend your applications to include virtually any kind of business logic and functionality. Like a database stored procedure, Apex can be used to create triggers that execute automatically in response to database operations. Apex can also access and invoke external Web services. As an integral part of the Force.com platform, Apex leverages the platform's multitenant architecture to ensure the scalability of any applications that run on it.

**User Interface as a Service: Builder and Visualforce**

Force.com provides two options for creating and customizing the UI of platform applications. Through Force.com's builder, a simple point-and-click/drag-and-drop interface lets you build and change the layout and the order of data fields on pages, rename and re-order tabs, create complex reports, and even create different views of the data for different users.

For more complete UI interface control, Force.com includes Visualforce, a complete framework for creating and running virtually any UI, for any application, on any device. With Visualforce, developers can extend a cloud application's boundaries in almost any direction. By using traditional Web development technologies in conjunction with rest of the Force.com capabilities, they can exercise pixel-level control over the application's appearance and behavior.

Force.com's newest feature, Force.com Sites, harnesses Visualforce to help you easily transform current applications into public Web sites. With Force.com Sites, you can create public Web sites and applications that run natively on the Force.com platform, extending your reach to new users on intranets, external Web sites, and online communities. Force.com Sites lets you publish data from your Force.com organization to any Web site or build public applications that run entirely on Force.com.

**Development as a Service: Comprehensive PaaS Developer Tools**

Force.com provides several tools and APIs that enterprise developers can use to easily harness the promise of cloud computing. Providing full access to the database, logic, and UI capabilities of Force.com, these technologies unite the productivity of development and IT collaboration tools with the power of the platform. Included in the development as a service (DaaS) layer are the Force.com Metadata API, the Force.com Integrated Development Environment (IDE), the Force.com Sandbox, and Force.com Code Share—a comprehensive set of services for building cloud-based enterprise and commercial applications.

### Force.com AppExchange: A Cloud Application Marketplace

The Force.com AppExchange is a marketplace that connects Force.com commercial application developers with potential users. Customers or application providers wanting to share their applications can publish them on the AppExchange. Anyone wanting to use those applications can press a button to install them instantly into their Salesforce CRM accounts, without any of the traditional software installation and configuration hassles. To date, more than 800 certified reliable, secure, and interoperable applications are available via the AppExchange.

## Force.com: A Closer Look

Now that you have a broad understanding of the Force.com platform, let's take a closer look at the specific technologies within each layer.

### Infrastructure as a Service

The foundation of Force.com is the infrastructure that supports the upper layers of the platform..

#### Redundancy

To deliver unmatched reliability, the infrastructure of Force.com consists of three geographically separated production data centers and a production-class lab facility that use near-real-time replication to mirror the data at each location. Salesforce.com's comprehensive and validated disaster recovery plans ensure the platform will remain available for applications even in the most calamitous situations.

Single points of failure don't exist in Force.com to further ensure maximum uptime and performance for all platform applications. For example, the platform uses a carrier-neutral network strategy and relies extensively on high-availability server and network technologies that employ redundancy at every layer.

#### Security

Rest assured: The physical, network, host, data transmission, and database security levels of the Force.com platform are world class. All data centers are SAS 70 Type II, SysTrust, and ISO 27001 certified. Each facility's security team monitors site perimeters 24/365, and five levels of biometric scanning and other technologies encapsulate internal operations centers to ensure only authorized personnel have system access. To protect data in transit between data centers, the system uses secure point-to-point data replication.

Salesforce.com performs both internal and external vulnerability assessments on a regular basis to further ensure system security. Internal assessments help guarantee that software releases are secure and include design, architecture, and code quality reviews by internal staff as well as by third parties. External assessments focus on other types of potential concerns, including exposure to such things as buffer and parameter overflow, SQL injection, cross-site scripting, and much more.

#### Performance and Scalability

Force.com's infrastructure design can scale both vertically and horizontally due to its unique pod architecture. A pod is a set of industry-standard resources (high-performance database, Web, application, search, email, storage, backup servers, load balancers, etc.) that work together to serve the needs of a limited collection of organizations and applications. To prevent demand overload of any one pod's resources, salesforce.com provisions a new pod when existing pods are at or nearing predefined capacity thresholds.

#### Monitoring

A collection of systems management tools closely monitor Force.com's health and performance 24/7 and alert the platform's team of specialized engineers to potential problems and resolutions. All situations of interest are made public at http://trust.salesforce.com/.

To adequately prepare for future growth of Force.com usage, weekly capacity planning reports measure and project the individual demands of each enterprise customer and collective demands placed on each data center.

### Change Management

Salesforce.com thoroughly tests and manages new releases of Force.com to ensure both quality and the transparency of changes to existing platform applications. As part of the process, salesforce.com reruns each application's code unit tests (more on these tests later) prior to production updates to ensure that application code continues to execute properly without modification. Current maintenance windows for Force.com are established based on the analysis of customer usage patterns and traffic, with normal 4-hour windows reserved for routine maintenance. In the future, salesforce.com plans to further minimize and eventually eliminate planned downtime.

## Database as a Service

The Force.com platform provides a powerful and intuitive data persistence layer, known as the Force.com Database. This layer includes a Web-based application designer (Force.com's builder) for declaring various application components and automatically generating a "native" application UI around them. Anyone, including users with varied skill sets—such as application developers, designers, and business analysts—can use the builder to quickly specify an application's data model, UI, reports, and more, all using out-of-the-box Force.com platform features. This section provides an introduction to Force.com Database services and supporting functionality.

### Objects, Fields, and Data Storage

The Force.com Database uses objects to store instances (or records) of data. An object includes a number of fields. An object can also relate to other objects, with relationship fields that map records in one object to records in another (more about relationships later).

The Force.com Database has several similarities to and differences from the functionality of traditional relational databases. A Force.com object and its fields are analogous to a relational table and its columns, and Force.com relationships are functionally analogous to relational database referential integrity constraints. Unlike relational database tables that are separate database objects with dedicated storage, however, Force.com maintains the structure of an object as metadata. The platform stores the application data for all objects in a few large database tables that serve as heap storage. Force.com's engine then renders virtual object records at runtime by analyzing the corresponding metadata. To optimize access to data in large storage tables, Force.com's engine relies on a set of specialized pivot tables that maintain denormalized data for various purposes such as indexing, uniqueness, and relationships.

### Basic Field Types

All fields in an object correspond to a particular data type that defines what type of data the field can contain, thus providing a basic level of domain integrity. The Force.com Database supports common scalar datatypes such as Date/Time, Number, and Text as well as several specialized datatypes that help simplify otherwise complicated or laborious application development tasks.

- **Auto Number** – A system-generated, read-only sequence number analogous to the SQL identity type. Auto-number fields are useful for providing a unique ID that is independent of the internal object ID. Force.com does not use these types of fields to maintain object relationships.
- **Checkbox** – An interface element for representing Boolean data.
- **Email, Phone, and URL** – Format-validated email, phone, and URL string representations.
- **Currency** – For representing a formatted number type, with optional multi-currency support.
- **Picklist and Multi-Select Picklists** – For creating lists of values.
- **Formula** – A read-only field that generates data from an expression. (For more information, see "Logic as a Service.")

When defining a text field for an object that contains sensitive data, developers can easily configure the field so that Force.com encrypts the corresponding data and optionally creates an input mask to hide screen information from prying eyes. Force.com encrypts fields using 128-bit AES (Advanced Encryption Standard) algorithm keys.

### Relational Field Types

The Force.com Database handles object relationships somewhat differently from relational databases to provide similar functionality. Instead of having to declare keys (primary keys and foreign keys) and relationships in terms of these keys, the Force.com Database uses "relationship" fields. A relationship field stores the ID of the parent record in a relationship and optionally provides UI representations in both the parent and child records.

Force.com provides two different relationship field types:

- **Lookup Relationship** – Creates a relationship that links one object to another object. The relationship field allows navigation from records in one object to the related records in another object (both visually and programmatically).
- **Master-Detail Relationship** – Creates a special type of relationship between two objects—the child, or "detail," and the parent, or "master." For every detail record in a master-detail relationship, Force.com requires a relationship field value and, once the record is created, prohibits subsequent updates to the value. Also, when someone deletes a master record that has dependent detail records, Force.com automatically "cascades" the delete by also deleting all corresponding detail records.

Lookup relationships are useful for creating one-to-one and one-to-many relationships. Master-detail relationships are useful whenever there is a tight binding between two objects. For example, consider a blog and blog posts. If someone deletes the entire blog, the blog posts should vanish, too. Master-detail relationships are also useful to establish many-to-many relationships with a junction object, which relates to two other objects in a master-detail relationship.

### Identity Fields

In contrast to relational databases, the Force.com platform has no notion of a primary key in an object. That's because the Force.com Database automatically declares an identity field (called ID) for every object and manages the identity data in every record. This ID field usually comes in a 15-character, case-sensitive form that appears as part of the URL when you use native Force.com applications. Consider this example:

https://na3.Salesforce.com/0015000000Gv7qJ

In this example, "0015000000Gv7qJ" is the record's identifier. Every application record has an ID, which provides a convenient shortcut for retrieving and displaying the record and associated metadata via the native Force.com Web application UI.

### System Fields

All objects include a number of read-only system fields that Force.com manages automatically. The ID field, discussed in the previous section, is one such field. Others include these:

- **CreatedDate** – The date and time when a record was created.
- **CreatedById** – The ID of the user who created a record.
- **LastModifiedById** – The ID of the user who last modified a record.
- **LastModifiedDate** – The date and time when a record was last modified by a user.
- **SystemModStamp** – The date and time when a record was last modified by a user or process (such as a trigger).

### The Name Field

An object's required Name field has a unique purpose—to manage a record's human-readable identifiers. Although a record's Name field does not have to be unique, it is the primary way users tend to distinguish one record from another. For example, an Account object might use "Account Name," a Case object might use "Case Number," and so on. The automatically generated UIs of native applications created with Force.com's builder always display the value for a record's Name as a link to the record itself (to its detail page).

The Name field in an object can be one of two types: a text string or an auto-number field. For an auto-number field, developers must specify the field's format and the starting number. Auto-number fields increment by 1 each time someone creates a record.

#### Other Object Features

In object-oriented programming languages, a class can have methods that add behaviors to associated objects. Similarly, Force.com objects are more than just a persistence layer for application data because objects can include associated UI and programmatic components that make the objects more useful and related application development tasks easier. For example:

- **Labels and Help** – Every object and record has a label that can include a description (for internal documentation) and help, which the natively generated UI automatically includes.
- **Notes and Attachments** – Users can create, view, and edit notes and add attachments for any record in an object that enables this functionality. With this capability, the object's users can easily add arbitrary text notes and upload associated documents for each record.
- **Track Field History** – Developers can configure specific fields in objects so that Force.com audits associated changes. Any time a user modifies any data in a field whose history is set to be tracked, Force.com automatically adds a new entry to a related history list. This history list tracks the date, time, nature of the change, and who made the change.
- **Security** – Database services provide a flexible security model organizations can use to control access to objects, records, and/or fields.

The "Logic as a Service" section later in this document discusses several programmatic features that simplify the creation of "automated" operations. These features include formula fields, roll-up summary fields, workflows, approval processes, and triggers.

#### Data Security

The Force.com platform provides a range of security features organizations can use to protect access to their data. Basic security features include these:

- **User authentication** features govern connections to the Force.com Database, such as SAML, IP range restrictions on user logons, session security, and auditing.
- **Administrative permissions** control access to certain areas of Force.com platform functionality for privileged users.
- **Object-level permissions** govern general object access, including Create, Read, Update, and Delete (CRUD) permissions.
- **Field-level permissions** govern access to an object's specific fields.
- **Profiles** make it easier to maintain application access permissions for users, especially for large user populations. A profile groups together related data access permissions, which typically corresponds to a "type" of user—for example, there might be separate profiles for a sales manager, a sales person, or a sales analyst. An administrator can assign a profile to each application user, which gives users the permissions they need to do their jobs. A profile also can control access to other aspects of security, such as which tabs or applications are available to profile users.

Force.com also provides mechanisms to implement record-level security. Central to these mechanisms is the concept of record ownership. A record's owner has all privileges for that record, including the ability to share the record with other users or transfer its ownership. Either a single user or a group of users (called a queue) can own a record.

The Force.com platform provides several record-sharing schemes: manual sharing, automatic sharing rules, and programmatic sharing.

- With **manual sharing**, simple UI buttons and screens make it easy for record owners to define who can access individual records.

- **Automatic sharing rules** automatically grant a particular type of sharing access to records owned by users of one group to those of another group. In this context, a group can be either a public group (a defined set of users) or a role (a defined a set of users with that role and optional subordinates). Sharing rules are only useful for granting wider access to data, not for restricting data access.
- **Programmatic sharing** with Apex is the final sharing option. Organizations can write classes that make it possible to implement sharing based on virtually any type of logical or data-based conditions. (For an introduction to Apex, see "Logic as a Service.")

#### Text Management and Text Searches

To improve application response times, the Force.com platform uses an external search service that optimizes full-text indexing and searches. As applications update textual data, the background processes of the search service asynchronously update tenant- and user-specific indexes in near real time. This separation of duties between the application engine and the search service lets platform applications efficiently process transactions without the overhead of text index updates. At the same time, the applications quickly give users accurate search results.

#### Multitenant-Aware Query Optimizer

To dynamically build applications in response to specific user requests, Force.com's runtime application generator relies heavily on its "multitenant-aware" query optimizer to execute internal operations as efficiently as possible. The query optimizer determines which user is executing a given application function. Then, using related tenant-specific metadata along with internal system pivot tables, the optimizer builds and executes data access operations as optimized database queries.

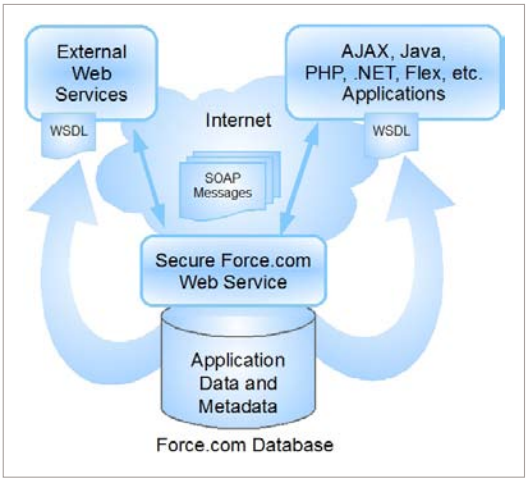
### Integration as a Service

Force.com includes a wide range of technologies that dramatically reduce the effort necessary to integrate platform applications with single sign-on, back-office ERP, desktop applications, middleware, on-premises legacy systems, and other cloud-based platforms such as Amazon Web Services, Facebook, and Google App Engine. This section provides information about the underlying Force.com technologies developers can use to build custom integration solutions that both produce and consume Web services. It also identifies several prepackaged integration solutions built on these technologies.

Before getting into feature details, it's important to understand that, in general, salesforce.com builds all Force.com integration technologies to be compliant with open Web services and service-oriented architecture (SOA) standards, including SOAP, WSDL, and WS-I Basic Profile. As a result, Force.com integrations are feasible with any development platform or middleware solution that supports Web service standards, including Java, .NET, PHP, and Perl.

#### Web Services API: Programmatic Access to Application Data

An external (or Force.com) application or Web service can access an organization's data using the secure Force.com Web Services API. Thousands of organizations rely on this API to work with data managed by Force.com, which executes more than 100 million API integration transactions per day.




Using the Force.com Web services API to access an organization's data is straightforward. After building the application's data model (tables, fields, and so on), developers download a Web Service Description Language (WSDL) file using a simple point-and-click interface available on the Force.com platform. The development environment uses this WSDL file to generate a corresponding API that provides common

data access methods—such as query, create, update, insert, upsert, and delete—for working with object instances (rows of data) in the organization's data model.

There are two types of Force.com WSDL files: an Enterprise WSDL file for developers building organization-specific applications and a Partner WSDL file for partners developing general client applications for multiple organizations.

An Enterprise WSDL file is a strongly typed representation of an organization's data model. This type of file provides the development environment with information about the organization's schema, data types, and fields, allowing for tighter integration between the WSDL file and the Force.com Web service. An Enterprise WSDL file changes if custom fields or custom objects are added to, renamed, or removed from an organization's application schema. In contrast, a Partner WSDL file is a loosely typed representation of the Force.com object model, so that a Partner WSDL provides an API that is useful for accessing data within any organization.

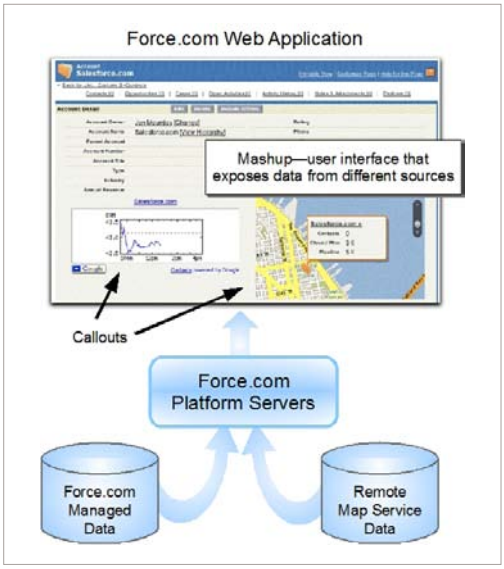
**Apex Web Services: Programmatic Access to Application Logic**

An external (or Force.com) application can execute the specific business logic routines of a Force.com application by using Apex Web services. With Apex Web services, organizations can expose Apex classes as secure Web services.

Creating a business logic routine as a Web service is also straightforward. First, a developer writes the Apex class and uses the "webService" annotation to specify that Force.com will expose the class as Web service. Next, the developer downloads the WSDL file that corresponds to the Apex class. Then, the development environment uses the WSDL file to generate an API for calling the Apex class from within applications.

**Force.com Callouts and Mashups: UI Integration**

A Force.com callout is a feature that a platform application can use to build a mashup—a UI that combines content from multiple sources. For example, a Force.com application might use a mashup to display a form for entering real estate data managed by Force.com alongside a callout that renders a map of the location's address using cartographic data managed by a remote mapping service such as Google Maps.




Force.com callouts are immune to common problems such as inconsistencies in the browser display and data access. Such problems typically plague traditional mashup solutions because callouts don't rely on the browser to combine the content. Instead, Force.com platform servers access all the internal and external content that corresponds to each mashup fragment (including callouts), build the page, and deliver a fully composed page to the client.

A developer can build a synchronous or asynchronous callout corresponding to any Web service that is accessible on the Internet, including public Web services and private Web services that correspond to an organization's on-premises application. To build a callout, the developer simply creates Apex stubs based on a Web service's WSDL file. Callouts are also possible with remote REST services.

**Outbound Messaging: Asynchronous Notifications and Callbacks**

Outbound messaging is a Force.com feature to automatically notify external applications and services of events that happen within the platform application. Outbound messaging enables real-time, asynchronous application integration by sending secure outbound SOAP messages from platform applications to any Web service endpoint accessible on the Internet.

Several steps are necessary for using outbound messaging. First, a developer uses a Web form to configure a workflow rule that defines the conditions under which the notification will be sent. Next, the developer creates an outbound message workflow action to specify the outbound message content and target endpoint URL. Once the workflow action is in place, the developer generates a WSDL file that corresponds to the outbound message and uses the WSDL file to build or configure a "listener" for messages. Because outbound messaging is based on open standards, developers can build custom listeners with languages such as .NET or Java, or simply configure prebuilt integration middleware facilities as listeners for outbound messages.

Force.com sends outbound messages with enough information so external services can, in turn, send back related messages to complete workflows. Each outbound message includes a token and a URL for callbacks.

Force.com's outbound messaging feature is both reliable and secure. The platform automatically retries sending messages that fail for up to 24 hours. Message transmission is secure because it is protected by HTTPS and X.509 certificates.

#### Prepackaged Integration Solutions

Developers can use the Force.com Web Services API, Apex Web services, callouts, and outbound messaging to address almost any type of custom integration requirement. Many prebuilt integration solutions, which are also built on these technologies, are available to organizations to more easily address common integration scenarios. For example:

- The Force,com AppExchange contains prebuilt mashups for integration with many business Web services.
- Native ERP connectors provide integration with SAP R/3 and Oracle Applications 11i.
- Native desktop connectors provide integration with Microsoft Office and Lotus Notes.
- The ApexConnect category of the AppExchange contains certified connectors for more than 30 middleware partners, including Cast Iron, Pervasive, and Tibco.
- Developer toolkits for Java, .NET, Amazon Web Services, Facebook, Google, and so on provide a level of abstraction that simplifies the steps necessary to build custom integration solutions.

### Logic as a Service

Force.com provides several features that developers and business analysts alike can use to build "smart" database applications—applications with attributes that help increase user productivity, improve data quality, automate manual processes, and adapt quickly to changing requirements. Overall, the Force.com platform supports three options for implementing an application's business processing logic.

- **Declarative logic**, including required and unique fields, audit history tracking, workflow rules, and approval processes
- **Formula-based logic**, including formula fields, data validation rules, workflow rules, and approval processes
- **Procedural logic**, including Apex triggers and classes

No matter which option an organization chooses to automate a business process, it is important to understand that all types of logic are centrally created on the Force.com platform and enforced by it. This functionality is important because it improves application designer productivity (the process only has to be implemented once for all applications) and ensures consistent behavior when any application accesses the underlying data.

This section provides a closer look at some of the more sophisticated Force.com features available for automating business processes: formula fields, roll-up summary fields, workflows, approval processes, and Apex triggers and classes.

### Formula Fields and Roll-Up Summary Fields

Database applications commonly display virtual fields that automatically calculate or otherwise derive data based on data in other fields. For example, the read-only Due Date field for a Job Application object might automatically default to 5 days from the date that a job application is created.

By declaring formula fields, anyone can easily add virtual fields to an object without complex coding. A formula field is a read-only field in a Force.com object that automatically derives its value from an expression. Force.com automatically updates a formula field's value when any of the source fields change. Formula fields are simple to declare using the builder's point-and-click UI.

The Due Date field requirement is a simple case in which a formula field with a date expression automatically sets a field's value. A more complex formula field might use an expression to derive the value of one field based on the value of other fields. For example, the calculation of a Line Total field is based on the result of calculating the value of the Quantity and Unit Price line items in a sales order.

Force.com also supports "cross-object" formula fields when there is a need for an object with a virtual field that is based on data in a different object. For example, a formula field in a Customer object can use an expression to look up a company's Web site URL from the customer's related Company object and then display the field as a hyperlink on the customer's data entry form.

A specialized variation of a cross-object formula field is a roll-up summary field. This type of field is useful for deriving a field in a master object that automatically counts, summarizes, or presents the minimum or maximum field values in related detail objects. For example, a Sales Order object might have a Sub Total field that summarizes the Line Total field in related line items. Again, roll-up summary fields, like all formula field types, are declarative and require no coding to implement.

### Validation Rules

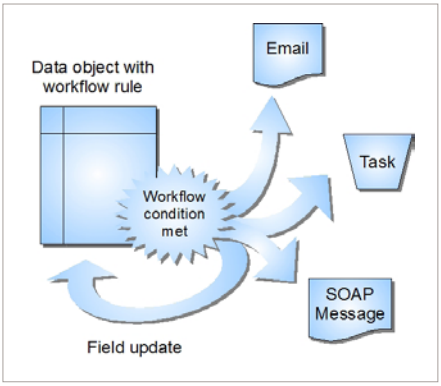
Data-centric applications require mechanisms to validate data to ensure data integrity. For example, it might be necessary to require a particular value in a certain field, that the dates in a Review Date field exclude weekend days, or that a Social Security Number requires the format "999-99-9999."

Using Force.com's builder, it's easy to implement a required field by checking a box when declaring the field in an object. Custom data integrity checks are also simple to implement by declaring validation rules that apply to the application's underlying objects. For each validation rule, a condition must be true for Force.com to generate an error; organizations can specify custom error message text to be automatically displayed when a user enters invalid data. The Force.com online help system includes a library of examples that makes it easy to get started with common validation rules.

### Workflows

Most organizations define sequences of logic and operations that correspond to standard business processes. By managing and automating these processes as much as possible, organizations can improve worker productivity and make their operations more reliable. Force.com's generalized solution for business process management is a feature called "workflow."

Workflows provide a simple way to extend objects with automated behaviors that simplify the development of Force.com applications. A workflow is an action that is bound to an object and is automatically triggered by inserting or changing a record in the object. A workflow can trigger a task, email alert, update a data field, or send a message to another application. For example, a workflow can automatically:



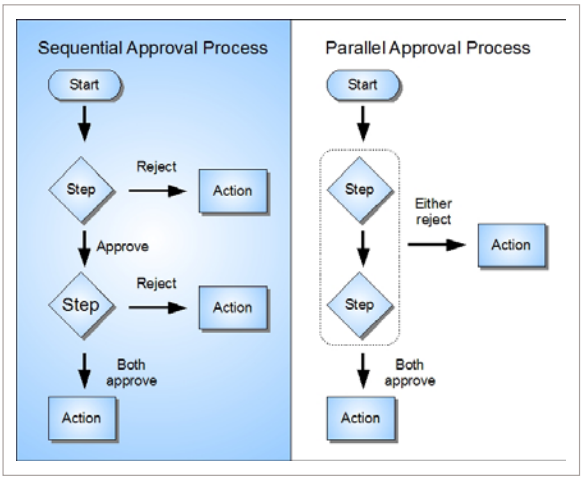

- Assign follow-up tasks to someone 1 week after a record update
- Send someone an email alert after inserting a record
- Change a record's Owner field at a specific date and time
- Trigger an outbound message to an external application system to initiate a related business process managed by the external system

Anyone can easily declare intricate workflows using the point-and-click builder workflow designer, without needing to write complicated code. A workflow rule declaration includes one or more criteria or a formula, either of which determines when Force.com triggers the workflow. A workflow rule can trigger one or more actions, any of which execute immediately or at a specific interval after the triggering event. The builder's workflow designer even makes it simple to specify non-trivial workflow actions; for example, to specify an email alert, a user picks an email template and selects recipients from a list.

### Approval Processes

An approval process is a complex, specialized type of automated workflow a Force.com application can use to model the approval processes for records managed by a business application. An approval process specifies the steps necessary to approve a record, who must approve the process at each step, and the actions to take after each step. An approval process also supports the locking of an approved record to avoid changes after the approval is complete. Let's take a closer look at each part of an approval process to understand how flexible this Force.com feature can be when implementing even the most complex business approval processes.

- An approval process can have one or more steps.
- Each step has a designated approver. Force.com makes it possible to either specify an approver or to use a look-up function that dynamically assigns the approver when someone creates a record.
- Each step can apply either to all the records that are part of the process or just to those with certain attributes.
- Multiple steps can happen in parallel or sequentially—for example, first an approval is sent to a department manager, then the department vice president, then the CEO. Developers can even use Apex to build dynamic routing options. These flow charts show sequential and parallel approval processes.




- Approvals can happen via an application UI action (for example, checking a box) or by someone sending an email.
- Each step triggers an action when someone approves, rejects, recalls, or first submits a record for approval.
- A step's action can be to send an email alert, create a task, update the value of a field, send an outbound message to an external application, or recall the record.

As with other Force.com features, developers and non-developers alike can build even the most complex approval processes using the simple point-and-click designer that is part of the builder. No coding is necessary.

### Apex and Complex Business Processes

For even greater flexibility, developers can use Apex to extend applications to include virtually any kind of business logic and functionality. Developers can encapsulate business logic in two different ways: as an anonymous stand-alone script that is executed on demand, or as a trigger that automatically executes before or after a specific database manipulation event (insert, update, delete, or undelete). In either case, Force.com compiles Apex and stores it as metadata. When an Apex routine is called for the first time by someone in an organization, Force.com's runtime interpreter loads the compiled version of the program into a cache for that organization. Thereafter, when any user from the same organization requires the same routine, Force.com can save memory and avoid

the overhead of recompiling the program by sharing the ready-to-run program that is already in memory.

Apex is much more than just another procedural language—it's an integral Force.com component that helps the platform deliver reliable multitenant applications. For example, Force.com automatically validates all embedded data access statements within an Apex class to prevent code that would otherwise fail at runtime. The platform then maintains corresponding object dependency information for valid Apex classes and uses this information to prevent changes to metadata that would otherwise break dependent applications.

To prevent malicious or unintentional monopolization of shared, multitenant platform resources, an extensive set of governors and resource limits is associated with Apex execution. For example, Force.com closely monitors the execution of an Apex script and limits how much CPU time it can use, how much memory it can consume, how many queries and DML statements it can execute, how many math calculations it can perform, how many outbound Web service calls it can make, and much more. Individual queries that the platform's optimizer judges to be too expensive to execute result in a runtime exception message to the caller. Although such limits might sound somewhat restrictive, they are necessary to protect the overall scalability and performance of the shared platform. In the long term, these limits help promote better coding techniques among platform developers and create a better user experience for everyone.

Because the Apex language syntax is similar to Java, many organizations will find it easy to get started building Force.com applications. For example, here's the Apex Code to create a new record in a fictional Account object:

```
Account a = new Account( name='Acme', billingCity='Edinburgh')
```

To query objects within Apex, the Force.com platform supports two query languages:

- The **Salesforce Object Query Language** (SOQL) is a query-only language. Although similar to SQL in some ways, it is an object query language that uses relationships, not joins, for more intuitive data navigation.
- The **Salesforce Object Search Language** (SOSL) is a simple language for searching across all persistent objects.

Developers can embed either type of query directly in Apex. For example, the following code retrieves all matching Account records and assigns them to an array using SOQL. Note the use of the embedded Apex Bind variable in the following query, which allows for statement sharing at application runtime, resulting in less overhead and better scalability.

```
String myName = 'Acme';
Account [] accts = [select ID from Account where name=:myName];
```

Apex can also contain data manipulation language (DML) operations to retrieve, insert, delete, and update data in the database, as well as to create and use save points to manage transactions. Here are some examples:

```
Acccount a = new account[] {new Account(name='foo'), new
Account(name='bar')};
insert a;
Savepoint sp = Database.setSavepoint();
delete a;
Database.rollback(sp);
```

Finally, here's a sample trigger definition:

```
trigger myAccountTrigger on Account (before insert, before
update) {
  if (Trigger.isInsert) {
```

```
      // do something
    }
    if (Trigger.isUpdate) {
    for(Account a: Trigger.new)
      if (a.name == 'bad')
        a.name.addError('Bad name');  // prevents  update
    }
  }
```

This example trigger fires before a user or application process inserts or updates an Account object. The Trigger.new array provides access to the target array of Account objects. Update and delete triggers can use the Trigger.old array to refer to old versions of the target objects.

To avoid potential platform problems introduced by poorly written Apex, salesforce.com strictly manages the deployment of a new production application. Before an organization can transition a new custom application from development to production status, salesforce.com requires unit tests that validate the functionality of the application's Apex routines. Submitted unit tests must cover no less than 75 percent of the application's source code. Salesforce.com executes submitted unit tests in the Force.com Sandbox environment to determine whether the application will adversely affect the performance and scalability of the multitenant population at large. The results of an individual unit test show basic information, such as the total number of lines executed, as well as specific information about the code that was not executed by the test.

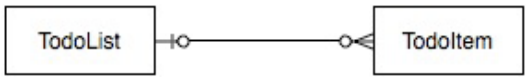
## User Interface as a Service

Force.com application providers have two tools available for creating on-demand applications: Force.com's builder and Visualforce.

### Force.com's Builder

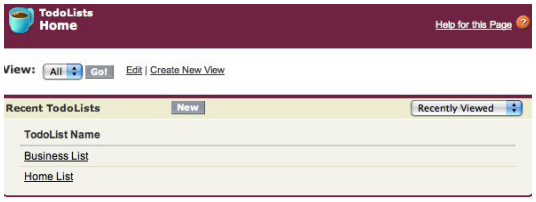
The builder is a declarative Web interface that makes it easy for anyone to create database objects and configure other aspects of the platform, such as workflows, Web services, and email services. The interface creates metadata, which Force.com uses to generate a default UI for each database object, with associated list, detail, create, edit, and delete pages. Using this declarative approach, users can create an entire application, including the persistence layer, without any programming. What follows is an illustrated tour of the builder environment that shows some of the resulting pages that comprise an application built with the builder.

As an example, consider two simple custom objects, TodoList and TodoItem. TodoItem has a lookup relationship field to the TodoList object. Furthermore, it has a Due Date (of type Date) and Description (of type Text Area) field.




It is possible to create these two objects with Force.com's builder within 3 minutes by simply by pointing and clicking. As developers create new objects like TodoList and TodoItem, Force.com automatically generates UIs for each object—absolutely no additional work is needed to build these UIs.

For example, the builder automatically generates list pages for each object and uses these pages as the default view for the tab associated with the object. The default list page for the TodoList object is shown with a display of the most recently accessed objects.

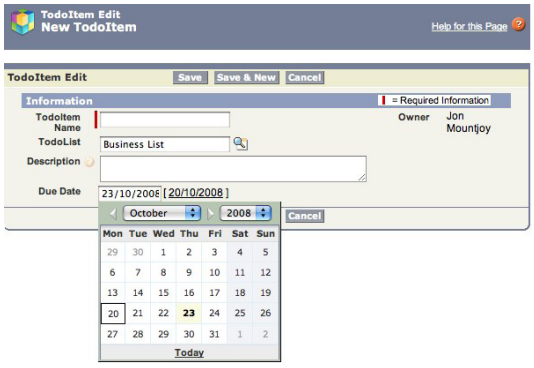


Users can also create and use alternative views for an object. For example, after pressing the Go button in the list above, this page becomes available.


The pages provide intuitive access to records, buttons for actions such as creating a new record, and, in the case of lists, the ability to perform actions on groups of records. For example, users can create new records with the default data entry page for the TodoList object. Required fields are automatically identified with a red bar.


Here is the resulting detail page for a TodoList record after entering "Business List" as the name. Note the default Edit, Delete and Clone action buttons. Each object has a number of standard fields created automatically by the builder, including a Last Modified By, Created By, and Owner field. The values of the Created By and Last Modified By fields, which Force.com includes for all objects, are automatically set to read-only.

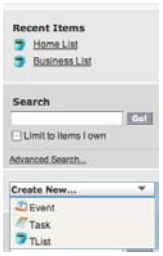

Here is the default page generated by the builder when a TodoItem record is created. The UI generation uses the type information for each field to display an appropriate input component. For example, Force.com renders date-oriented fields such as Due Date with a pop-up calendar. Note that because TodoItems are in a 1:many relationship with TodoList records, the interface automatically lets users look up a related TodoList item from this creation page as well. The search icon next to the TodoList field pops up a dialog box so that users can search for the TodoList they want to relate to the current TodoItem.



Here is a filled-in TodoItem record—a detail page. Note that native UIs automatically include hyperlinks whenever they display the Name field for another record. For example, when someone clicks on this page's "Home List" link, the application redirects to the "Home List" detail page. Thus, natively generated UIs automatically provide an intuitive surfacing of target relationships, generating not only pages for each object, but also the navigation between them.


Interfaces can also display a list of detail records on the associated master page. For example, here is a detail page for an item that includes detail records.


The default UI for all native Force.com applications includes a sidebar, as shown here. With this list of items, which can include items from more than one type of object, users can quickly access recent items. Users can also search through all objects in the database and configure the sidebar to create new records of a particular object type.




With Force.com, users can associate tags with objects. For example, after enabling the tag functionality, users can create and modify tags for each record. Users can then find other objects that have those tags and even permit public tagging—so that any users who can access the record can add their own tags.


Finally, the Force.com platform provides a rich reporting environment. Application designers can create tabular, matrix, and summary reports as well as graphical representations, including bar graphs and pie charts.


Anyone can easily modify generated, native builder UIs. Page layouts define the appearance of various input and list pages. Various other page modifications are also possible, such as adding search functionality. For more complex customizations, however, application developers can use the Visualforce UI layer to generate pages.

**Visualforce**

Visualforce is a complete framework for creating cloud application UIs and making it possible to design, build, and deliver any kind of interface and interaction entirely in the cloud. The UIs made possible with Visualforce can extend the look and feel of standard builder applications or replace them entirely with a unique style and a set of sophisticated interactions.

With Visualforce, developers can use standard Web development technologies—including HTML, AJAX, and Adobe Flex—to create UIs for their cloud applications. Through a Model-View-Controller (MVC) model[1], developers can wire together these interfaces with Apex. As a result, Visualforce offers not only pixel-level control over the definition of an interface, but makes it possible to create new and sophisticated user interactions based on wizards, branching, and other UI logic.

As the Internet and associated Web standards become ubiquitous—appearing on virtually every desktop and increasingly on every device—the opportunities for delivering cloud applications are growing as well. Tools such as Visualforce support this trend by enabling UIs that support a growing set of contexts from which users access these applications. For example, approaches to cloud-based applications may range from simply customizing a native builder application page to creating new applications that will run on kiosks or mobile devices such as an iPhone. In each of these use cases—either extending the current look and feel of an existing interface or creating a new UI—Visualforce provides the features to make those tasks as easy as possible.

At a high level, examples of applications well suited to Visualforce include these:

- **Native Force.com builder application customization** – With more than 75 predefined UI components, Visualforce makes it easy to extend the standard Force.com look and feel to include new kinds of user interactions, such as wizards and multi-object screens. Instead of implementing custom cascading style sheets (CSS) to emulate the Force.com interface, developers simply invoke common components such as page headers, fields, and related lists via special <apex:tags>. Just as useful is the capability to selectively override specific interface pages—such as a standard edit page or a custom tab's home page—while leaving the standard interface intact for all other functions. This fine-grained control over an application's interface provides the best of both worlds, so that developers can choose the automatically generated UI for some areas and use Visualforce to selectively modify others.
- **New Web and database applications** – Although there are many kinds of applications that can be built on the database, integration, and logic core components of the Force.com platform, not all applications fit the platform's default, out-of-the-box "database browsing" metaphor. For such applications, Visualforce provides the full-page control needed to create different kinds of metaphors. Although the resulting applications may look and behave differently from natively generated applications in every way, they can share the service delivery infrastructure and other services of the Force.com platform.
- **New devices** – The full control provided by Visualforce is also ideal for bringing cloud applications to a new generation of Internet-enabled devices. By being able to shape the presentation of data to a specific context—for example, by making it appear in one format when viewed on an iPhone and in another in a desktop Web browser—Visualforce makes it possible not just to deliver the right application to the right device, but to deliver a single application to multiple devices, with each device providing different views into the same data.

With Visualforce, salesforce.com delivers the first cloud-based implementation of an MVC architecture. In Visualforce, the MVC model is implemented with standard and custom objects—the staples of Force.com development—as well as with three new objects: pages, components, and controllers.

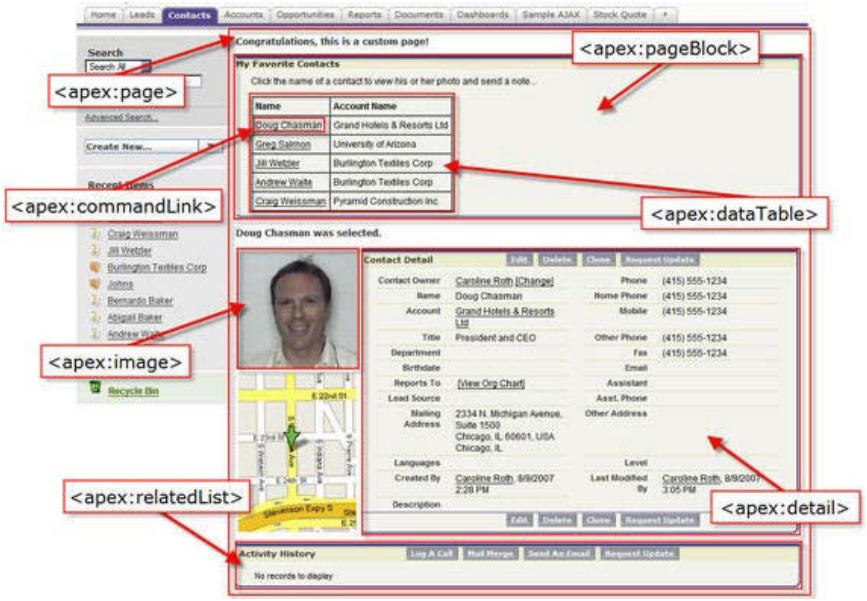
- **Pages**, along with components, are the basic creative building blocks for application designers. Similar to a standard Web page, a Visualforce page uses HTML, CSS, and DHTML to specify the appearance of the application's interface, with the option of using other Web technologies such as CSS, AJAX, and Adobe Flex for additional flexibility. Pages have a unique URL for access, just as they would on a traditional application server. Visualforce components complement this standard markup. These components, which are similar to tag libraries in other systems, make it possible to invoke complex components with a single line of Visualforce.

[1] MVC is a widely used interface architecture based on the separation of data presentation from data manipulation. In such an architecture, the "model" represents the data model, the "view" represents the presentation of the data (UI), and the "controller" represents the business logic that manipulates the data and controls the UI.

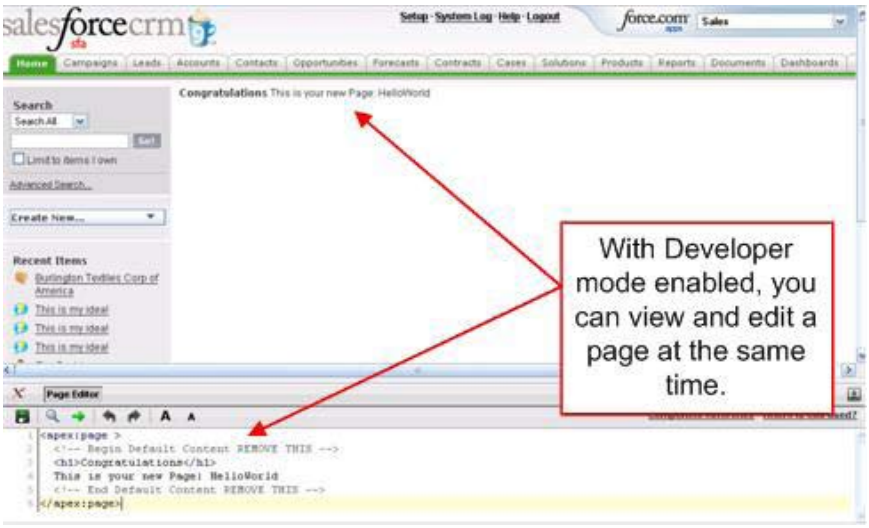
Because Force.com servers render these components and then deliver the fully composed page to the client, users get better performance and enhanced functionality compared to client-only techniques.

- **Components**, built with special markup <apex:tags>, are the key for reusing common interface elements and for binding both standard and custom elements to data. Visualforce has more than 75 predefined components that developers can assemble with minimal coding, in typical HTML building-block fashion. Some components implement common Salesforce CRM interface elements, while others make available new features such as AJAX-based partial page refreshes. Components may provide various levels of granularity, such as displaying multiple lines of data or executing embedded functionality of an application's metadata on a Visualforce page. Style sheets can even include components, making it easy to change a component's style regardless of how it was created.
- **Controllers** are the basic functional building blocks that control the application's logic. Implemented in Apex, controllers provide the underlying business rules for the interface as well as the "connective tissue" between the application's page presentation and the underlying data. Any given page interacts with a given controller through components, which bring in the data to be displayed in the interface and send it out to for storage in the database. The controller provides access to the data and specifies what happens when the user interacts with an interface component. Visualforce provides prebuilt, standard controllers for standard interactions such as view, edit, and save, which developers can implement without additional coding. Developers can encapsulate new behaviors that go beyond predefined interactions—called custom controllers—by using Apex to access new data sources, navigation elements, and behaviors. Because custom controllers can maintain state across page interactions, it is possible to construct interactions such as wizards, whose logic spans steps on multiple pages.

Visualforce markup defines which interface components to include on the page and what those components look like. Because Visualforce markup is ultimately rendered into HTML, designers can use Visualforce tags alongside standard HTML, JavaScript, Adobe Flex, or any other code that can execute within an HTML page. The example below shows the simple Apex tags that correspond to the associated page.



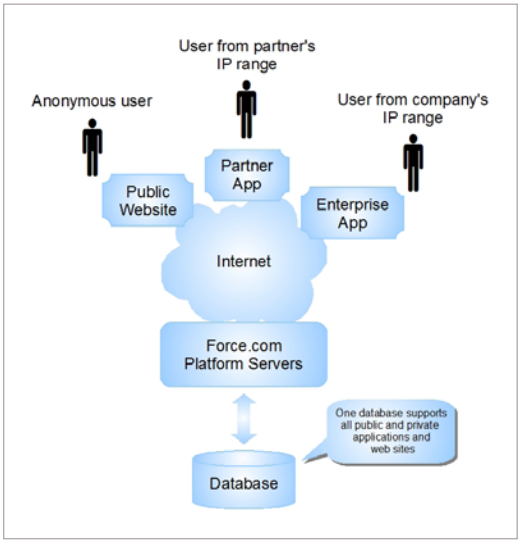

As this screen shows, developers can edit an application's markup in the same window that shows a preview of the display, so they can instantly verify how a particular edit affects the interface by simply saving the code. The Visualforce development mode editor pane also includes auto-completion, syntax highlighting, and "quick fix" features so developers can create components on the fly.




**Force.com Sites**

The security policy of a traditional on-premises business application environment typically doesn't allow Internet access to internal databases that operate behind secure firewalls, even though such data access might be useful to customers and partners. Instead, most organizations configure separate systems for internal and external users and then undertake complex and time-consuming integration projects to ensure the secure transfer of information between systems.

With Force.com Sites, organizations can reduce the cost and complexity of IT operations by building both internal and external-facing applications and Web sites based on the same underlying database, securely and effortlessly, without the need for separate systems and complicated integrations. For example, the same company database that manages products, customers, sales orders, and technical support requests on behalf of internal applications can also power the company's public Web site with product catalogs, customer surveys, sales promotions, and knowledge bases. Data modifications, such as price updates, are immediately available to both internal and external users. There is no need for any type of data replication or transfer because the same centralized database serves the needs of both public and private applications.




Organizations that already use Force.com to deliver internal applications can quickly build and deploy external Web applications and pages as well. Because Force.com Sites is based on existing Force.com technologies such as the Force.com Database and Visualforce, there is almost no learning curve. There are four simple steps for building Force.com Sites:

1. Build the application and its database schema using Force.com's builder.
2. Design a Web interface using Visualforce.
3. Create a Force.com Site with the URL of your choice.
4. Activate the site.

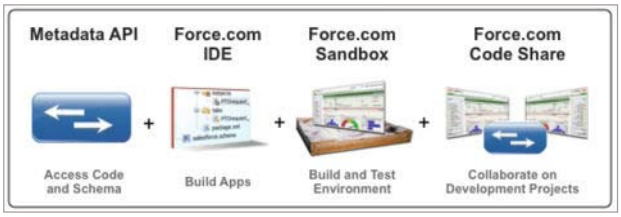
Force.com Sites fully supports the branding of an organization's Web pages. An organization can use Force.com Sites to expose any information within in its Force.com database through a URL of its choice; even URLs based on a custom domain name rather than the "force.com" domain. And standard Visualforce features make it easy to build site pages that match the look and feel of the organization's brand.

With the security options available for Force.com Sites, organizations can build applications and Web sites that meet different types of business requirements. At the most granular level, organizations can create "public" sites that permit wide-open, anonymous access. Force.com Sites can also support intranet-style applications that use IP address-range filters to limit access to connections that originate from, for example, an organization's employees or a trusted partner. For even more security control, Force.com supports secure authentication schemes for sites that require user logins. Of course, organizations can also use standard Force.com security features to control which data objects and fields are available in any particular scenario.

Force.com Sites also supports most of the traditional features businesses expect from a Web server. For example, a Force.com Ste can have a robots.txt file that controls how search engine robots and Web spiders access site pages. Syndication feed support lets users subscribe to updates about Force.com Sites in external feed readers. And developers can improve site performance by using simple tags that control page caching.

## Development as a Service

Development as a service (DaaS) is a suite of Force.com features that lets developers use traditional practices for creating cloud applications. These features include the Force.com Metadata API, Force.com Integrated Development Environment (IDE), Force.com Sandbox, and Force.com Code Share. DaaS expands the cloud-computing development process to encompass external tools such as integrated development environments, source control systems, and batch scripts to facilitate development and deployment.

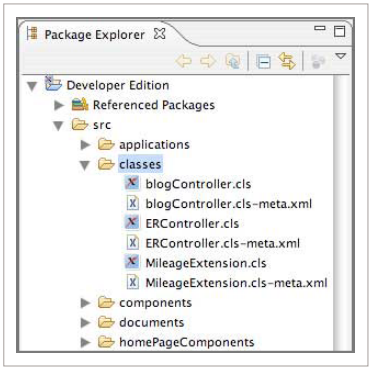



### Force.com Metadata API

The Force.com platform enables DaaS by providing a rich and powerful metadata model and API. The Force.com Metadata API provides an alternative to the platform's interface by allowing developers to directly modify the XML files that control their organizations' metadata. Developers can also use the Metadata API to migrate configuration changes between organizations and create tools for managing organization and application metadata.

### Force.com IDE

The Force.com IDE is a free, powerful client application for creating, modifying, and deploying Force.com applications. Based on the Eclipse platform, it provides programmers with a familiar environment they can use to code, compile, test, package, and deploy applications, all from within the IDE. Much of the actual work, such as compilation, happens on the Force.com platform. The Force.com IDE performs the communication and result parsing transparently.



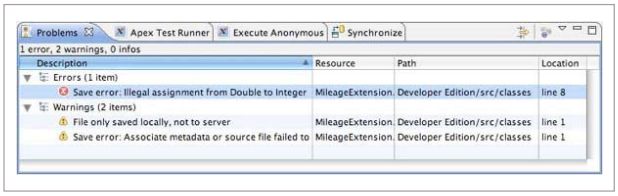

The Force.com IDE also features an embedded schema explorer for a live view of an application's database schema and a browser for viewing and editing metadata components. Using the standard synchronization features of the IDE, multiple developers can share a source code repository. This section provides a brief tour of the Force.com IDE and its features.

The Force.com IDE includes a number of features that facilitate code development. For example, like most integrated development environments, the Force.com IDE organizes application resources into projects. However, unlike traditional software development projects in which source code is compiled to create executable binaries, the resources in a Force.com project live within a Force.com organization and are copied into the local project for editing. The Package Explorer in the IDE shows a typical Force.com project.

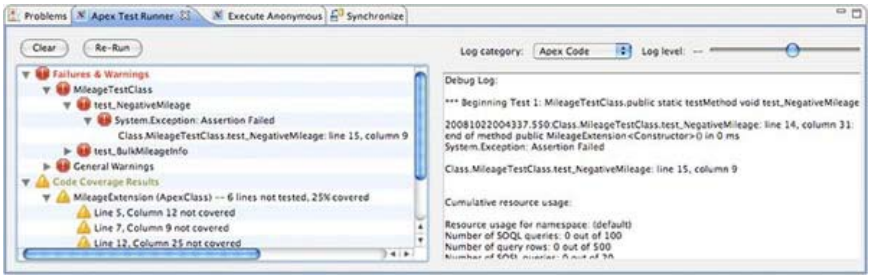
The Force.com IDE provides a code editor for adding, modifying, and testing Apex. A developer can use this editor to write database triggers as well as any kind of Apex class such as Visualforce controllers, email handlers, or classes holding business logic. The editor includes a number of features:

- Syntax highlighting for Apex keywords
- Code assist for static types and schema objects
- Click-to-line integration with errors and testing views
- Unlimited undo and redo
- Tabbed views of source code and XML metadata definitions
- Individual class or complete test-code execution

When a developer saves Apex Code, the IDE automatically sends it to the server for compilation. The Problems view displays any errors that result from that compilation. Double-clicking on an error in the Problems view goes right to the file and line where the problem was detected. Once the problem is fixed, the developer can save the file to send it back to the server, which updates the Problems view automatically.



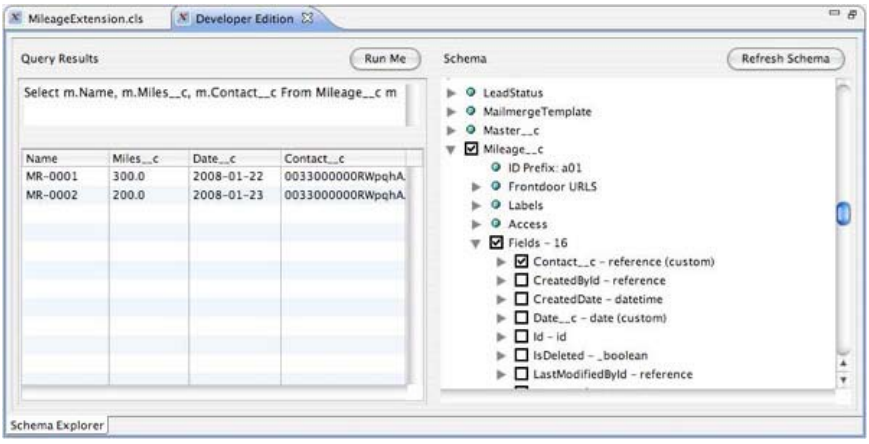

Earlier, this paper mentioned that Apex includes the ability to define and execute unit tests—pieces of code that verify that an application works as intended. To deploy to a production application, all Apex classes and triggers need at least 75 percent code coverage from tests. The definition of each unit test is a test method, which can be part of any normal Apex class or a special Test class. The Force.com IDE provides an Apex Test Runner view to test methods to see which tests are passing or failing. All Apex test execution occurs on Force.com servers.




In addition to Apex classes and triggers, other types of metadata can be downloaded into Force.com projects, including Visualforce pages, schema objects, and workflow rules. These components are represented as XML documents and open in an editor that is aware of available XML tags for each component type. Because the metadata files are XML-formatted and text based, they can also be compared using text diff tools, managed in a version control system such as CVS or Subversion, and even deployed from one organization to another.

The Schema Explorer is a tool for browsing the objects and fields in a Force.com organization. The browser presents the logged-in user's view of the Force.com Database model, including object visibility, permissions, data types, lookup values, and other information useful in developing applications on the Force.com platform. The Schema Explorer is shown as a hierarchical tree view

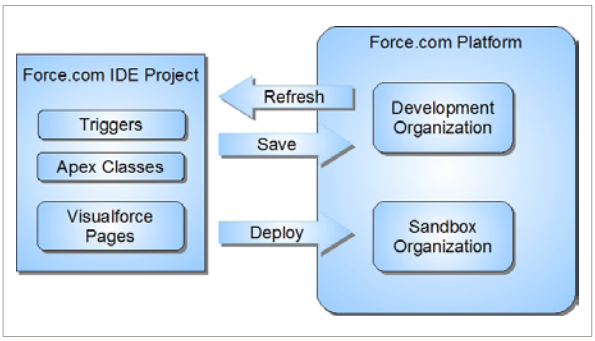
of an organization's schema. It is read only and does not interact with the rest of the workspace in any way.




The right side panel of the browser is a hierarchical tree view of all objects in the database. A developer can use this view to drill down to an object of interest and select fields within that object. As a result, the query pane on the left is filled automatically with an SOQL query that returns those objects and fields. A user can also execute an ad-hoc SOQL query and display its results.

In a typical working environment, the Force.com IDE is in communication with the Force.com platform servers. For example, when a developer saves an Apex class, the IDE sends the Apex class to the Force.com servers. The servers compile the class and return the results (for example, parsing errors or compilation errors) to the IDE, which then displays them. This approach is the typical mode of operation—online mode. Projects can also be in offline mode. In offline mode, any changes a developer makes are simply made on the local machine. When the IDE goes back online, it synchronizes those changes.

Once an organization creates application components and tests them in a development environment, it can migrate them to a different organization for testing, staging, publication, or production use by end users. The Force.com IDE facilitates deployments and is highly configurable. It only takes a few mouse clicks to deploy a project or a subset of a project to an organization.




Once developers determine what to deploy, they can either validate the deployment—fully executing the deploy process on the server, including running tests, but without actually saving changes—or execute the deployment and save all changes. If the deployment fails for any reason, the IDE displays a list of each error or test failure.

**Force.com Sandbox**

The Force.com Sandbox provides a separate cloud-based application environment for development, quality assurance, and training. It is also useful for testing new customizations and for building and testing integrations and custom applications.

Because the Force.com IDE is built on the Eclipse platform, developers can use standard Eclipse features such as version-control support. With a version-control system, a team of developers can track changes to a set of files over time. Using Eclipse's version-control features in tandem with the Force.com IDE makes it possible to store Force.com applications in source control, compare current and historical versions of a component, roll back changes, and work collaboratively by sharing a single application definition with an entire development team. Eclipse can connect to

the popular, open-source Subversion system using a plug-in called Subclipse. Once it is installed, a developer can retrieve source code from a Subversion source code repository, including Code Share projects created by the Force.com developer community.

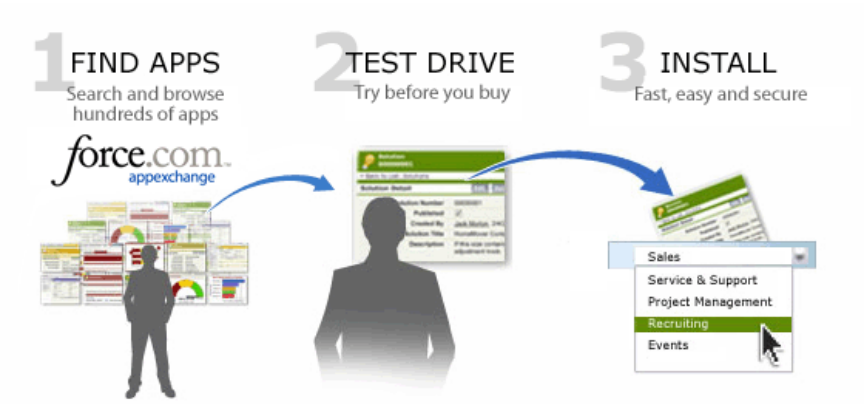
**Force.com Code Share**

With Force.com Code Share, developers across engineering organizations—or around the world—can collaborate on the development, testing, and deployment of cloud applications. Because Code Share integrates with most standards-based source control management systems, developers can store the definitions of their Force.com applications in source control and deploy those applications in either their Force.com Sandbox or production environments.

## Force.com AppExchange

The Force.com AppExchange is a cloud application marketplace that makes it easy for IT managers and business leaders to find SaaS solutions that are certified as reliable, secure, and interoperable to meet their business needs. The AppExchange currently provides access to 800+ cloud-based applications, resulting in almost 300,000 test drives and more than 40,000 application installations.

The AppExchange not only makes it easy to try new applications, but mitigates the risk of installing an application that ultimately doesn't meet an organization's needs. Related features let users leverage input from the AppExchange community to accelerate application deployments and maximize business benefits. Organizations can install applications easily via the Web and then customize those applications either declaratively, using the builder, or programmatically, using Apex.




Salesforce.com's certification team evaluates each ISV's applications to ensure they meet the industry's most rigorous performance and security standards. These standards met the requirements of leaders in the financial services community, such as Chrysler Financial and E*TRADE FINANCIAL. In addition to evaluating an ISV's application, salesforce.com may also test and evaluate the ISV's network and hosting capabilities. The AppExchange also leverages the best-practice principles of today's social networks. These approaches encourage continuous innovation among ISVs and let customers interact with vendors and with one another using feedback, recommendations, and blogs. An application that passes these tests is added to the AppExchange directory, with an "AppExchange Certified Application" logo displayed next to its listing. ISVs must re-certify their applications annually to retain their certification status.

IT and business decision-makers can easily find specific applications on the AppExchange by using the Search Apps bar on nearly every page or by choosing a category from the site's drop-down menu. Categories cover a wide range of industry segments and business functions, from industries such health care and manufacturing to functions such as marketing and human resources.

## Conclusions

As the cloud-computing paradigm shift gains momentum, more and more application builders, both commercial vendors and enterprise architects, are looking to deliver cloud-based applications to avoid the inefficiencies of the traditional on-premises application model. However, developing cloud-based applications requires both a new mind set and new tools to meet the unique challenges of the model.

Force.com is the world's first cloud application platform. It is also the most mature, reliable, secure, and scalable platform that can be used by anyone to manage the entire life cycle of a cloud-based application. The tightly integrated, comprehensive stack of Force.com technologies is engineered specifically to meet the operational and technical requirements of delivering applications in the cloud.

- A global, secure infrastructure composed of disparate data centers provides redundancy to ensure the constant availability of cloud-based applications.
- The Force.com Database and its metadata-driven architecture are the core technologies that enable the platform to deliver robust, configurable, and scalable multitenant applications.
- Open, standards-based integration solutions allow Force.com applications to interact with other systems, including on-premises applications. Force.com connectors provide ready-made solutions to many common application integration requirements. When necessary, developers can build custom integrations using the platform's core Web services API, Apex Web services, callouts and mashups, and outbound messaging.
- Force.com objects can include associated, centralized logic to help automate and ensure consistent behaviors that correspond to business logic. Anyone can build formula fields, rollup-summary fields, validation rules, workflows, and approval processes, without any code whatsoever. For more complex business processes, developers can use Apex Code, salesforce.com's programming language, to build triggers that implement and automate virtually any type of business logic.
- Force.com automatically generates native application UIs for all objects. These interfaces can be declaratively modified by anyone, using the builder, Force.com's integrated application development tool. For more customized control, developers can use Visualforce to modify or completely override the look and functionality of native application UIs.
- With the Force.com Metadata API, Force.com IDE, Force.com Sandbox, and Force.com Code Share, developers can build powerful applications using tried-and-tested tools and methodologies, including source control systems and batch scripts that streamline development and deployment.
- Commercial application providers can market their offerings on the Force.com AppExchange, the cloud application marketplace. The AppExchange makes it easy for potential application users to shop for, try out, purchase, and install applications that meet their business needs, much like music lovers shop for songs on iTunes.

All Force.com platform features are delivered "as a service," thus relieving application providers from the traditional headaches of purchasing, installing, configuring, and maintaining software. Now, with the Force.com platform, developers and business analysts can concentrate on building award-winning applications and leave the heavy lifting to salesforce.com.

**Corporate Headquarters**
The Landmark @ One Market
Suite 300
San Francisco, CA, 94105
United States

1-800-NO-SOFTWARE
www.salesforce.com

**Latin America**
+1-415-536-4606

**Japan**
+81-3-5785-8201

**Asia/Pacific**
+65-6302-5700

**Europe, Middle East & Africa**
+4121-6953700

salesforce.com Success. Not Software.