

Apex Code: The World's First On-Demand Programming Language



Contents

Extending the Power of the Apex Platform 1

Multi-tenancy and Programming Languages..... 1

Apex Code Design and Syntax..... 2

Apex Code Example2

Features and Functionality 2

Online Salesforce.com Developer Resources 3

Extending the Power of the Apex Platform

The Apex platform has been a core part of Salesforce deployments for years—supporting thousands of custom applications and integrations with all the benefits of the on-demand model. The platform encompasses a complete feature set for building business applications such as data models and objects to manage data, a workflow engine for managing collaboration of data between users, a user interface model to handle forms and other interactions, and a Web services API for programmatic access and integration.

New to this platform is Apex Code, the world’s first on-demand programming language.

Apex Code extends the powerful and proven success of the Apex platform by introducing the ability to write code that runs on salesforce.com servers. This language makes possible the development of a new class of applications and features deployed entirely on demand. These applications infuse intelligence into existing apps by capturing business logic and rules—such as data validation—and enable entirely new kinds of apps on demand—such as complex inventory checking and order fulfillment.

Among the concepts behind Apex Code, the following provide an especially good introduction to the language’s potential and implementation.

Multi-tenancy and Programming Languages

Like all aspects of Salesforce application and platform technologies, Apex Code is “on demand,” running exclusively on salesforce.com servers without requiring any local servers or software. And unlike on premises technologies, the language runs in a multi-tenant environment, providing the economic and manageability benefits of a shared service, while keeping the definition, data, and behavior of each customer’s application entirely separate from each other. For developers, this combination delivers the best of both worlds: the convenience, scalability, and safety of an on-demand database, *and* the flexibility and control of a procedural language. In other words, Apex Code allows developers and IT organizations to focus on innovation, not infrastructure.

Unique in both design and implementation, Apex Code uses differ from other programming languages like C++, Java, and .NET. Unlike these, Apex Code is much more limited in scope. Whereas a language and platform like Java can be used to build almost any kind of application—from a video game to a vending machine—Apex Code is used exclusively to build business applications that manage data and processes within the larger context of the Apex platform framework. But within this scope, Apex Code offers a uniquely powerful and productive approach to creating functionality and logic that allows developers to focus only on the elements specific to their application, while leaving the rest of the “plumbing” to the platform’s framework. And where languages like Java and C++ execute at a low level—utilizing as many and as much system resources as the underlying hardware can support—Apex Code is abstracted and governed, utilizing only as many resources as is allowed.

This “governor” is essential. Until now, multi-tenancy has been associated strictly with data: a multi-tenant database allows data operations for one “tenant” to be completely isolated from the other tenants. But those operations have not included complex business logic, because even a few ill-considered lines of code can create a resource-consuming “monster” that can potentially impact all applications and the people using them. If a piece of code goes into an infinite loop or wants to fetch a billion records, the rest of the system can grind to a halt. The breakthrough for Apex Code is that it isolates and contains such behavior.

The technology breakthrough came in the form of an additional layer of abstraction—a sort of “virtual, virtual machine” that governs the executing of a given component of Apex Code. This layer of abstraction is a remarkable achievement—enabling the first business programming language that runs in a multi-tenant environment. This multi-tenant virtual machine monitors the code execution, including the total number of data queries issued, the amount of data retrieved, the number of data operations and transactions performed, and the number of control structures like loops that are executed. By constraining and monitoring the code at all times, the Apex “virtual, virtual machine” enables the flexibility and power of a programming language, without sacrificing the safety and control required for a multi-tenancy environment.

Apex Code Design and Syntax

Apex Code is designed explicitly for expressing business logic and manipulating data, rather than generically supporting other programming tasks such as user interfaces and interaction. Apex Code is therefore conceptually closer to the stored procedure languages common in traditional database environments, such as PL/SQL and Transact-SQL. But unlike those languages, which due to their heritage can be terse and difficult to use, Apex Code uses a Java-like syntax, making it straightforward for most developers to understand. And like Java, Apex Code is strongly typed, meaning that the code is compiled by the developer before it is executed, and that variables must be associated with specific object types during this compile process. Control structures are also Java-like, with loops and iterators borrowing that syntax directly.

Because Apex Code is a process and data language, developers will primarily interact with APIs to query, manipulate, and save information in their custom and standard objects. Developers can select data using the existing Salesforce Object Query Language (SOQL) syntax already found in the existing Web services API, as well as a new addition to that syntax that can retrieve information from multiple objects via a single query. In general, interacting with the Web service API via a language like Java is very similar to the experience of interacting with the data APIs in Apex Code.

This following Apex Code example illustrates both the language syntax and how it can be used. The code defines a trigger that prevents null fields, as well as duplicate pairings of email address and zip code, from being entered into the system.

Apex Code Example

```
// Define a trigger for the Lead object, and set it to
// fire on the insert and update events
trigger leadDupCheck on Lead(before insert, before update) {

    // Make sure the email and zip code fields aren't null
    if (Trigger.new.Email != null && Trigger.new.PostalCode != null ) {
        // Query for all leads that have the same email and zip code
        // as the new lead
        Lead[] dup = [select id from Lead WHERE email = :Trigger.new.Email
                      AND PostalCode = :Trigger.new.PostalCode ];

        if ( dup.size() > 0 ) {
            // Lead exists, so display an error message
            Trigger.new.email.addError('Lead is a duplicate');
        } else {
            // No dup is found, so let the processing continue
        }
    }
}
```

Figure 1. Apex Code Example

In this example, one can see some key concepts at work: the Java-like syntax of the variable declaration and if statement, the use of SOQL to retrieve data just as one would with the Web service API, and the use of the trigger declaration to define the trigger's scope, that is, what events it acts upon.

Features and Functionality

In addition to the essential capabilities of running on demand in a multi-tenant environment, Apex Code brings a number of other features that greatly expand the power developers have in using the Apex platform and in the range of applications they can build.

Apex Code and event model. Apex Code can be tied to the execution of the platform, enabling developers to exert fine-grain control over an application. When thinking about the Apex code, it's useful to consider the analogy of stored procedures and triggers, since the language is fundamentally tied to behaviors on the data, as opposed to providing a higher level UI language or representation. Hence developers can tie Apex Code into almost every aspect of an application's behavior: overriding the behavior in existing buttons, creating a new button, manipulating the control of a custom link, programming the control of an inline S-control, or even overriding the behaviors associated with a related list and data.

Consider an app that enables the user to create a new lead in Salesforce by clicking the save button to commit that record to the database. With Apex code, developers can create and execute code residing on salesforce.com's server to intercede just after the button is clicked. The code might check for any duplicate records, and if it finds any, implement a data quality scenario that notifies the user. Otherwise, the record commits to the database as is normally the case. (See Apex Code Example above.)

Transaction control. Because Apex Code is closely bound to Salesforce data, developers can readily add transactional features to their applications. For example, if one user is referencing a field while somebody else is trying to delete it, the system is aware of the conflict. Apex Code also features data commits and rollbacks, which are especially important when working across multiple objects.

Packaging, re-use and Web services. Apex Code uses a packaging model similar to that of Java, in which reusable packages of code can be invoked from each other or from within triggers. Unlike Java, however, Apex is not object-oriented in the sense that those packages can be modified through inheritance. Significantly, any method defined in a package can optionally be automatically exposed as a Web service, and thus can be invoked via the Web service API or directly through the AJAX toolkit.

Performance, scalability and upgrades. Because Apex Code runs on demand, scalability, compatibility, and maintenance issues are salesforce.com's responsibility, not yours. Apex-developed applications can scale indefinitely to support additional users, without your having to deploy additional servers. Applications potentially run faster because a single query can obtain information from multiple objects.

When newer versions of Salesforce and the Apex code itself are introduced, your code is never rendered obsolete. Salesforce.com ensures backward compatibility by maintaining processor-specific versions of Apex virtual machines, which in turn correspond to the API. As a result, your code continues to operate without modification.

Apex Code and the AppExchange. Apex Code can be packaged alongside custom objects, S-controls and other platform features, allowing developers to redistribute their Apex Code-enhanced apps via the same AppExchange directory available today.

Online Salesforce.com Developer Resources

Salesforce.com has a complete developer site available at <http://developer.salesforce.com> with resources that help developers quickly create successful applications for the Apex platform. The site includes links to online demos, presentations, documentation, and code samples showing how Apex Code works in a variety of app development scenarios. Developers can also sign up for a free Developer Edition at their site to begin developing their on demand apps today.

For More Information

Contact your account executive to learn how we can help you accelerate your CRM success.

The Americas
The Landmark @ One Market
Suite 300
San Francisco, CA 94105
United States of America
1-800-NO-SOFTWARE
www.salesforce.com

Latin America
Alfonso Napoles Gandara 50
4th floor
Col. Santa Fe
Mexico City
Mexico 01012
+52-55-9171-1882
www.salesforce.com

Japan
Ebisu Business Tower 18F
1-19-19 Ebisu, Shibuya-ku
Tokyo, 150-0013
Japan
+81-3-5793-8301
www.salesforce.com/jp

Asia/Pacific
9 Temasek Boulevard
#40-01 Suntec Tower 2
Singapore 038989
+65-6302-5700
www.salesforce.com/au

Europe, Middle East & Africa
Ch. de la Dent d'Oche 1B
1024 Ecublens
Switzerland
+353-1-2723-500
www.salesforce.com

