

APEX - OVERVIEW

What is Apex?

Apex is a proprietary language which has been developed by Salesforce.com. As per the official definition, Apex is a strongly typed, object-oriented programming language that allows developers to execute the flow and transaction control statements on the Force.com platform server in conjunction with calls to the Force.com API.

It has a Java-like syntax and acts like database stored procedures. It enables the developers to add business logic to most system events, including button clicks, related record updates, and Visualforce pages. Apex code can be initiated by Web service requests and from triggers on objects. Apex is included in Performance Edition, Unlimited Edition, Enterprise Edition, and Developer Edition.



Features of Apex as a Language

Integrated

Apex has built in support for DML operations like INSERT, UPDATE, DELETE and also DML Exception handling. It has support for inline SOQL and SOSL query handling which returns the set of sObject records. We will study the sObject, SOQL, SOSL in detail in future chapters.

Java like syntax and easy to use

Apex is easy to use as it uses the syntax like Java. For example, variable declaration, loop syntax and conditional statements.

Strongly Integrated With Data

Apex is data focused and designed to execute multiple queries and DML statements together. It issues multiple transaction statements on Database.

Strongly Typed

Apex is strongly typed language. It uses direct reference to schema objects like sObject and any invalid reference quickly fails if it is deleted or if is of wrong data type.

Multitenant Environment

Apex runs in a multitenant environment. Consequently, the Apex runtime engine is designed to

guard closely against runaway code, preventing it from monopolizing shared resources. Any code that violates limits fails with easy-to-understand error messages.

Upgrades Automatically

Apex is upgraded as part of Salesforce releases. We don't have to upgrade it manually.

Easy Testing

Apex provides built-in support for unit test creation and execution, including test results that indicate how much code is covered, and which parts of your code could be more efficient.

When Should Developer choose Apex?

Apex should be used when we are not able to implement the complex business functionality using the pre-built and existing out of the box functionalities. Below are the cases where we need to use apex over Salesforce configuration.

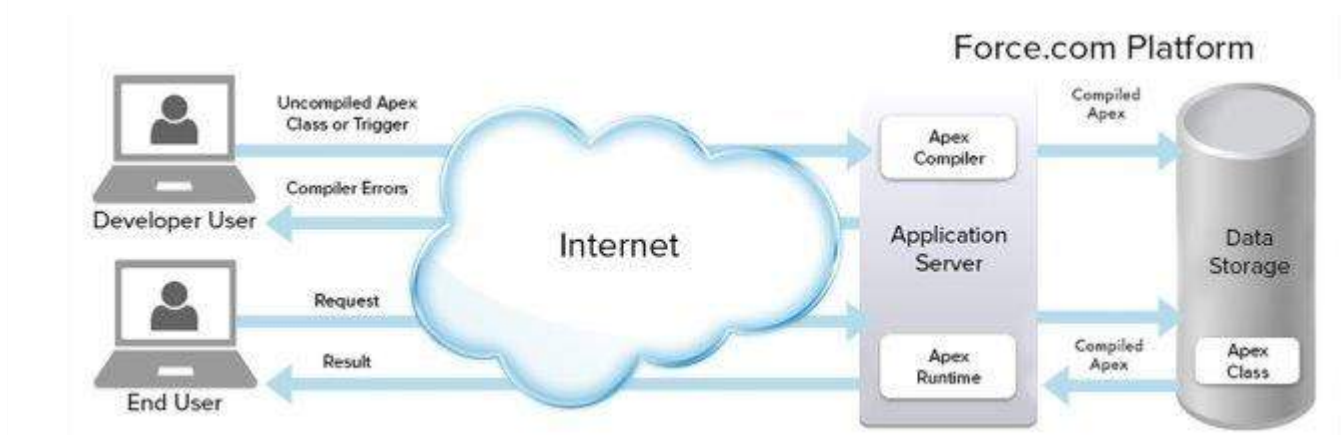
Apex Applications

We can use Apex when we want to:

- Create Web services with integrating other systems.
- Create email services for email blast or email setup.
- Perform complex validation over multiple objects at the same time and also custom validation implementation.
- Create complex business processes that are not supported by existing workflow functionality or flows.
- Create custom transactional logic *logic that occurs over the entire transaction, not just with a single record or object* like using the Database methods for updating the records.
- Perform some logic when a record is modified or modify the related object's record when there is some event which has caused the trigger to fire.

Working Structure of Apex

As shown in the below diagram *Reference: Salesforce Developer Documentation*, Apex runs entirely on demand Force.com Platform:



Flow of Actions:

There are two sequence of actions when the developer saves the code and when an end user perform some action which invokes the Apex code as shown below:

- **Developer Action:** When a developer writes and saves Apex code to the platform, the platform application server first compiles the code into a set of instructions that can be understood by the Apex runtime interpreter, and then saves those instructions as metadata.
- **End User Action:** When an end-user triggers the execution of Apex, by clicking a button or accessing a Visualforce page, the platform application server retrieves the compiled

instructions from the metadata and sends them through the runtime interpreter before returning the result. The end-user observes no differences in execution time as compared to the standard application platform request.

Since Apex is proprietary language of Salesforce.com, it does not support some features which a general programming language supports. For example, below are some features which Apex does not support:

- It cannot show the elements in User interface.
- You cannot change the standard SFDC provided functionality and also it is not possible to prevent the standard functionality execution.
- Temporary file creation is not supported.
- Creating multiple threads is also not possible as we could do it in other languages.

Understanding the Apex syntax

Apex code typically contains many things that we might be familiar with from other programming languages.

Variable Declaration: As strongly typed language, you must declare every variable with data type in Apex. As seen in the code below *screenshotbelow*, `lstAcc` is declared with data type as List of Accounts.

SOQL Query: This will be used to fetch the data from Salesforce database. The query shown in screenshot below is fetching data from Account object.

Loop Statement: This loop statement is used for iterating over a list or iterating over a piece of code for specified number of times. In this code shown in screenshot below, iteration will be same as the number of records we have in `lstAcc`.

Flow Control Statement: The If statement is used for flow control in this code. Based on certain condition, it is decided whether to go for execution or to stop the execution of the particular piece of code. For example, in the code shown below, it is checking whether the list is empty or it contains records.

DML Statement: Performs the records insert, update, upsert, delete operation on the records in database. For example, the below code is updating Accounts with new field value.

Following is a sample of how an Apex code snippet would look like. We are going to study all these Apex programming concepts further in this tutorial.

```
1 public class MyFirstApexClass () {
2     List<Account> lstAcc = new List<Account>();
3     lstAcc = {SELECT id, Name, Description FROM Account WHERE Name LIKE 'Test%'};
4     List<Account> lstAccountUpdated List<Account>();
5     for (Account objAcc: lstAcc) {
6         objAcc.Description = 'This Account for Testing';
7         lstAccountUpdated.add(objAcc); //updated accounts
8     }
9     if(lstAccountUpdated != null && lstAccountUpdated.size() > 0) {
10         update lstAccountUpdated; //Perform DML
11     }
12 }
```

The image shows an Apex code snippet with several annotations in red boxes and arrows pointing to specific lines of code:

- Variable Declaration:** Points to line 2: `List<Account> lstAcc = new List<Account>();`
- SOQL query:** Points to line 3: `lstAcc = {SELECT id, Name, Description FROM Account WHERE Name LIKE 'Test%'};`
- Loop Statement:** Points to line 5: `for (Account objAcc: lstAcc) {`
- Flow control statement:** Points to line 9: `if(lstAccountUpdated != null && lstAccountUpdated.size() > 0) {`
- DML Statement:** Points to line 10: `update lstAccountUpdated; //Perform DML`