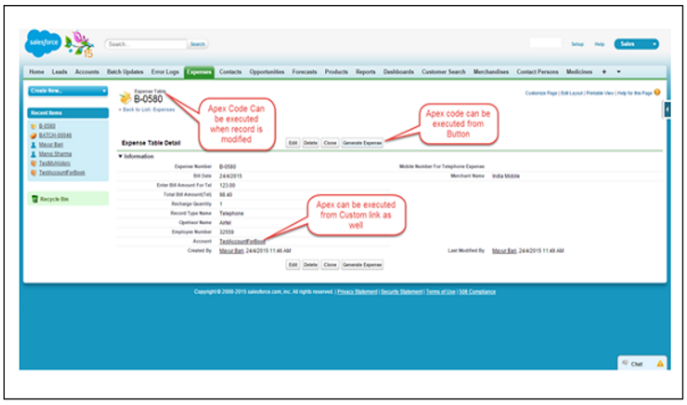
UNIT_V
Get Started with Apex

Apex is a programming language that uses Java-like syntax and acts like database stored procedures. Apex enables developers to add business logic to system events, such as button clicks, updates of related records, and Visualforce pages.

What is Apex?

Apex is a proprietary language developed by the 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.


Apex Code Can be executed when record is modified
Apex code can be executed from Button
Apex can be executed from Custom link as well

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 a 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 can 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.

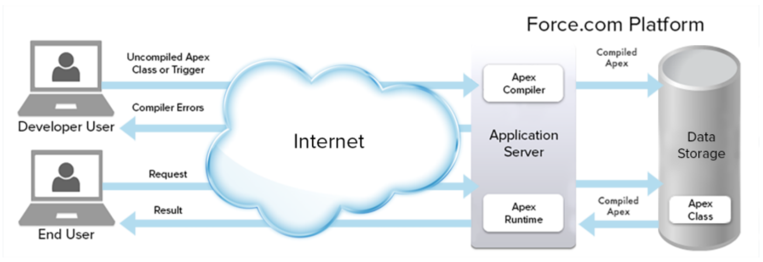
Apex Applications We can use Apex when we want to −

- 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.

As a language, Apex is:

- Hosted—Apex is saved, compiled, and executed on the server—the Lightning Platform.
- Object oriented—Apex supports classes, interfaces, and inheritance.
- Strongly typed—Apex validates references to objects at compile time.
- Multitenant aware—Because Apex runs in a multitenant platform, it guards closely against runaway code by enforcing limits, which prevent code from monopolizing shared resources.
- Integrated with the database—It is straightforward to access and manipulate records. Apex provides direct access to records and their fields, and provides statements and query languages to manipulate those records.
- Data focused—Apex provides transactional access to the database, allowing you to roll back operations.
- Easy to use—Apex is based on familiar Java idioms.
- Easy to test—Apex provides built-in support for unit test creation, execution, and code coverage. Salesforce ensures that all custom Apex code works as expected by executing all unit tests prior to any platform upgrades.
- Versioned—Custom Apex code can be saved against different versions of the API.



Apex Language Highlights

Like other object-oriented programming languages, these are some of the language constructs that Apex supports:

- Classes, interfaces, properties, and collections (including arrays).
- Object and array notation.
- Expressions, variables, and constants.
- Conditional statements (if-then-else) and control flow statements (for loops and while loops).

Unlike other object-oriented programming languages, Apex supports:

- Cloud development as Apex is stored, compiled, and executed in the cloud.
- Triggers, which are similar to triggers in database systems.

- Database statements that allow you to make direct database calls and query languages to query and search data.
- Transactions and rollbacks.
- The global access modifier, which is more permissive than the public modifier and allows access across namespaces and applications.
- Versioning of custom code.

The differences are mentioned in the table below:SOQL vs SOSL

SOQL (Salesforce Object Query Language)	SOSL (Salesforce Object Search Language)
Only one object can be searched at a time	Many objects can be searched at a time
Can query any type of field	Can query only on email, text or phone
Can be used in classes and triggers	Can be used in classes, but not triggers
DML Operation can be performed on query results	DML Operation cannot be performed on search results
Returns records	Returns fields

An Apex transaction represents a set of operations that are executed as a single unit. The operations here include the DML operations which are responsible for querying records. All the DML operations in a transaction either complete successfully, or if an error occurs even in saving a single record, then the entire transaction is rolled back.

Difference between public and global class in Apex?

Global class is accessible across the Salesforce instance irrespective of namespaces. Whereas, **public classes** are accessible only in the corresponding namespaces.

Use sObjects

Because Apex is tightly integrated with the database, you can access Salesforce records and their fields directly from Apex. Every record in Salesforce is natively represented as an ***sObject*** in Apex. For example, the Acme account record corresponds to an Account sObject in Apex. The fields of the Acme record that you can view and modify in the user interface can be read and modified directly on the sObject as well.

The following table lists some populated fields of the Acme account example record. The Account sObject is an abstraction of the account record and holds the account field information in memory as an object.

Each Salesforce record is represented as an sObject before it is inserted into Salesforce. Likewise, when persisted records are retrieved from Salesforce, they're stored in an sObject variable.Standard and custom object records in Salesforce map to their sObject types in Apex. Here are some common sObject type names in Apex used for standard objects.

- Account
- Contact
- Lead
- Opportunity

If you’ve added custom objects in your organization, use the API names of the custom objects in Apex. For example, a custom object called Merchandise corresponds to the Merchandise__c sObject in Apex.

sObject and Field Names

The names of sObjects correspond to the API names of the corresponding standard or custom objects. Similarly, the names of sObject fields correspond to the API names of the corresponding fields.

API names of object and fields can differ from their labels. For example, the Employees field has a label of Employees and appears on the account record page as Employees but its API name is NumberOfEmployees. To access this field in Apex, you’ll need to use the API name for the field: NumberOfEmployees.

The following are highlights of some rules used for API names for custom objects and custom fields.

For custom objects and custom fields, the API name always ends with the __c suffix. For custom relationship fields, the API name ends with the __r suffix. For example:

- A custom object with a label of Merchandise has an API name of Merchandise__c.
- A custom field with a label of Description has an API name of Description__c.
- A custom relationship field with a label of Items has an API name of Items__r.

In addition, spaces in labels are replaced with underscores in API names. For example, a custom field name of Employee Seniority has an API name of Employee_Seniority__c.

Write SOQL Queries

To read a record from Salesforce, you must write a query. Salesforce provides the Salesforce Object Query Language, or SOQL in short, that you can use to read saved records. SOQL is similar to the standard SQL language but is customized for the Lightning Platform.

Because Apex has direct access to Salesforce records that are stored in the database, you can embed SOQL queries in your Apex code and get results in a straightforward fashion. When SOQL is embedded in Apex, it is referred to as ***inline SOQL***.

To include SOQL queries within your Apex code, wrap the SOQL statement within square brackets and assign the return value to an array of sObjects. For example, the following retrieves all account records with two fields, Name and Phone, and returns an array of Account sObjects.

```
Account[] accts = [SELECT Name,Phone FROM Account];
```

Basic SOQL Syntax

This is the syntax of a basic SOQL query:

```
SELECT fields FROM ObjectName [WHERE Condition]
```

Copy

The WHERE clause is optional. Let's start with a very simple query. For example, the following query retrieves accounts and gets Name and Phone fields for each account.

```
SELECT Name,Phone FROM Account
```

Copy

The query has two parts:

1. SELECT Name,Phone: This part lists the fields that you would like to retrieve. The fields are specified after the SELECT keyword in a comma-delimited list. Or you can specify only one field, in which case no comma is necessary (e.g. SELECT Phone).
2. FROM Account: This part specifies the standard or custom object that you want to retrieve. In this example, it's Account. For a custom object called Invoice_Statement, it is Invoice_Statement__c.

Write SOSL Queries

Salesforce Object Search Language (SOSL) is a Salesforce search language that is used to perform text searches in records. Use SOSL to search fields across multiple standard and custom object records in Salesforce. SOSL is similar to Apache Lucene.

Adding SOSL queries to Apex is simple—you can embed SOSL queries directly in your Apex code. When SOSL is embedded in Apex, it is referred to as ***inline SOSL***.

This is an example of a SOSL query that searches for accounts and contacts that have any fields with the word 'SFDC'.

```
List<List<SObject>> searchList = [FIND 'SFDC' IN ALL FIELDS

                    RETURNING Account(Name), Contact(FirstName,LastName)];
```

Differences and Similarities Between SOQL and SOSL

Like SOQL, SOSL allows you to search your organization's records for specific information. Unlike SOQL, which can only query one standard or custom object at a time, a single SOSL query can search all objects.

Another difference is that SOSL matches fields based on a word match while SOQL performs an exact match by default (when not using wildcards). For example, searching for 'Digital' in

SOSL returns records whose field values are 'Digital' or 'The Digital Company', but SOQL returns only records with field values of 'Digital'.

SOQL and SOSL are two separate languages with different syntax. Each language has a distinct use case:

- Use SOQL to retrieve records for a single object.
- Use SOSL to search fields across multiple objects. SOSL queries can search most text fields on an object.

Basic SOSL Syntax

SOSL allows you to specify the following search criteria:

- Text expression (single word or a phrase) to search for
- Scope of fields to search
- List of objects and fields to retrieve
- Conditions for selecting rows in the source objects

This is the syntax of a basic SOSL query:

```
FIND 'SearchQuery' [IN SearchGroup] [RETURNING ObjectsAndFields]
```

SearchQuery is the text to search for (a single word or a phrase). Search terms can be grouped with logical operators (AND, OR) and parentheses. Also, search terms can include wildcard characters (*, ?). The * wildcard matches zero or more characters at the middle or end of the search term. The ? wildcard matches only one character at the middle or end of the search term.