What is a Class?

A class is a template or blueprint from which objects are created. An object is an instance of a class. This is standard definition of Class. Apex Classes are similar to Java Classes.

Example:

For example, InvoiceProcessor class describes the class which has all the methods and actions which we could do with the Invoice. If you create an instance of this class then it will represent the single invoice which is currently in context.

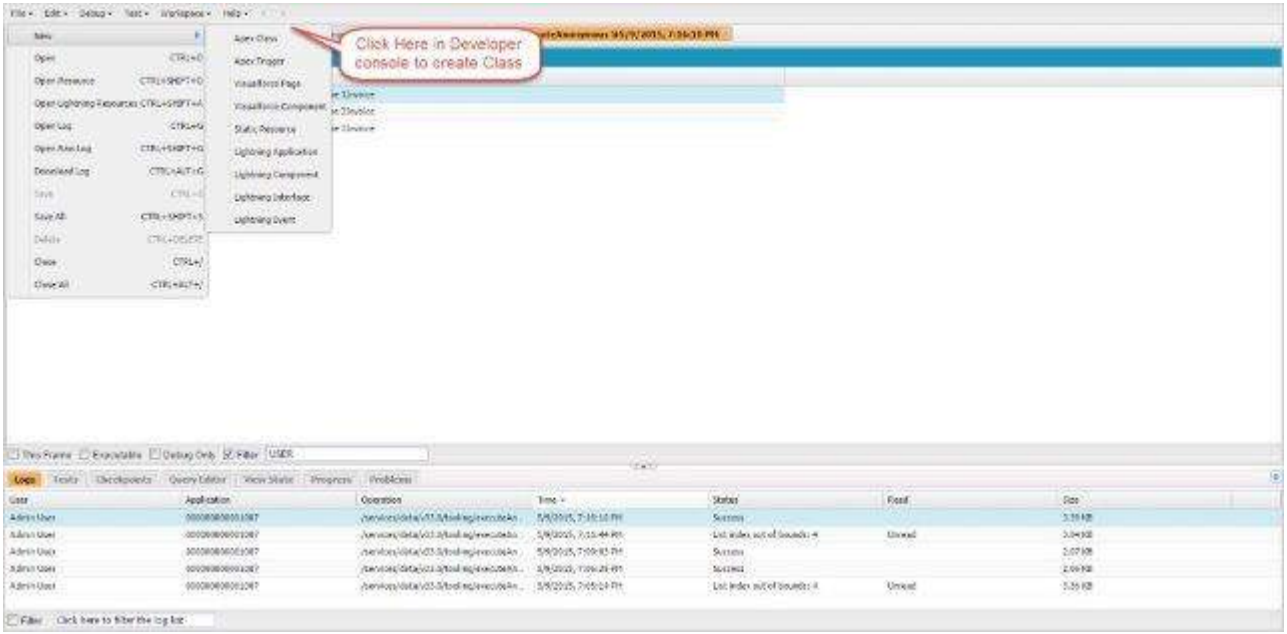
Creating Classes

You could create class in Apex from Developer Console, Force.com Eclipse IDE and from Apex Class detail page as well.

From Developer Console:

Step 1: Go to Name and click on Developer Console.

Step 2: Click on File => New and then click on Apex class



From Force.com IDE:

Step 1: Open Force.com Eclipse IDE

Step 2: Create a New Project by clicking on File=>New=>Apex Class.

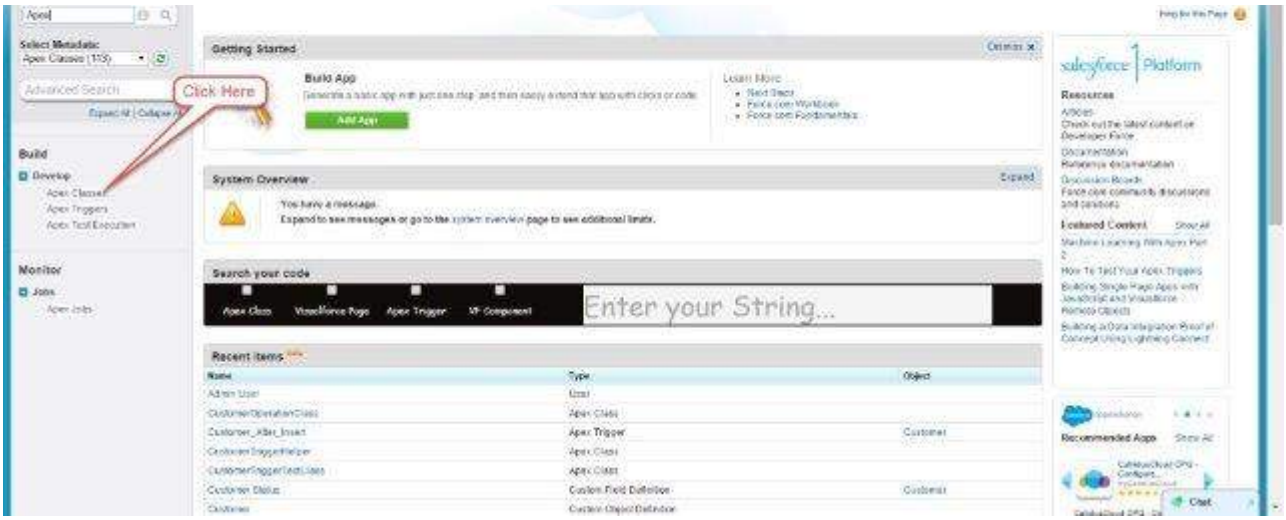
Step 3: Provide the Name for the Class and click on OK.

Once this is done, the new class will be created.

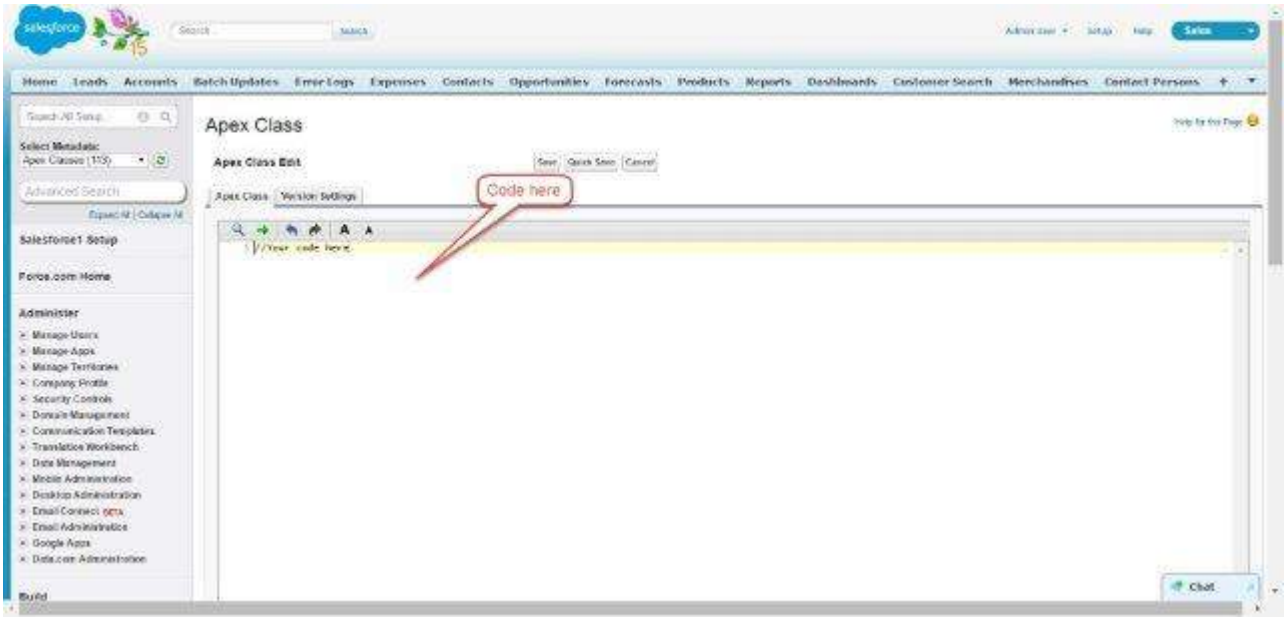
From Apex Class Detail Page:

Step 1: Click on Name=>Setup

Step 2: Search for 'Apex Class' and click on the link. It will open the Apex Class details page.



Step 3: Click on 'New' and then provide the Name for class and then click Save.



Apex Class Structure

Below is the sample structure for Apex class definition.

Syntax:

```
private | public | global
[virtual | abstract | with sharing | without sharing]
class ClassName [implements InterfaceNameList] [extends ClassName]
{
// Classs Body
}
```

This definition uses a combination of access modifiers, sharing modes, class name and class body. We will look at all these options further.

Example:

Below is the sample structure for Apex class definition:

```
public class MySampleApexClass {//Class definition and body
 public static Integer myValue = 0;  //Class Member variable
 public static String myString = ''; //Class Member variable

 public static Integer getCalculatedValue () {
        //Method definition and body
  //do some calculation
```

```
  myValue = myValue+10;
  return myValue;
 }
}
```

Access Modifiers

Private: If you declare the access modifier as 'Private', then this class will be known only locally and you cannot access this class outside of that particular piece. By default, classes have this modifier.

Public: If you declare the class as 'Public' then this implies that this class is accessible to your organization and your defined namespace. Normally, most of the Apex classes are defined with this keyword.

Global: If you declare the class as 'global' then this will accessible by all apex code irrespective of your organization. If you have method defined with webservice keyword then you must declare the containing class with global keyword.

Sharing Modes

With Sharing:

This is a special feature of Apex Classes in Salesforce. When a class is specified with 'With Sharing' keyword then it has following implications: When the class will get executed, it will respect the User's access settings and profile permission. Suppose, User's action has triggered the record update for 30 records, but user has access to only 20 records and 10 records are not accessible. Then, if the class is performing the action to update the records, only 20 records will be updated to which the user has access and rest of 10 records will not be updated. This is also called as User mode.

Without Sharing:

Even if the User does not have access to 10 records out of 30, all the 30 records will be updated as the Class is running in System mode, i.e. it has been defined with Without Sharing keyword. This is called System Mode.

Virtual:

If you use 'virtual' keyword then it indicates that this class can be extended and overrides are allowed. If you would like to override the methods, then the classes should be declared with virtual keyword.

Abstract:

If you declare the class as 'abstract', then it will only contain the signature of method and not the actual implementation.

Class Variables

Syntax:

```
[public | private | protected | global] [final] [static] data_type variable_name [=
value]
```

In the above syntax:

- Variable data type and variable name are mandatory
- Access modifiers and value are optional.

Example:

```
public static final Integer myvalue;
```