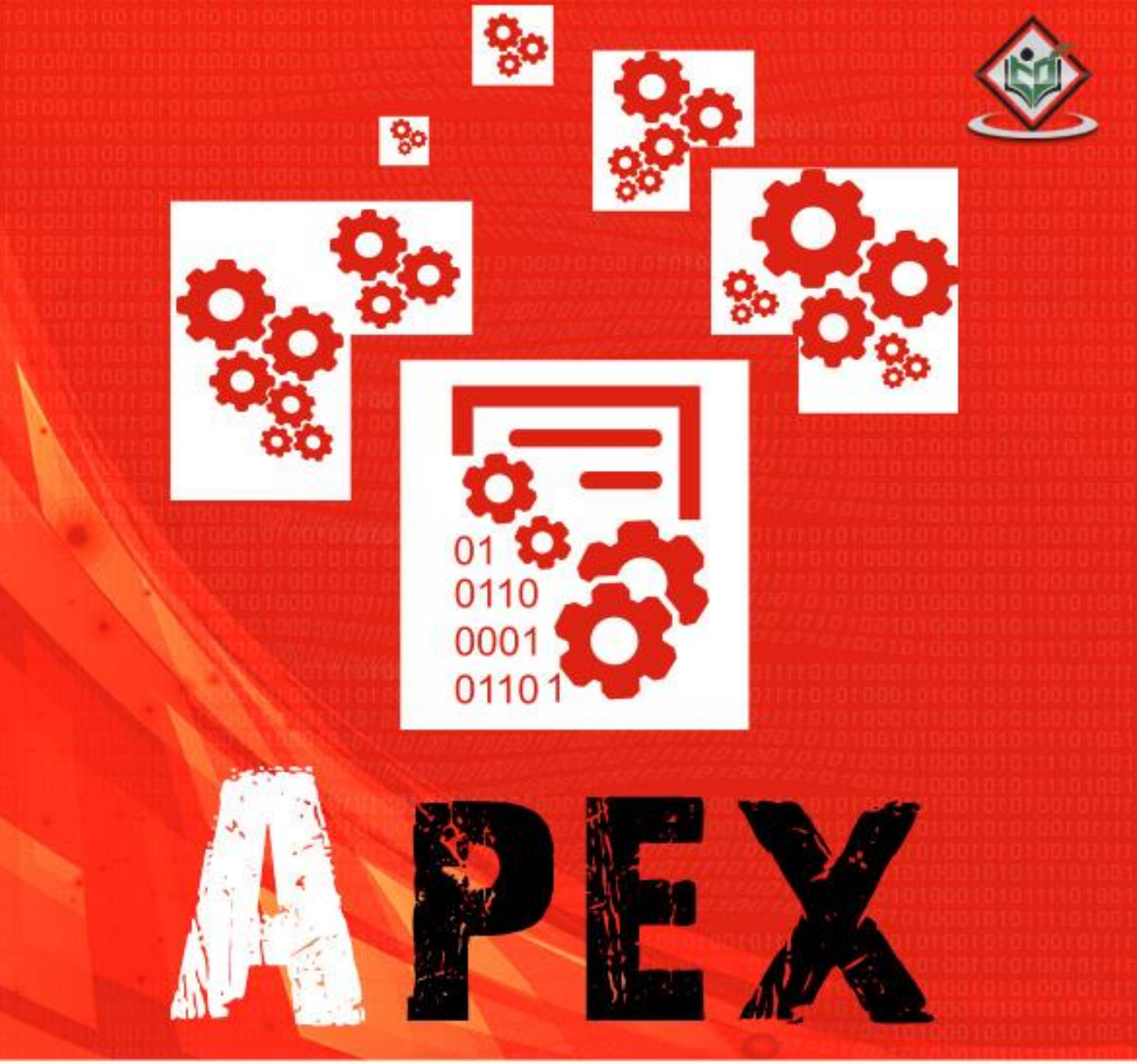


APEX

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

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

Audience

This tutorial is targeted for Salesforce programmers beginning to learn Apex. This will bring you to an Intermediate level of expertise in Apex programming covering all the important aspects of Apex with complete hands-on code experience.

Prerequisites

Basic knowledge of Salesforce platform and development is needed. Apex is a programming language which has to be used with Salesforce. This tutorial assumes that you already have set up the Salesforce instance which will be used to do our Apex programming.

Copyright & Disclaimer

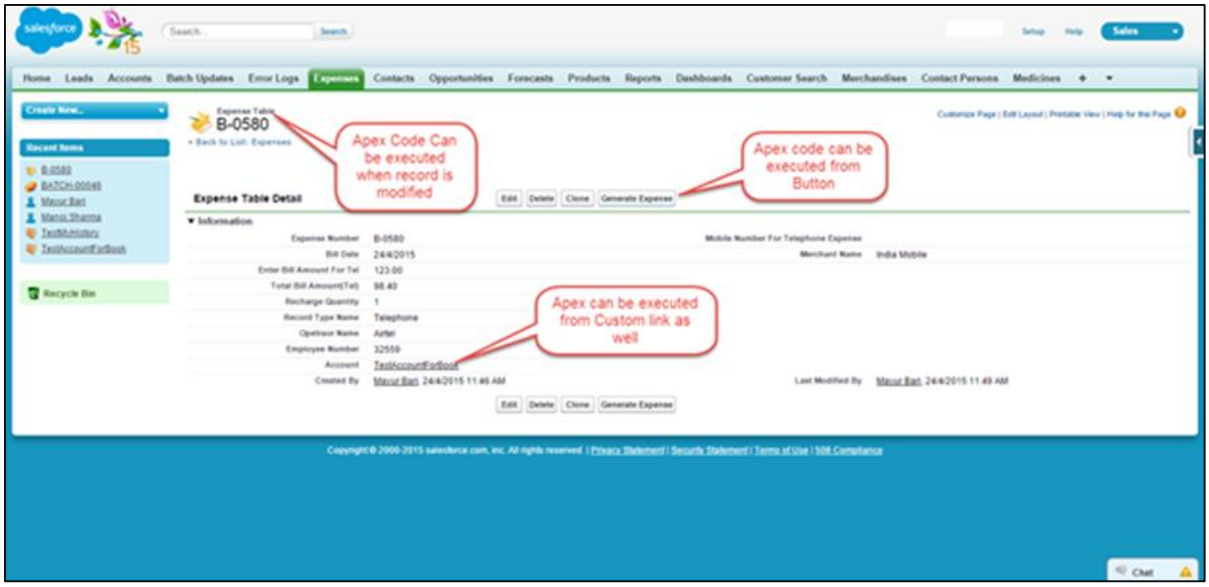
Table of Contents

1. APEX – OVERVIEW

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.



Features of Apex as a Language

Let us now discuss the 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.

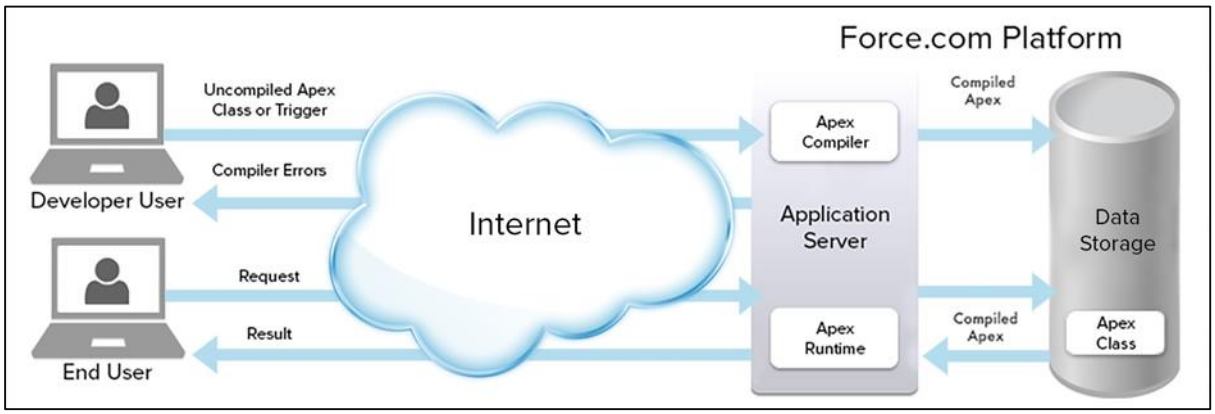
Apex Applications

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.

Working Structure of Apex

As shown in the diagram below (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 performs 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 the proprietary language of Salesforce.com, it does not support some features which a general programming language does. Following are a few 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 can 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 (screenshot below), 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 a specified number of times. In the code shown in the screenshot below, iteration will be same as the number of records we have.

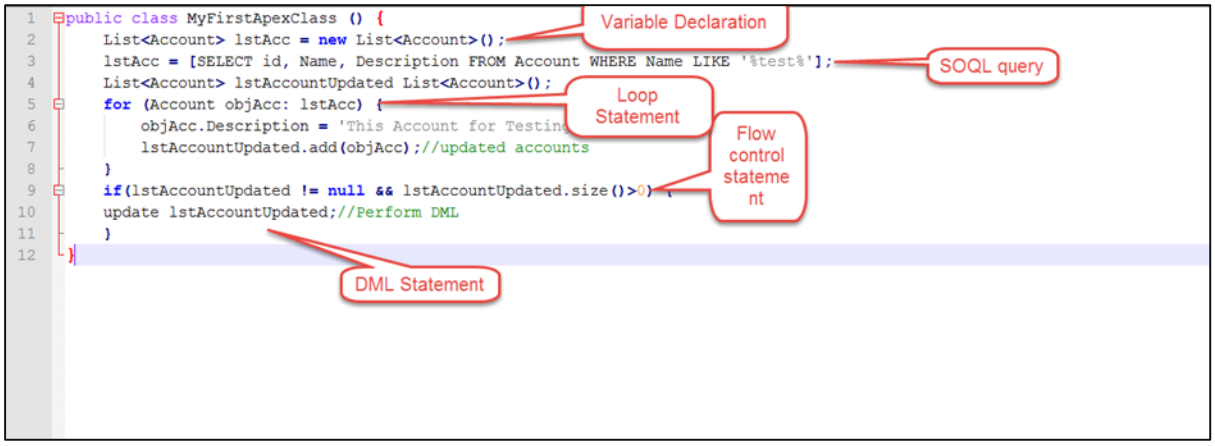
Flow Control Statement

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 code given below helps in updating Accounts with new field value.

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

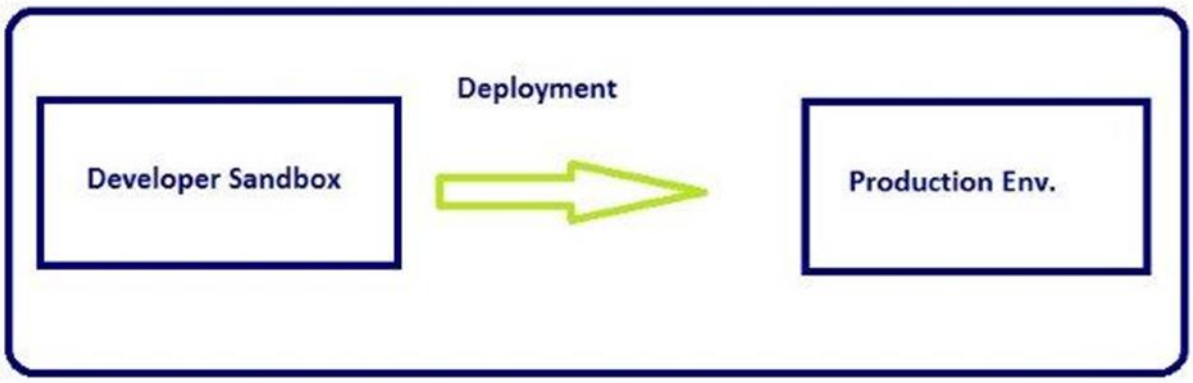


2. APEX – ENVIRONMENT

In this chapter, we will understand the environment for our Salesforce Apex development. It is assumed that you already have a Salesforce edition set up for doing Apex development.

You can develop the Apex code in either Sandbox or Developer edition of Salesforce. A Sandbox organization is a copy of your organization in which you can write code and test it without taking the risk of data modification or disturbing the normal functionality. As per the standard industrial practice, you have to develop the code in Sandbox and then deploy it to the Production environment.

For this tutorial, we will be using the Developer edition of Salesforce. In the Developer edition, you will not have the option of creating a Sandbox organization. The Sandbox features are available in other editions of Salesforce.



Apex Code Development Tools

In all the editions, we can use any of the following three tools to develop the code:

- Force.com Developer Console
- Force.com IDE
- Code Editor in the Salesforce User Interface

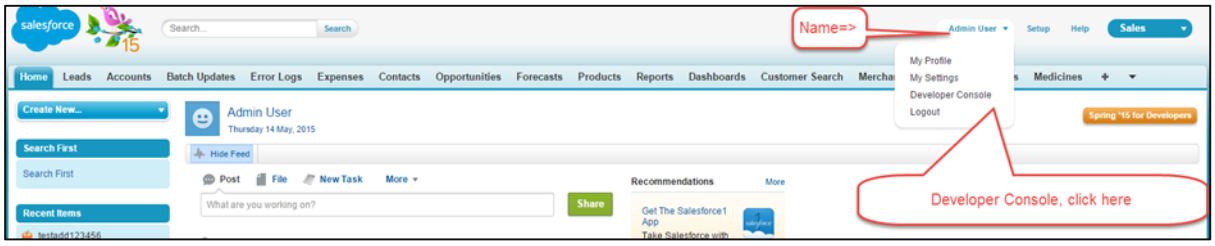
Note: We will be utilizing the Developer Console throughout our tutorial for code execution as it is simple and user friendly for learning.

Force.com Developer Console

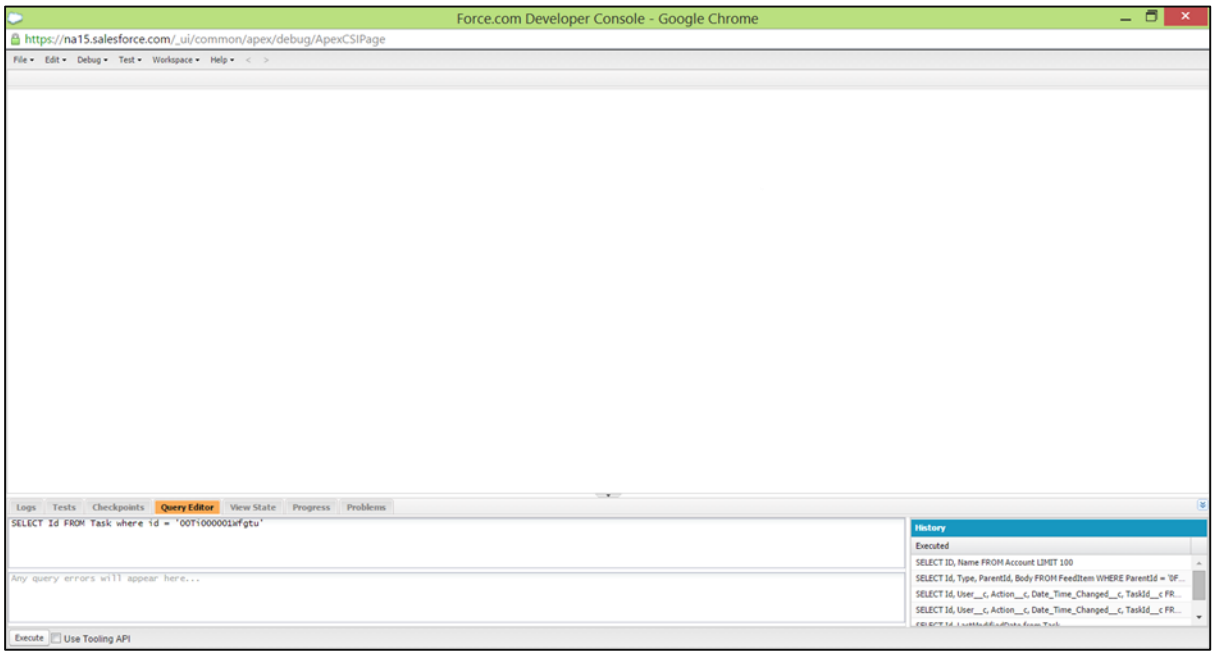
The Developer Console is an integrated development environment with a collection of tools you can use to create, debug, and test applications in your Salesforce organization.

Follow these steps to open the Developer Console:

Step 1: Go to Name->Developer Console



Step2: Click on "Developer Console" and a window will appear as in the following screenshot.



Following are a few operations that can be performed using the Developer Console.

- **Writing and compiling code** - You can write the code using the source code editor. When you save a trigger or class, the code is automatically compiled. Any compilation errors will be reported.
- **Debugging** - You can view debug logs and set checkpoints that aid in debugging.
- **Testing** - You can execute tests of specific test classes or all classes in your organization, and you can view test results. Also, you can inspect code coverage.
- **Checking performance** - You can inspect debug logs to locate performance bottlenecks.

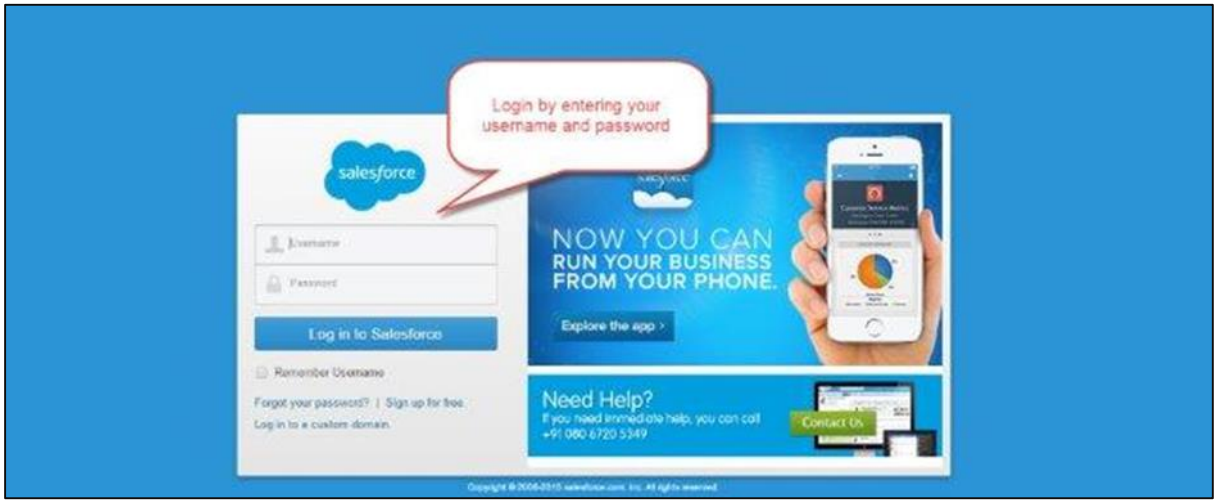
- **SOQL queries** - You can query data in your organization and view the results using the Query Editor.
- **Color coding and autocomplete** - The source code editor uses a color scheme for easier readability of code elements and provides auto completion for class and method names.

Executing Code in Developer Console

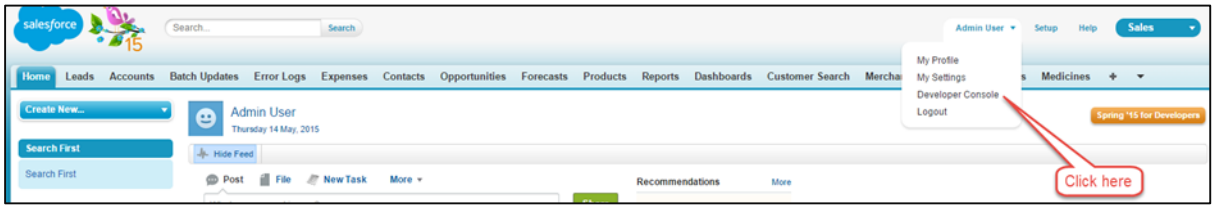
All the code snippets mentioned in this tutorial need to be executed in the developer console. Follow these steps to execute steps in Developer Console.

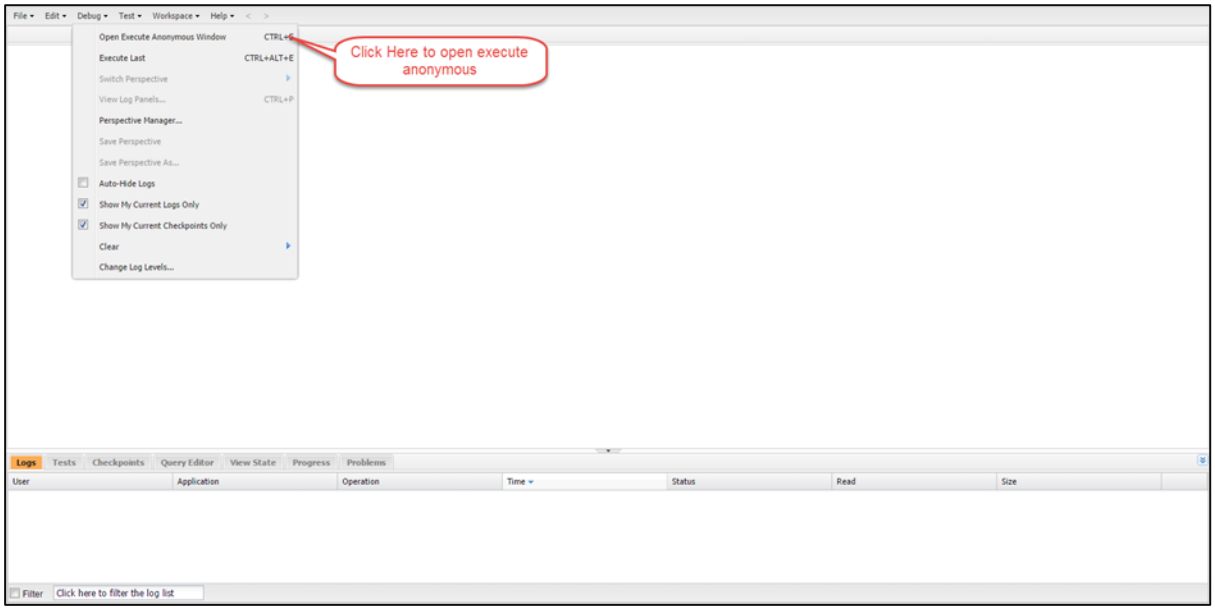
Step 1: Login to the Salesforce.com using **login.salesforce.com**. Copy the code snippets mentioned in the tutorial. For now, we will use the following sample code:

```
String myString = 'MyString';
System.debug('Value of String Variable'+myString);
```

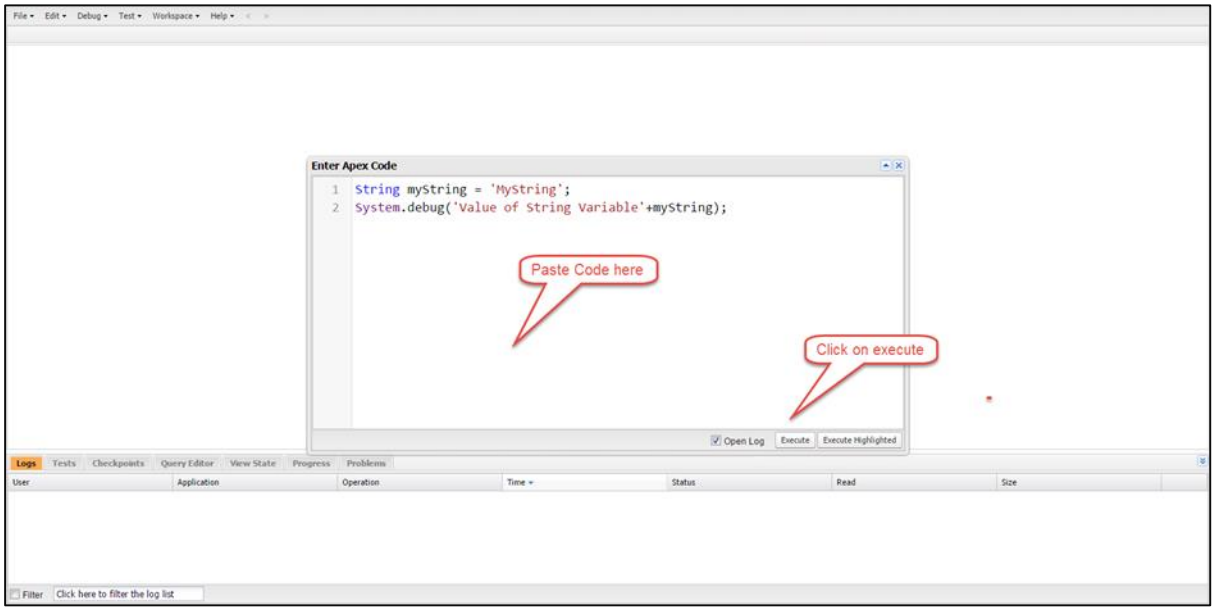


Step 2: To open the Developer Console, click on Name -> Developer Console and then click on Execute Anonymous as shown below.

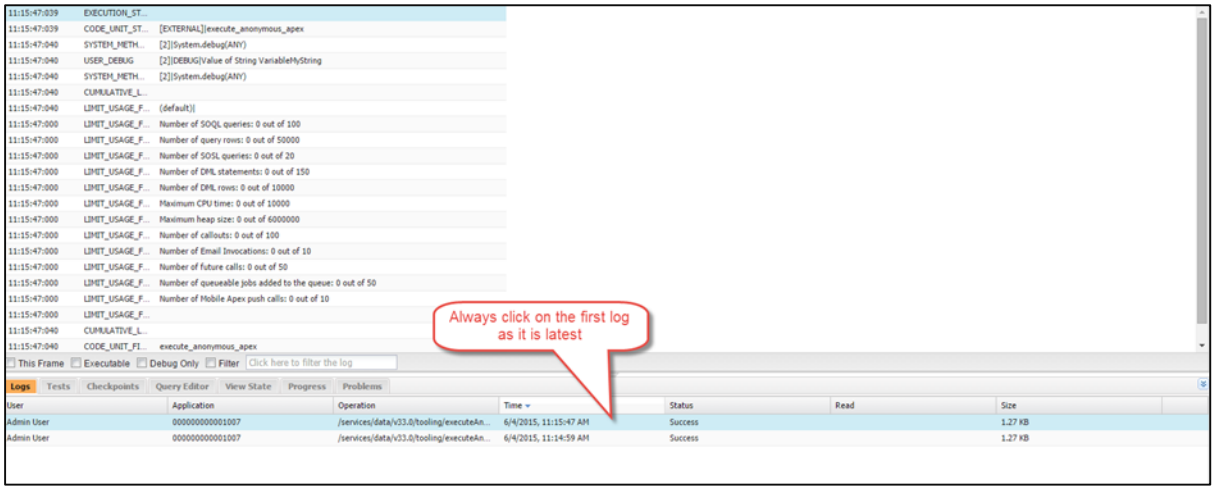




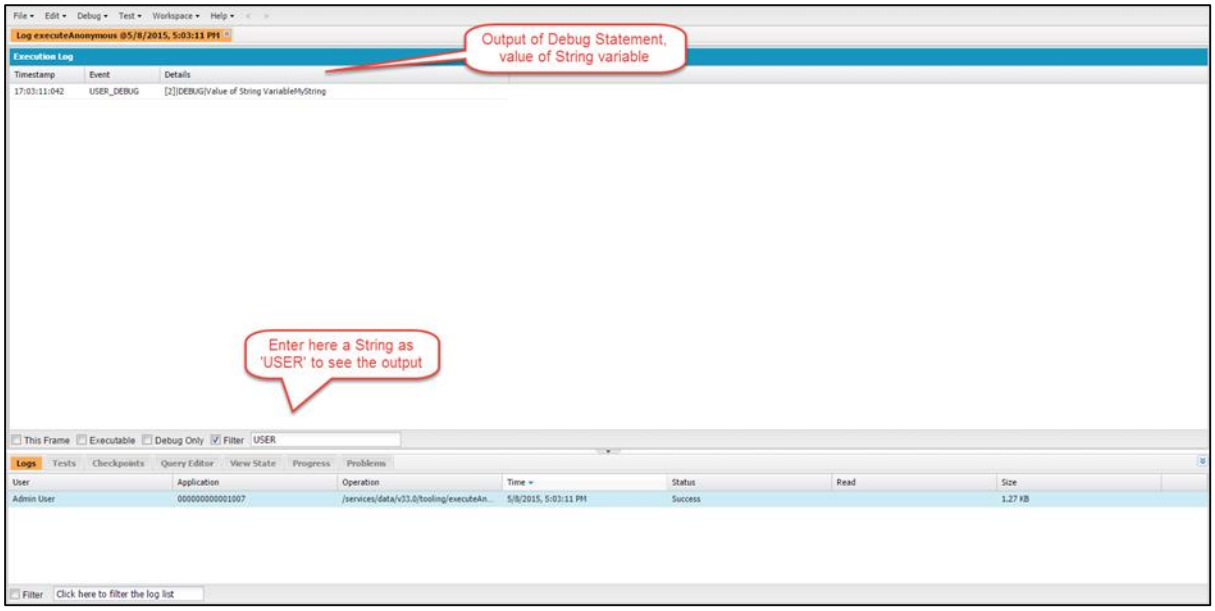
Step 3: In this step, a window will appear and you can paste the code there.



Step 4: When we click on **Execute**, the debug logs will open. Once the log appears in window as shown below, then click on the log record:



Then type 'USER' in the window as shown below and the output statement will appear in the debug window. This 'USER' statement is used for filtering the output.



So basically, you will be following all the above mentioned steps to execute any code snippet in this tutorial.

3. APEX – EXAMPLE

Enterprise Application Development Example

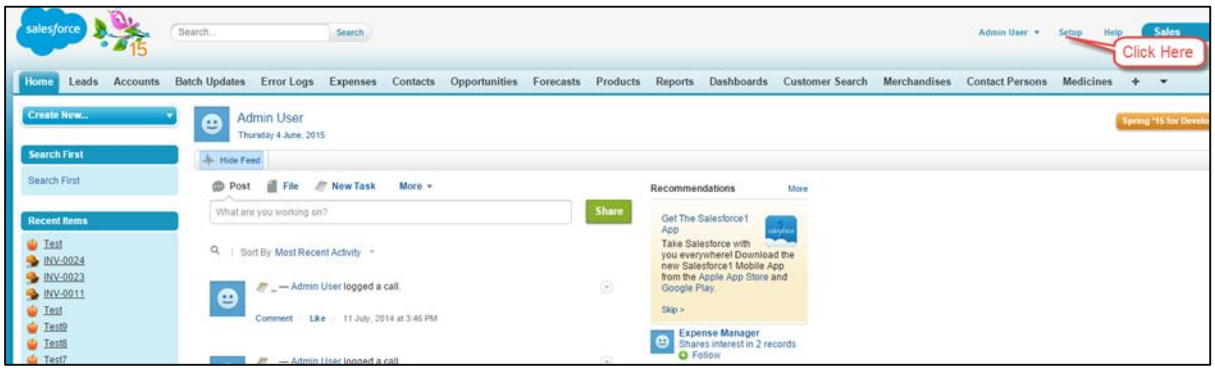
For our tutorial, we will be implementing the CRM application for a Chemical Equipment and Processing Company. This company deals with suppliers and provides services. We will work out small code snippets related to this example throughout our tutorial to understand every concept in detail.

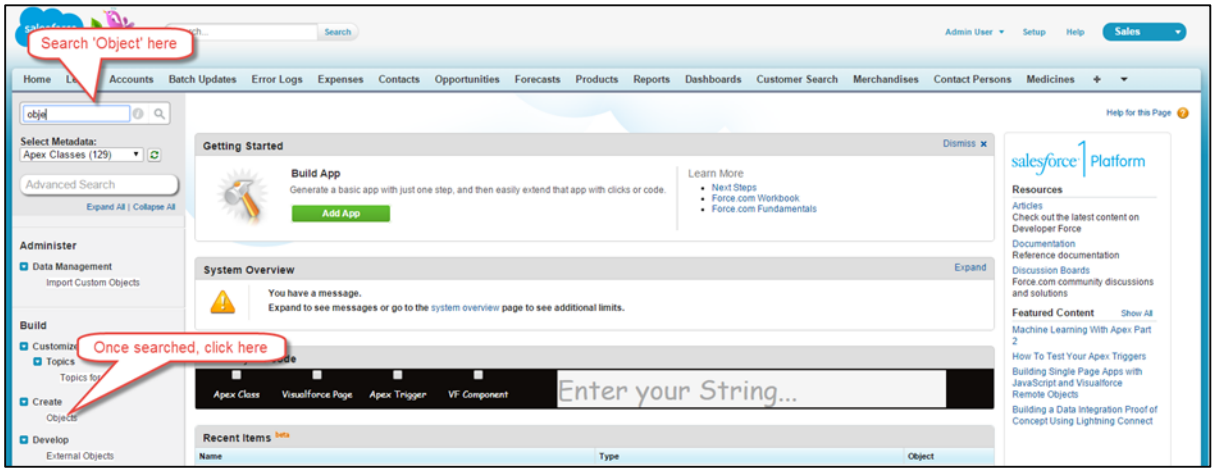
For executing the code in this tutorial, you will need to have two objects created: Customer and Invoice objects. If you already know how to create these objects in Salesforce, you can skip the steps given below. Else, you can follow the step by step guide below.

Creating Customer Object

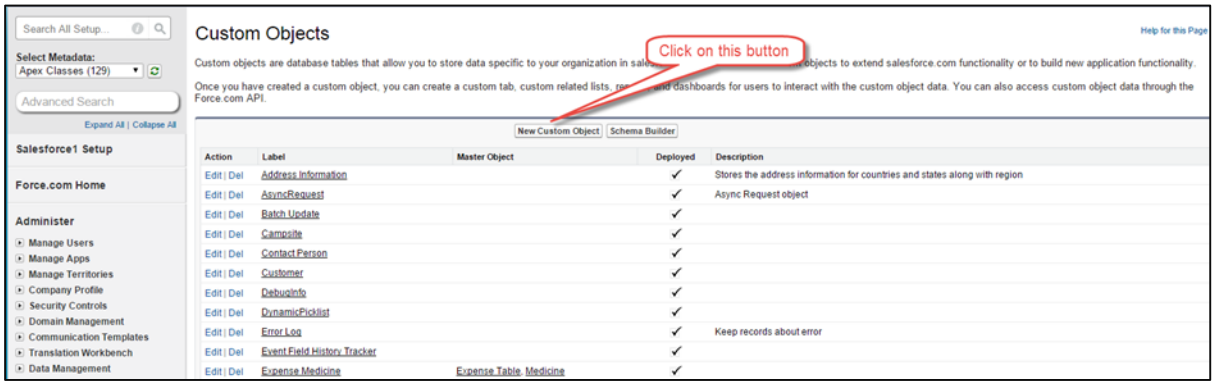
We will be setting up the Customer object first.

Step 1: Go to Setup and then search for 'Object' as shown below. Then click on the Objects link as shown below:

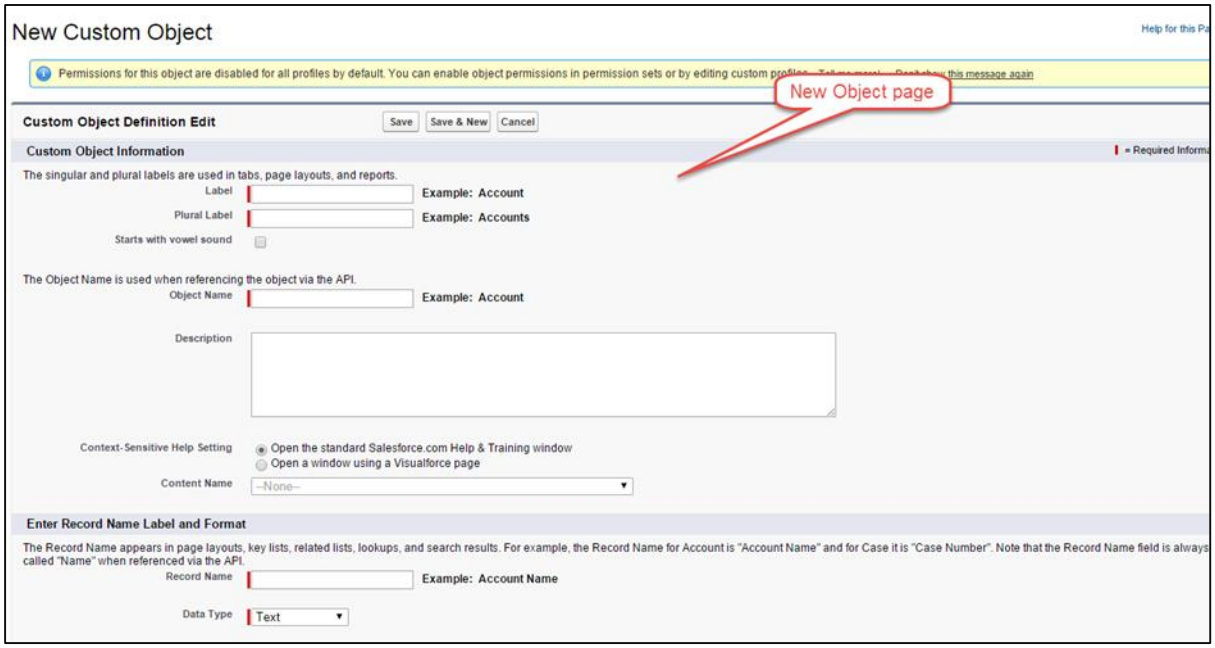




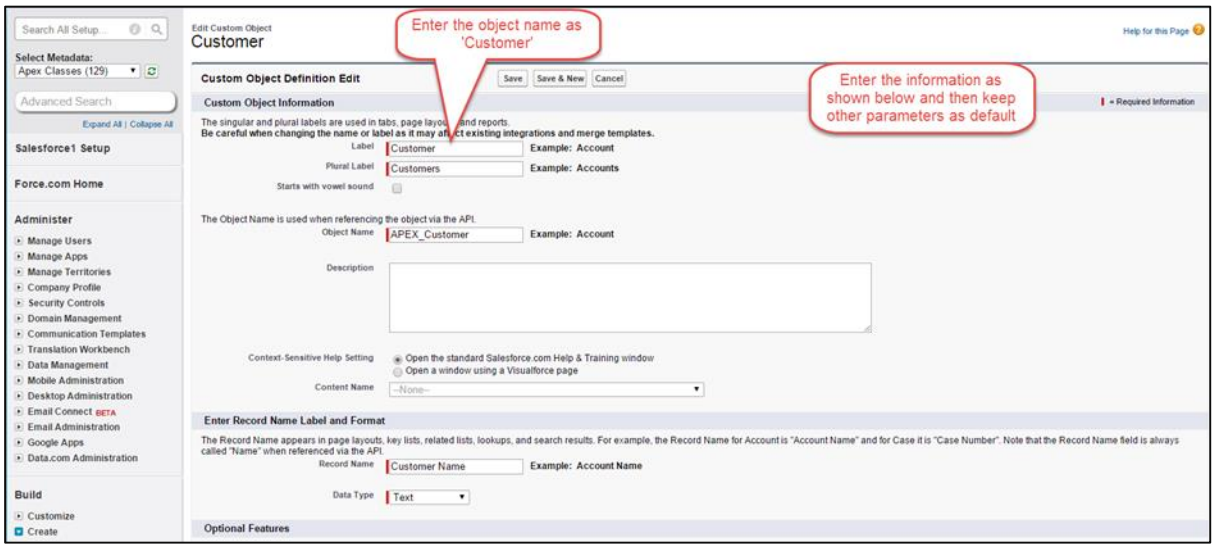
Step 2: Once the object page is opened, then click on the **'Create New Object'** button as shown below:



Step 3: After clicking on button, the new object creation page will appear and then enter all the object details as entered below. Object name should be Customer. You just have to enter the information in the field as shown in the screenshot below and keep other default things as it is.



Enter the information and then click on the 'Save' button:

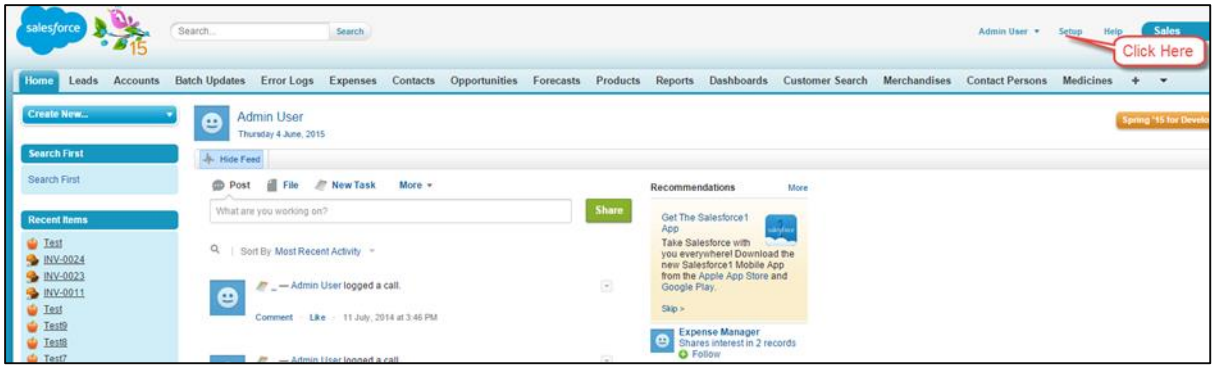


By following the above steps, we have successfully created the Customer object.

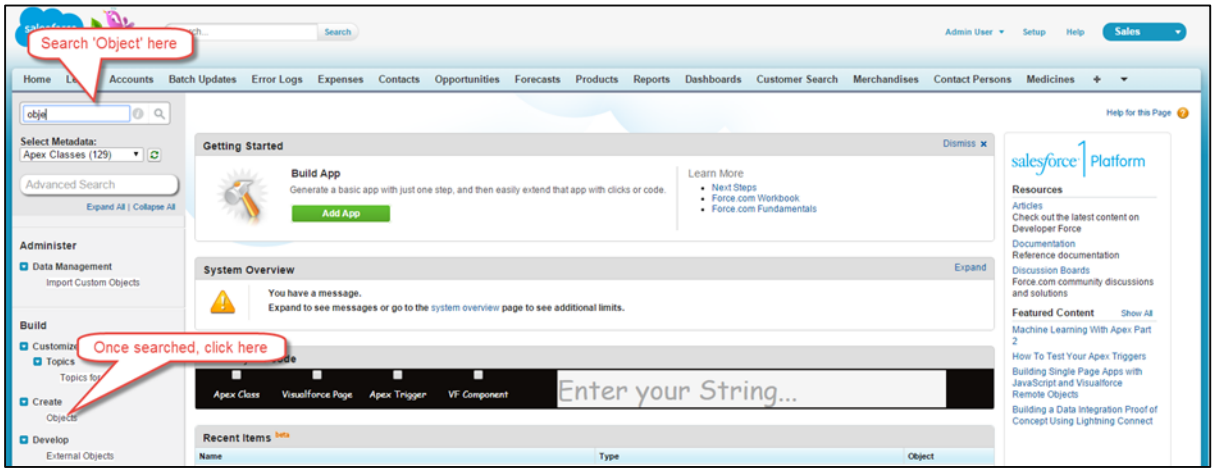
Creating the Custom Fields for Customer object

Now that we have our Customer object set up, we will create a field 'Active' and then you can create the other fields by following similar steps. The Name and API name of the field will be given in the screenshot.

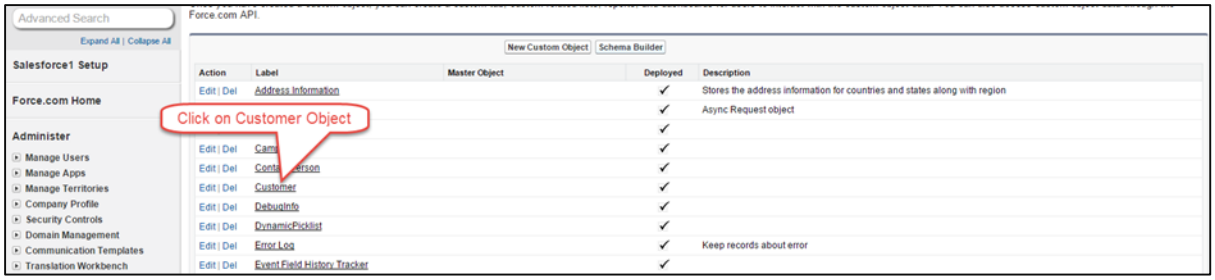
Step 1: We will be creating a field named as 'Active' of data type as Checkbox. Go to Setup and click on it.



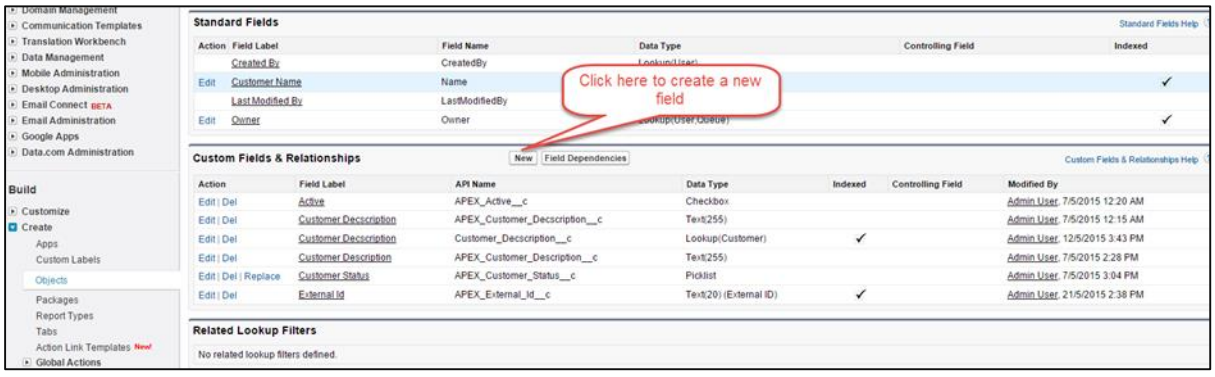
Step 2: Search for 'Object' as shown below and click on it:



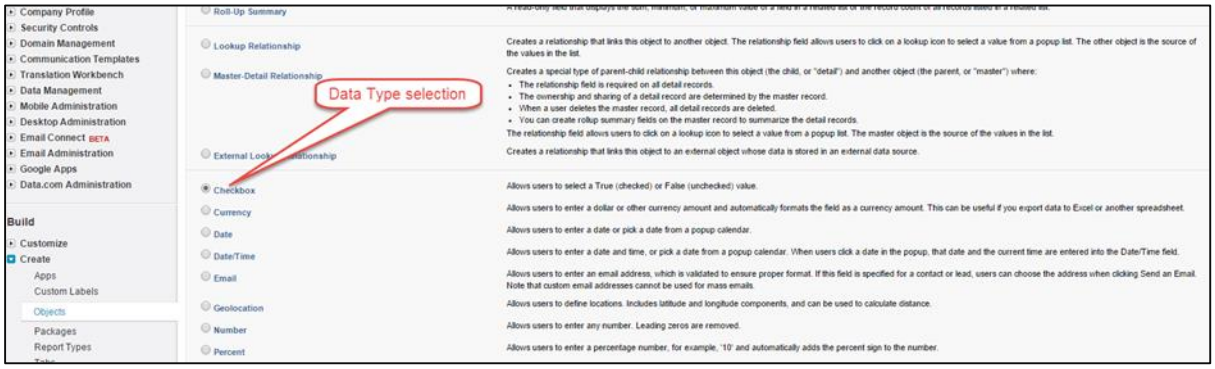
Step 3: Click on object 'Customer':



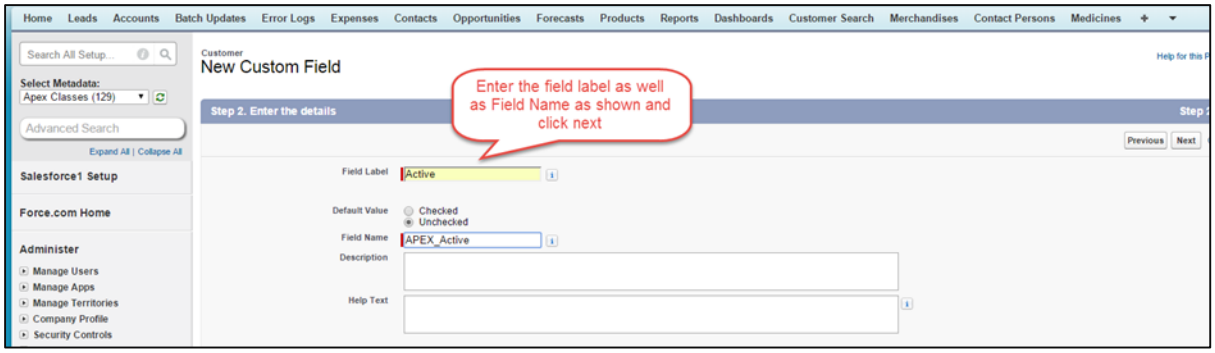
Step 4: Once you have clicked on the Customer object link and the object detail page appears, click on the New button:



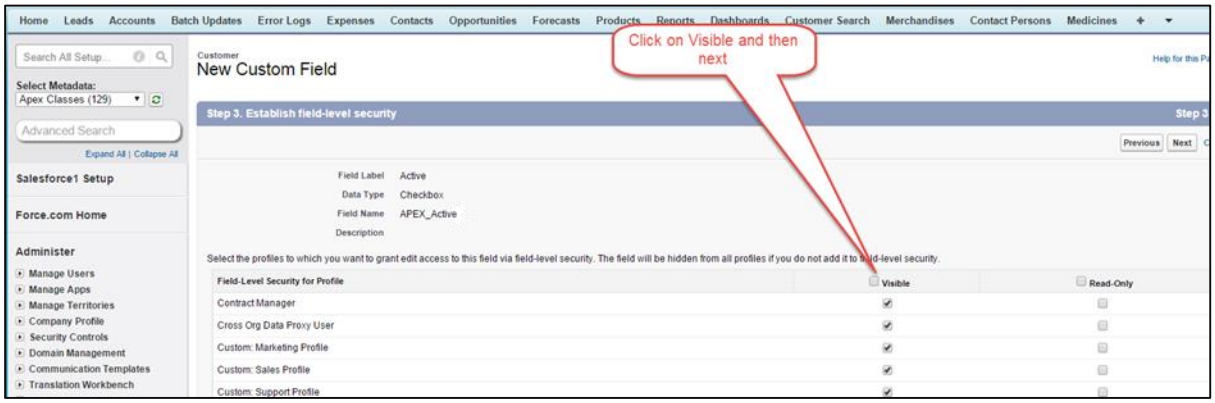
Step 5: Now, select the data type as Checkbox and click Next:



Step 6: Enter the field name and label as shown below:

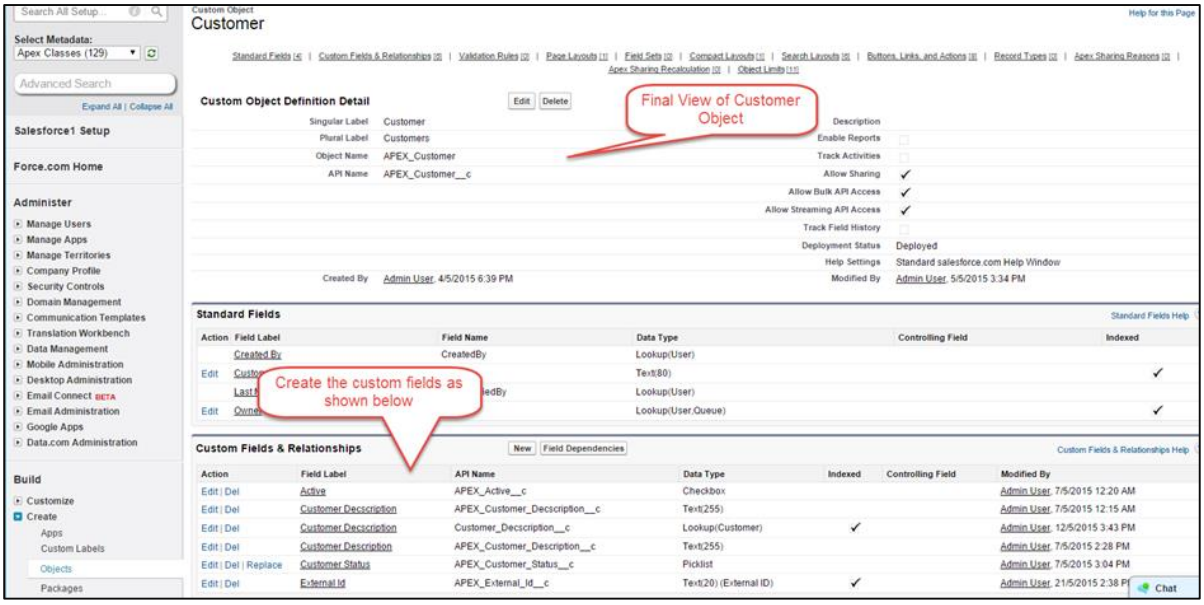


Step 7: Click on Visible and then click Next:



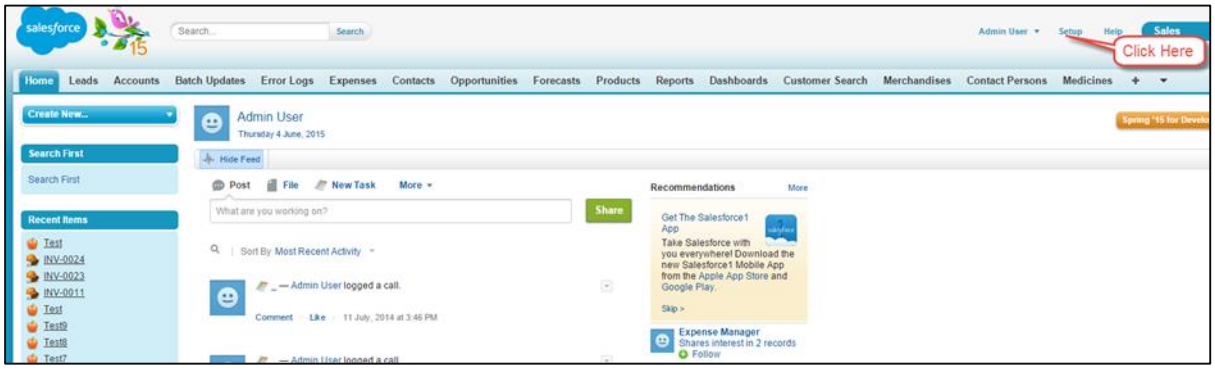
Step 8: Now click on 'Save'.

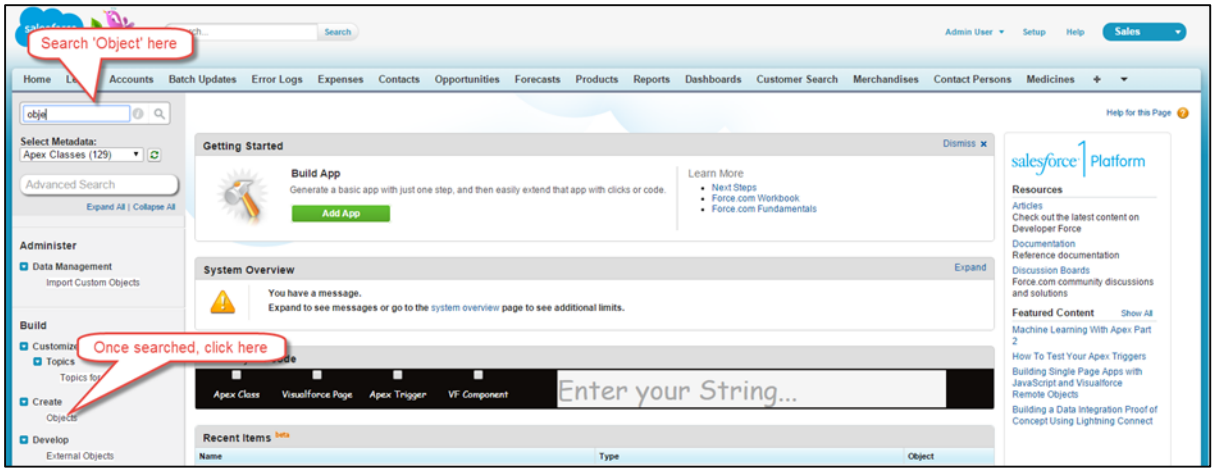
By following the above steps, our custom field 'Active' is created. You have to follow all the above custom field creation steps for the remaining fields. This is the final view of customer object once all the fields are created:



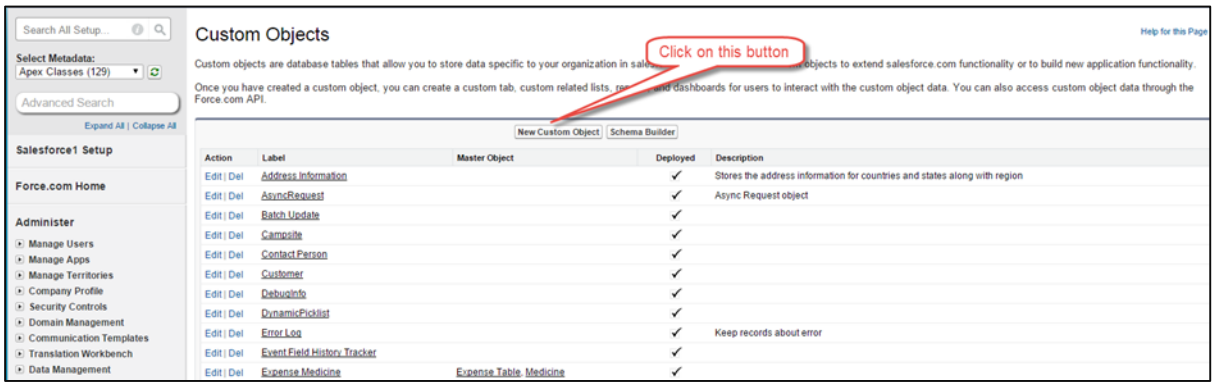
Creating Invoice Object

Step 1: Go to Setup and search for 'Object' and then click on the Objects link as shown below:





Step 2: Once the object page is opened, then click on the 'Create New Object' button as shown below:



Step 3: After clicking on the button, the new object creation page will appear as shown in the screenshot below. You need to enter the details here. The object name should be Invoice. This is similar to how we created the Customer object earlier in this tutorial.