Testing is the integrated part of Apex or any other application development. In Apex, we have separate test classes to develop for all the unit testing.

Test Classes

In SFDC, the code must have 75% code coverage in order to be deployed to Production. This code coverage is performed by the test classes. Test classes are the code snippets which test the functionality of other Apex class.

Let's write a test class for one of our code which we have written previously. We will write test class to cover our Trigger and Helper class code. Below is the trigger and helper class which needs to be covered.

```
//Trigger with Helper Class
trigger Customer_After_Insert on APEX_Customer__c (after update) {
    CustomerTriggerHelper.createInvoiceRecords(Trigger.new, trigger.oldMap);//Trigger
calls the helper class and does not have any code in Trigger
}

//Helper Class:
public class CustomerTriggerHelper {
    public static void createInvoiceRecords (List<apex_customer__c> customerList,
Map<id, apex_customer__c> oldMapCustomer) {
        List<apex_invoice__c> InvoiceList = new List<apex_invoice__c>();
        for (APEX_Customer__c objCustomer: customerList) {
            if (objCustomer.APEX_Customer_Status__c == 'Active' &&
oldMapCustomer.get(objCustomer.id).APEX_Customer_Status__c == 'Inactive') {//condition to
check the old value and new value
                APEX_Invoice__c objInvoice = new APEX_Invoice__c();
                objInvoice.APEX_Status__c = 'Pending';
 objInvoice.APEX_Customer__c = objCustomer.id;
                InvoiceList.add(objInvoice);
            }
        }
        insert InvoiceList;//DML to insert the Invoice List in SFDC
    }
}
```

Creating Test Class

Data Creation

We need to create data for test class in our test class itself. Test class by default does not have access to organization data but if you set @isTest*seeAllData = true* then it will have the access to organization's data as well.

@isTest annotation

By using this annotation, you declared that this is a test class and it will not be counted against the organization's total code limit.

testMethod keyword

Unit test methods are the methods which does no take arguments, commit no data to the database, send no emails, and are declared with the testMethod keyword or the isTest annotation in the method definition. Also, test methods must be defined in test classes, that is, classes annotated with isTest.

In our example that we will see, 'myUnitTest' is our test method.

Test.startTest and **Test.stopTest**

These are the standard test methods which are available for test classes. These methods contain the event or action for which we would be simulating our test. Like in this example, we would be testing our trigger and helper class to simulate the fire trigger by updating the records as we have done to start and stop block. This also provides separate governor limit to the code which is in start and stop block.

System.assert

This method checks the desired output with the actual. In this case, we are expecting an Invoice record to be inserted so we added assert to check the same.

Example

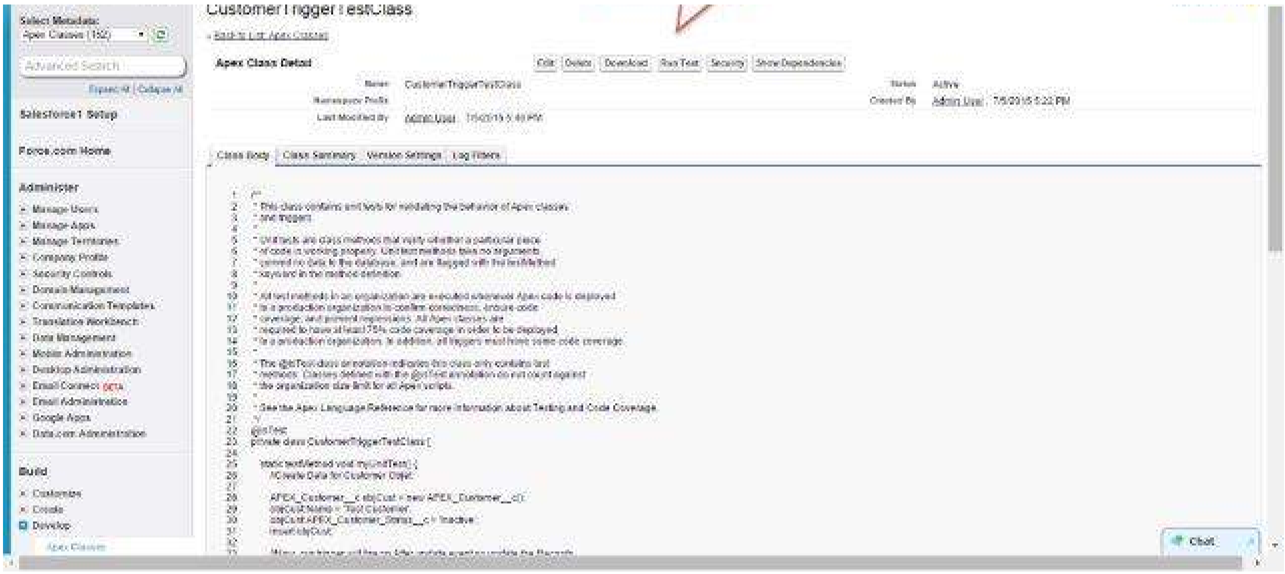
```
/**
 * This class contains unit tests for validating the behavior of Apex classes
 * and triggers.
 *
 * Unit tests are class methods that verify whether a particular piece
 * of code is working properly. Unit test methods take no arguments,
 * commit no data to the database, and are flagged with the testMethod
 * keyword in the method definition.
 *
 * All test methods in an organization are executed whenever Apex code is deployed
 * to a production organization to confirm correctness, ensure code
 * coverage, and prevent regressions. All Apex classes are
 * required to have at least 75% code coverage in order to be deployed
 * to a production organization. In addition, all triggers must have some code coverage.
 *
 * The @isTest class annotation indicates this class only contains test
 * methods. Classes defined with the @isTest annotation do not count against
 * the organization size limit for all Apex scripts.
 *
 * See the Apex Language Reference for more information about Testing and Code Coverage.
 */
@isTest
private class CustomerTriggerTestClass {
    static testMethod void myUnitTest() {
        //Create Data for Customer Objet

        APEX_Customer__c objCust = new APEX_Customer__c();
        objCust.Name = 'Test Customer';
        objCust.APEX_Customer_Status__c = 'Inactive';
        insert objCust;

        //Now, our trigger will fire on After update event so update the Records
        Test.startTest();//Starts the scope of test
        objCust.APEX_Customer_Status__c  = 'Active';
        update objCust;
        Test.stopTest();//Ends the scope of test

        //Now check if it is giving desired results using system.assert Statement.New
invoice should be created
        List<apex_invoice__c> invList = [SELECT Id, APEX_Customer__c FROM
APEX_Invoice__c WHERE APEX_Customer__c = :objCust.id];
        system.assertEquals(1,invList.size());//Check if one record is created in
Invoivce sObject
    }
}
```

Running the Test Class

Follow the below steps to run the test class:

Step 1: Go to Apex classes=>click on the class name 'CustomerTriggerTestClass'

Step 2: Click on Run Test button as shown:



CustomerTriggerTestClass

« Back to List: Apex Classes

Apex Class Detail

Name: CustomerTriggerTestClass

Namespace Prefix:

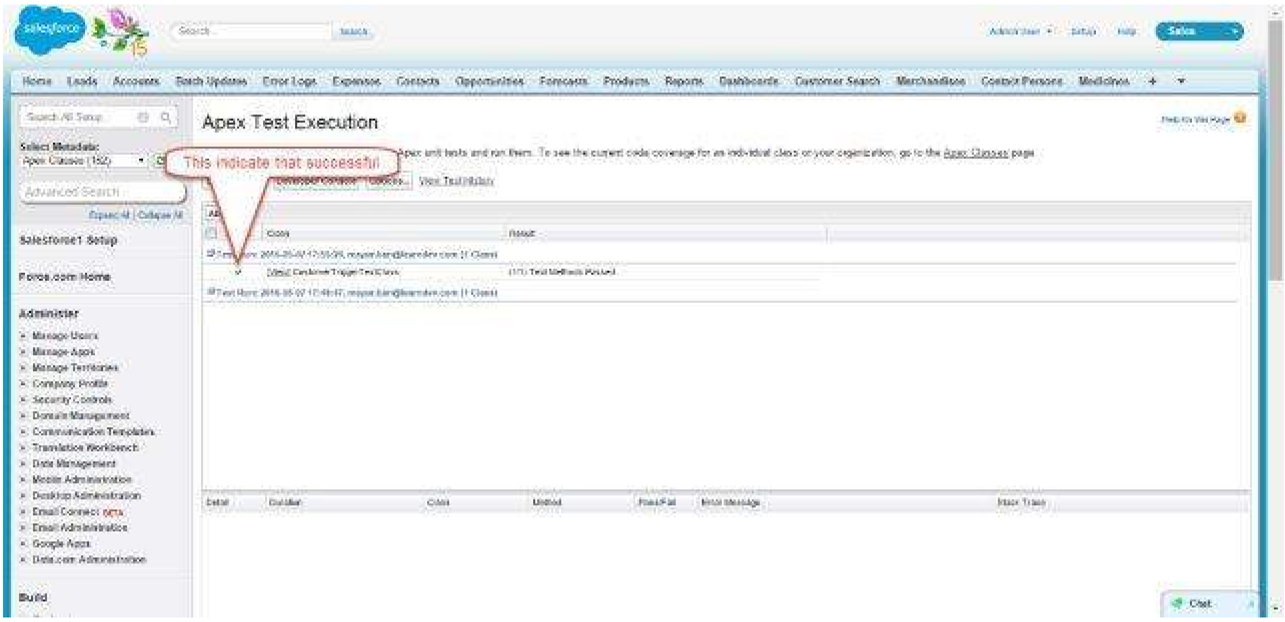
Last Modified By: Admin User, 7/6/2015 5:40 PM

Status: Active

Created By: Admin User, 7/6/2015 5:22 PM

```
/**
 * This class contains unit tests for validating the behavior of Apex classes
 * and triggers.
 *
 * Unit tests are class methods that verify whether a particular piece
 * of code is working properly. Unit test methods take no arguments,
 * commit no data to the database, and are flagged with the testMethod
 * keyword in the method definition.
 *
 * All test methods in an organization are executed whenever Apex code is deployed
 * to a production organization to confirm correctness, ensure code
 * coverage, and prevent regressions. All Apex classes are
 * required to have at least 75% code coverage in order to be deployed
 * to a production organization. In addition, all triggers must have some code coverage.
 *
 * The @isTest class annotation indicates this class only contains test
 * methods. Classes defined with the @isTest annotation do not count against
 * the organization size limit for all Apex scripts.
 *
 * See the Apex Language Reference for more information about Testing and Code Coverage.
 */
@isTest
private class CustomerTriggerTestClass {

    static testMethod void myUnitTest() {
        //Create Data for Customer Objet

        APEX_Customer__c objCust = new APEX_Customer__c();
        objcust.Name = 'Test Customer';
        objCust.APEX_Customer_Status__c = 'Inactive';
        insert objCust;

        // Now, our trigger will fire on After update event so update the Records
```

Step 3: Check status



Apex Test Execution

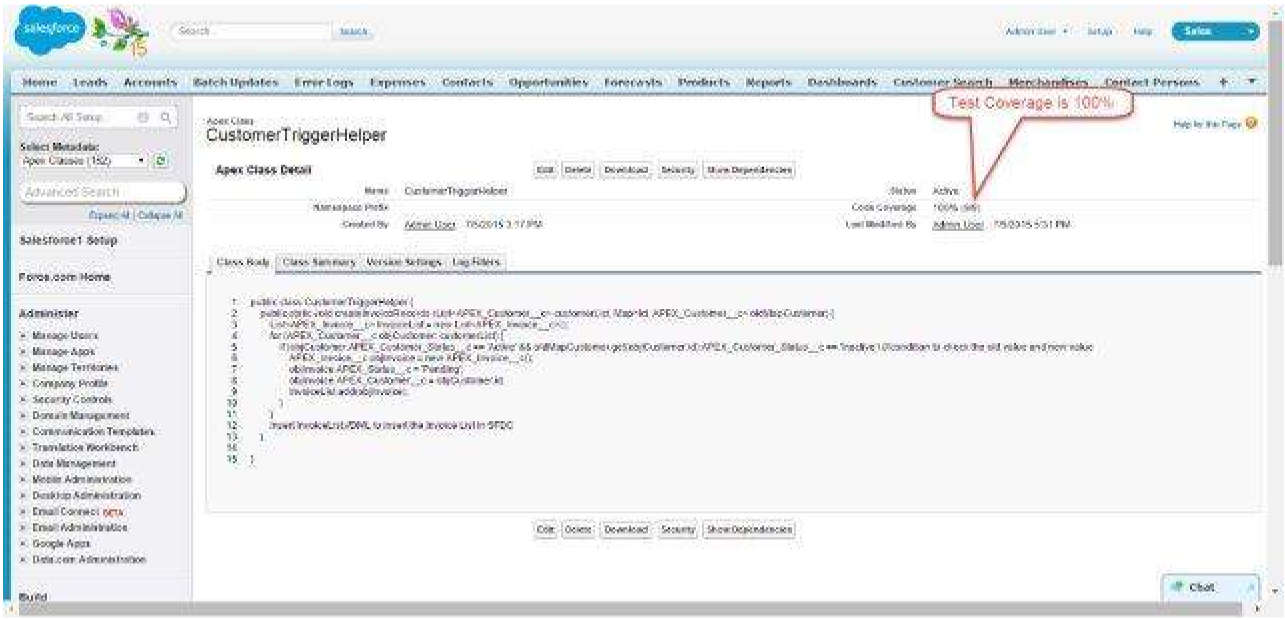
…Apex unit tests and run them. To see the current code coverage for an individual class or your organization, go to the Apex Classes page.

	Class	Result
Test Run: 2015-06-07 17:55:36, mayur.kan@learndev.com (1 Class)		
✓	[View] CustomerTriggerTestClass	(1/1) Test Methods Passed
Test Run: 2015-06-07 17:46:47, mayur.kan@learndev.com (1 Class)		

Detail	Duration	Class	Method	Pass/Fail	Error Message	Stack Trace

Step 4: Now check the class and trigger for which we have written the test

Class:



Apex Class

CustomerTriggerHelper

Apex Class Detail

Name: CustomerTriggerHelper

Namespace Prefix:

Created By: Admin User, 7/6/2015 3:17 PM

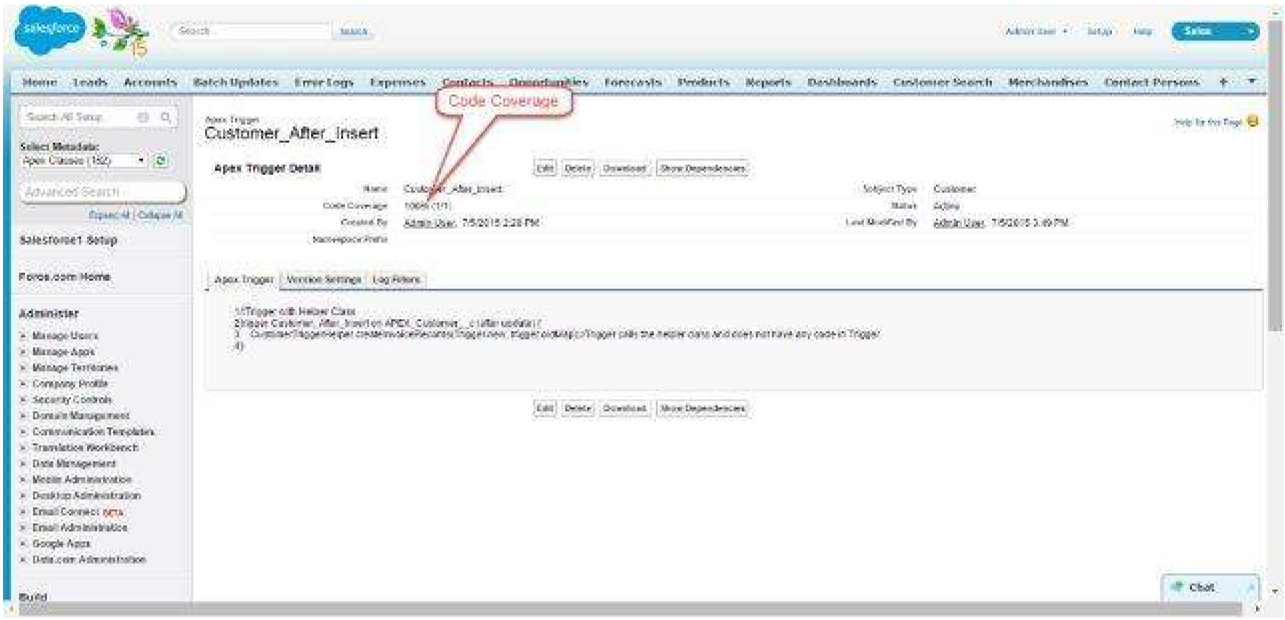
Status: Active

Code Coverage: 100% (9/9)

Last Modified By: Admin User, 7/6/2015 5:51 PM

```
public class CustomerTriggerHelper {
    public static void createInvoiceRecords (List<APEX_Customer__c> customerList, Map<Id, APEX_Customer__c> oldMapCustomer) {
        List<APEX_Invoice__c> InvoiceList = new List<APEX_Invoice__c>();
        for (APEX_Customer__c objCustomer: customerList) {
            if (objCustomer.APEX_Customer_Status__c == 'Active' && oldMapCustomer.get(objCustomer.id).APEX_Customer_Status__c == 'Inactive') {//condition to check the old value and new value
                APEX_Invoice__c objInvoice = new APEX_Invoice__c();
                objInvoice.APEX_Status__c = 'Pending';
                objInvoice.APEX_Customer__c = objCustomer.id;
                InvoiceList.add(objInvoice);
            }
        }
        insert InvoiceList;//DML to insert the Invoice List in SFDC
    }

}
```

Trigger:



Apex Trigger
Customer_After_Insert

Apex Trigger Detail

Name	Customer_After_Insert	Object Type	Customer
Code Coverage	100% (1/1)	Status	Active
Created By	Admin User, 7/5/2015 2:28 PM	Last Modified By	Admin User, 7/5/2015 3:49 PM
Namespace Prefix			

```
//Trigger with Helper Class
trigger Customer_After_Insert on APEX_Customer__c (after update) {
   CustomerTriggerHelper.createInvoiceRecords(Trigger.new, trigger.oldMap);//Trigger calls the helper class and does not have any code in Trigger
}
```