Apex Triggers

Get StartedGet Started with Apex Triggers with Apex Triggers :-

```
trigger AccountAddressTrigger on Account (before insert,before update) {
    for (Account account : trigger.new){
        if(account.Match_Billing_Address__c==true){
            account.ShippingPostalCode=account.BillingPostalCode;
        }
    }
}
```

Bulk Apex Triggers

```
trigger ClosedOpportunityTrigger on Opportunity (after insert,after update) {
List<Task> tasklist = new List<Task>();
    for(opportunity opp: Trigger.New){
        if(opp.StageName == 'Closed Won'){
            tasklist.add(new Task(Subject= 'Follow Up Test Task', Whatid=opp.Id));
        }
    }
    if(tasklist.size()>0){
        insert tasklist;
    }
}
```

Apex Testing

```
 @isTest
  public class TestRestrictContactByName {

  @isTest static void Test_insertupdateContact(){
      Contact cnt = new Contact();
      cnt.LastName = 'INVALIDNAME';
      Test.startTest();
      Database.SaveResult result = Database.insert(cnt, false);
      Test.stopTest();

      System.assert(!result.isSuccess());
      System.assert(result.getErrors().size() > 0);
      System.assertEquals('The Last Name "INVALIDNAME" is not allowed for
DML',result.getErrors()[0].getMessage());
  }
}
```

Create Test Data for Apex Tests

```
public class RandomContactFactory {
    public static List<Contact> generateRandomContacts(Integer numcnt, string lastname){
        List<Contact> contacts = new List<Contact>();
        for (Integer i=0;i<numcnt;i++){
            Contact cnt= new Contact(FirstName = 'Test'+i, LastName= lastname);
            contacts.add(cnt);
        }
        return contacts;
    }

}
```

Asynchronous Apex

Use Future Methods

```
public class AccountProcessor {
   @future
   public static void countContacts(List<Id> accountIds){
      List<Account> accounts = [Select Id, Name from Account Where Id IN : accountIds];
      List<Account> updatedAccounts = new List<Account>();
      for(Account account : accounts){
        account.Number_of_Contacts__c = [Select count() from Contact Where AccountId =:
account.Id];
         System.debug('No Of Contacts = ' + account.Number_of_Contacts__c);
         updatedAccounts.add(account);
      }
      update updatedAccounts;
   }

}
```

For Test Class-

```
@isTest
public class AccountProcessorTest {
   @isTest
   public static void testNoOfContacts(){
      Account a = new Account();
      a.Name = 'Test Account';
      Insert a;

      Contact c = new Contact();
      c.FirstName = 'Bob';
      c.LastName =  'Willie';
      c.AccountId = a.Id;

      Contact c2 = new Contact();
      c2.FirstName = 'Tom';
      c2.LastName = 'Cruise';
```

```
        c2.AccountId = a.Id;

        List<Id> acctIds = new List<Id>();
        acctIds.add(a.Id);

        Test.startTest();
        AccountProcessor.countContacts(acctIds);
        Test.stopTest();
    }

}
```

Apex Integration Services

Apex REST Callouts

AnimalLocator Class

```
public class AnimalLocator {
   public static string getAnimalNameById(Integer animalId){
      Http http=new Http();
      HttpRequest req=new HttpRequest();
      req.setEndPoint('https://th-apex-http-callout.herokuapp.com/animals/' + animalId);
      req.setMethod('GET');
      HttpResponse res=http.send(req);
      Map<String,Object> animals = new Map<String,Object>();
      if (res.getStatusCode() == 200) {
         Map<String,Object> results = (Map<String,Object>)JSON.deserializeUntyped(res.getBody());
         animals = (Map<String,Object>)results.get('animal');
      }
     else{System.debug('The status code returned was not expected: ' + res.getStatusCode() + ' ' +
res.getStatus());}
      return (string)animals.get('name');
   }
}
```

AnimalLocatorMock Class

```
@isTest
global class AnimalLocatorMock implements HttpCalloutMock {
   global HttpResponse respond(HttpRequest request){
        HttpResponse response = new HttpResponse();
      response.setHeader('Content-Type', 'application/json');
      response.setBody('{"animal":{"id":1,"name":"chicken","eats":"chicken food","says":"cluck cluck"}}');
      //response.setBody('{"animal":{"id":1,"name":"chicken"}}');
      response.setStatusCode(200);
      return response;
   }
}
```

AnimalLocatorTest Class

```
@isTest
public with sharing class AnimalLocatorTest {
   @isTest
    static void testGetCallout() {
      Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
      String result = AnimalLocator.getAnimalNameById(1);
      String expectedResult = 'Chicken';
      System.assertEquals(result,expectedResult);
      result = AnimalLocator.getAnimalNameById(4);
      expectedResult = 'Could not find an Animal with a matching ID';
      System.assertEquals(result,expectedResult);
   }
}
```

Apex SOAP Callouts

ParkLocator

```
public class ParkLocator {
   public  static string[] country(String country) {
      ParkService.ParksImplPort prk = new ParkService.ParksImplPort();
      return prk.byCountry(country);
   }
}
```

ParkServiceMock

```
@isTest
global class ParkServiceMock implements WebServiceMock {
  global void doInvoke(
         Object stub,
         Object request,
         Map<String, Object> response,
         String endpoint,
         String soapAction,
         String requestName,
         String responseNS,
         String responseName,
```

```
        String responseType) {
      parkService.byCountryResponse response_x = new parkService.byCountryResponse();
      response_x.return_x = new List<String>{'Lal Bhag', 'Cubbon Park', 'Pazhassi Dam'};
      response.put('response_x', response_x);
  }
}
```

ParkLocatorTest

```
@isTest
private class ParkLocatorTest {
   @isTest static void testCallout() {

      Test.setMock(WebServiceMock.class, new ParkServiceMock());
      String country = 'India';
      System.assertEquals(new List<String>{'Lal Bhag', 'Cubbon Park', 'Pazhassi Dam'},
ParkLocator.country(country));
   }
}
```

Apex Web Services

AccountManager

```
@RestResource(urlMapping='/Accounts/*/contacts')
global with sharing class AccountManager{
   @HttpGet
   global static Account getAccount(){
      RestRequest request = RestContext.request;
      String accountId = request.requestURI.substringBetween('Accounts/','/contacts');
      system.debug(accountId);
      Account objAccount = [SELECT Id,Name,(SELECT Id,Name FROM Contacts) FROM Account
WHERE Id = :accountId LIMIT 1];
      return objAccount;
   }
}
```

AccountManagerTest

```
@isTest
private class AccountManagerTest{
    static testMethod void testMethod1(){
        Account objAccount = new Account(Name = 'test Account');
        insert objAccount;
        Contact objContact = new Contact(LastName = 'test Contact',
                                         AccountId = objAccount.Id);
        insert objContact;
        Id recordId = objAccount.Id;
        RestRequest request = new RestRequest();
        request.requestUri =
            'https://sandeepidentity-dev-ed.my.salesforce.com/services/apexrest/Accounts/'
            + recordId +'/contacts';
        request.httpMethod = 'GET';
        RestContext.request = request;
        // Call the method to test
        Account thisAccount = AccountManager.getAccount();
        // Verify results
        System.assert(thisAccount!= null);
        System.assertEquals('test Account', thisAccount.Name);
    }
}
```