# Apex Triggers

## Get Started with Apex Triggers

### AccountAddressTrigger

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

### ClosedOpportunityTrigger

|  |
| --- |
| 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 Get Started with Apex Unit Tests

### VerifyDate

public class VerifyDate {  
   
 //method to handle potential checks against two dates  
 public static Date CheckDates(Date date1, Date date2) {  
 //if date2 is within the next 30 days of date1, use date2. Otherwise use the end of the month  
 if(DateWithin30Days(date1,date2)) {  
 return date2;  
 } else {  
 return SetEndOfMonthDate(date1);  
 }  
 }  
   
 //method to check if date2 is within the next 30 days of date1  
 @TestVisible private static Boolean DateWithin30Days(Date date1, Date date2) {  
 //check for date2 being in the past  
 if( date2 < date1) { return false; }  
   
 //check that date2 is within (>=) 30 days of date1  
 Date date30Days = date1.addDays(30); //create a date 30 days away from date1  
 if( date2 >= date30Days ) { return false; }  
 else { return true; }  
 }  
  
 //method to return the end of the month of a given date  
 @TestVisible private static Date SetEndOfMonthDate(Date date1) {  
 Integer totalDays = Date.daysInMonth(date1.year(), date1.month());  
 Date lastDay = Date.newInstance(date1.year(), date1.month(), totalDays);  
 return lastDay;  
 }  
  
}

### TestVerifyDate

@isTest  
private class TestVerifyDate {  
   
 @isTest static void Test\_CheckDates\_case1(){  
 Date d = VerifyDate.CheckDates(date.parse('01/01/2020'), date.parse('01/05/2020'));  
 System.assertEquals(date.parse('01/01/2020'), D);  
 }  
   
 @isTest static void Test\_CheckDates\_case2(){  
 Date d = VerifyDate.CheckDates(date.parse('01/01/2020'), date.parse('05/05/2020'));  
 System.assertEquals(date.parse('01/31/2020'), D);  
}  
   
 @isTest static void Test\_DateWithin30Days\_case1(){  
 Boolean flag = VerifyDate.DateWithin30Days(date.parse('01/01/2020'), date.parse('12/30/2019'));  
 System.assertEquals(false, flag);  
 }  
   
 @isTest static void Test\_DateWithin30Days\_case2(){  
 Boolean flag = VerifyDate.DateWithin30Days(date.parse('01/01/2020'), date.parse('02/02/2019'));  
 System.assertEquals(false, flag);  
 }  
   
 @isTest static void Test\_DateWithin30Days\_case3(){  
 Boolean flag = VerifyDate.DateWithin30Days(date.parse('01/01/2020'), date.parse('01/15/2020'));  
 System.assertEquals(true, flag);  
 }  
   
 @isTest static void Test\_SetEndOfMonthDate(){  
 Date returndate = VerifyDate.SetEndOfMonthDate(date.parse('01/01/2020'));  
 }  
}

## Test Apex Triggers

### RestrictContactByName

|  |
| --- |
| trigger RestrictContactByName on Contact (before insert, before update) {    //check contacts prior to insert or update for invalid data  For (Contact c : Trigger.New) {  if(c.LastName == 'INVALIDNAME') { //invalidname is invalid  c.AddError('The Last Name "'+c.LastName+'" is not allowed for DML');  }   }   } |

### TestRestrictContactByName

@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

### RandomContactFactory

public class RandomContactFactory {  
  
 public static List<Contact> generateRandomContacts(Integer num, String lastName){  
 List<Contact> contactList = new List<Contact>();  
 for(Integer i = 1; i<=num; i++){  
 Contact ct = new Contact(FirstName = 'Test'+i, LastName =lastName);  
 contactList.add(ct);  
 }  
 return contactList;  
 }  
}

# Apex Integration Services

## Apex REST Callouts

### AnimalLocator

public class AnimalLocator {  
 public static String getAnimalNameById(Integer x){  
 Http http = new Http();  
HttpRequest req = new HttpRequest();  
req.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/' + x);  
req.setMethod('GET');  
 Map<String, Object> animal= new Map<String, Object>();  
HttpResponse res = http.send(req);  
if(res.getStatusCode() == 200) {  
Map<String, Object> results = (Map<String, Object>)JSON.deserializeUntyped(res.getBody());  
 animal = (Map<String, Object>) results.get('animal');  
}  
 return (String)animal.get('name');  
 }  
}

### AnimalLocatorTest

@isTest  
private class AnimalLocatorTest{  
 @isTest static void AnimalLocatorMock(){  
 Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());  
 string result = AnimalLocator.getAnimalNameById(3);  
 string expectedResult = 'chicken';  
 System.assertEquals(result,expectedResult);  
 }  
}

### AnimalLocatorMock

@isTest  
global class AnimalLocatorMock implements HttpCalloutMock {  
 // Implement this interface method  
 global HTTPResponse respond(HTTPRequest request) {  
 // Create a fake response  
 HttpResponse response = new HttpResponse();  
 response.setHeader('Content-Type', 'application/json');  
 response.setBody('{"animals": ["majestic badger", "fluffy bunny", "scary bear", "chicken", "mighty moose"]}');  
 response.setStatusCode(200);  
 return response;   
 }  
}

## Apex SOAP Callouts

### ParkLocator

public class ParkLocator {  
 public static string[] country(String country) {  
 parkService.parksImplPort park = new parkService.parksImplPort();  
 return park.byCountry(country);  
 }  
}

### ParkLocatorTest

|  |
| --- |
| @isTest private class ParkLocatorTest {  @isTest static void testCallout() {   // This causes a fake response to be generated  Test.setMock(WebServiceMock.class, new ParkServiceMock());  // Call the method that invokes a callout  // Double x = 1.0;  // Double result = AwesomeCalculator.add(x, y);    String country = 'Germany';  String[] result = ParkLocator.Country(country);    // Verify that a fake result is returned  System.assertEquals(new List<String>{'Hamburg Wadden Sea National Park', 'Hainich National Park'},result);   } } 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');  Account result = [Select Id, Name,(Select Id, Name from contacts) from Account where Id=:accountId Limit 1];  return result;  }  } AccountManagerTest @IsTest private class AccountManagerTest{   @isTest static void testAccountManager() {  Id recordId = createTestRecord();    Contact objCont = new Contact();  objCont.LastName = 'Test1';  objCont.AccountId = recordId ;  insert objCont ;   Contact objCont2 = new Contact();  objCont2 .LastName = 'Test2';  objCont2 .AccountId = recordId ;  insert objCont2 ;    // Set up a test request  RestRequest request = new RestRequest();  request.requestUri =  'https://na1.salesforce.com/services/apexrest/Accounts/'+recordId+'/contacts';  request.httpMethod = 'GET';  RestContext.request = request;  // Call the method to test  Account lst = AccountManager.getAccount();  //Add assert to check the list of size  }   // Helper method  static Id createTestRecord() {  // Create test record  Account accTest = new Account(Name='Test Record');  insert accTest ;  return accTest.Id;  }   } Asynchronous ApexUse Future MethodsAccountProcessor public class AccountProcessor  {  @future  public static void countContacts(List<id> accountIds)   {  List<Account> accountsToUpdate = new List<Account>();  List<Account> accounts = [Select id,Name,(Select Id from Contacts ) from Account where Id in :accountIds];  For(Account acc : accounts)  {  List<Contact> contactList = acc.Contacts ;    acc.Number\_of\_Contacts\_\_c = contactList.size();  accountsToUpdate.add(acc);  }  } } AccountProcessorTest @isTest private class AccountProcessorTest  {  @isTest  private static void countContactsTest()  {  Account acc = new Account(Name = 'Test Account');  insert acc;    Contact c1 = new Contact(FirstName='Abc',  LastName='xyz',  AccountId=acc.Id);  insert c1;    Contact c2 = new Contact(FirstName='qwe',  LastName='rty',  AccountId=acc.Id);  insert c2;    List<Id> Idls = new List<Id>();   Idls.add(acc.Id);  //starting the test  Test.startTest();  AccountProcessor.countContacts(Idls);  Test.stopTest();      }  } |

## Use Batch Apex

### LeadProcessor

public class LeadProcessor implements Database.Batchable<sObject> {  
   
 public Database.QueryLocator start(Database.BatchableContext bc) {  
 return Database.getQueryLocator(  
 [SELECT Id, Name FROM Lead]  
 );  
 }  
 public void execute(Database.BatchableContext bc, List<Lead> leads){  
 for(Lead l: leads){  
 l.LeadSource = 'Dreamforce';  
 }  
 update leads;  
 }  
   
 public void finish(Database.BatchableContext bc){  
 System.debug('Done');   
 }  
 }

## LeadProcessorTest

@isTest  
private class LeadProcessorTest {  
 @isTest  
 private static void testLeadProcessorBatch(){  
   
 // Load Test Data  
 List<Lead> leads = new List<Lead>();  
 for(Integer i=1;i<=200;i++){  
 leads.add(new Lead(LastName='LeadLstName', Company='SalesForce'));  
 }  
 insert leads;  
 // perfor test   
 Test.startTest();  
 LeadProcessor lp = new LeadProcessor();  
 Id batchId = Database.executeBatch(lp,200);  
 Test.stopTest();  
   
 //check result  
 List<Lead> updatedLeads = [SELECT Id FROM Lead WHERE LeadSource='Dreamforce'];  
 System.assertEquals(200, updatedLeads.size(),'ERROR: at least 1 Lead record not updated correctly');  
 }  
}

## Control Processes with Queueable Apex

## AddPrimaryContact

public class AddPrimaryContact implements Queueable  
{  
 private Contact c;  
 private String state;  
 public AddPrimaryContact(Contact c, String state)  
 {  
 this.c = c;  
 this.state = state;  
 }  
 public void execute(QueueableContext context)   
 {  
 List<Account> ListAccount = [SELECT ID, Name ,(Select id,FirstName,LastName from contacts ) FROM ACCOUNT WHERE BillingState = :state LIMIT 200];  
 List<Contact> lstContact = new List<Contact>();  
 for (Account acc:ListAccount)  
 {  
 Contact cont = c.clone(false,false,false,false);  
 cont.AccountId = acc.id;  
 lstContact.add( cont );  
 }  
   
 if(lstContact.size() >0 )  
 {  
 insert lstContact;  
 }  
   
 }  
}

### AddPrimaryContactTest

@isTest  
public class AddPrimaryContactTest   
{  
 @isTest static void TestList()  
 {  
 List<Account> Teste = new List <Account>();  
 for(Integer i=0;i<50;i++)  
 {  
 Teste.add(new Account(BillingState = 'CA', name = 'Test'+i));  
 }  
 for(Integer j=0;j<50;j++)  
 {  
 Teste.add(new Account(BillingState = 'NY', name = 'Test'+j));  
 }  
 insert Teste;  
  
 Contact co = new Contact();  
 co.FirstName='demo';  
 co.LastName ='demo';  
 insert co;  
 String state = 'CA';  
   
 AddPrimaryContact apc = new AddPrimaryContact(co, state);  
 Test.startTest();  
 System.enqueueJob(apc);  
 Test.stopTest();  
 }  
 }

## Schedule Jobs Using the Apex Scheduler

### DailyLeadProcessor

global class DailyLeadProcessor implements Schedulable{  
 global void execute(SchedulableContext ctx){  
 List<Lead> leads = [SELECT Id, LeadSource FROM Lead WHERE LeadSource = ''];  
   
 if(leads.size() > 0){  
 List<Lead> newLeads = new List<Lead>();  
   
 for(Lead lead : leads){  
 lead.LeadSource = 'DreamForce';  
 newLeads.add(lead);  
 }  
   
 update newLeads;  
 }  
 }  
}

### DailyLeadProcessorTest

@isTest  
private class DailyLeadProcessorTest{  
 //Seconds Minutes Hours Day\_of\_month Month Day\_of\_week optional\_year  
 public static String CRON\_EXP = '0 0 0 2 6 ? 2022';  
   
 static testmethod void testScheduledJob(){  
 List<Lead> leads = new List<Lead>();  
   
 for(Integer i = 0; i < 200; i++){  
 Lead lead = new Lead(LastName = 'Test ' + i, LeadSource = '', Company = 'Test Company ' + i, Status = 'Open - Not Contacted');  
 leads.add(lead);  
 }  
   
 insert leads;  
   
 Test.startTest();  
 // Schedule the test job  
 String jobId = System.schedule('Update LeadSource to DreamForce', CRON\_EXP, new DailyLeadProcessor());  
   
 // Stopping the test will run the job synchronously  
 Test.stopTest();  
 }  
}