

## **APEX TRIGGERS**

### **•GET STARTED WITH APEX TRIGGERS:**

#### **1.AccountAddressTrigger.apxt**

```
trigger AccountAddressTrigger on Account (before insert, before update) {
    for(Account a: Trigger.New){
        if(a.Match_Billing_Address__c == true && a.BillingPostalCode!= null){
            a.ShippingPostalCode=a.BillingPostalCode;
        }
    }
}
```

### **•BULK APEX TRIGGERS:**

#### **1.ClosedOpportunityTrigger.apxt**

```
trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {
    List<Task> taskList = new List<Task>();
    for(Opportunity opp : [SELECT Id, StageName FROM Opportunity WHERE
    StageName='Closed Won' AND Id IN : Trigger.New]){
        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 TEST:**

#### **1.VerifyDate.apxc**

```
public class VerifyDate {
    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);
        }
    }
}
```

```

}
private static Boolean DateWithin30Days(Date date1, Date date2) {
    Date date30Days = date1.addDays(30); //create a date 30 days away from date1
    if( date2 > date30Days ) { return false; }
    else { return true; }
}
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;
}
}

```

## **2.TestVerifyDate.apxc**

```

@isTest
private class TestVerifyDate {
    @isTest static void testCheckDates() {
        Date now = Date.today();
        Date lastOfTheMonth = Date.newInstance(now.year(), now.month(),
        Date.daysInMonth(now.year(), now.month()));
        Date plus60 = Date.today().addDays(60);
        Date d1 = VerifyDate.CheckDates(now, now);
        System.assertEquals(now, d1);
        Date d2 = VerifyDate.CheckDates(now, plus60);
        System.assertEquals(lastOfTheMonth, d2);
    }
}

```

## **•TEST APEX TRIGGERS:**

### **1.RestrictContactByName.apxt**

```

trigger RestrictContactByName on Contact (before insert) {
    For (Contact c : Trigger.New) {
        if(c.LastName == 'INVALIDNAME') { //invalidname is invalid
            c.AddError('The Last Name "'+c.LastName+'" is not allowed for DML');
        }
    }
}

```

## •CREATE TEST DATA FOR APEX TESTS:

### 1.RandomContactFactory.apxc

```
public class RandomContactFactory {  
    public static List<Contact> generateRandomContacts(Integer num, String lastName) {  
        List<Contact> contacts = new List<Contact>();  
        for (Integer i = 0; i < num; i++) {  
            Contact c = new Contact(FirstName=i.format(), LastName=lastName);  
            contacts.add(c);  
        }  
        return contacts;  
    }  
}
```

## ASYNCHRONOUS APEX

## •USE FUTURE METHODS:

### 1.AccountProcessor.apxc

```
public without sharing class AccountProcessor {  
    //Add annotation to declare a future method  
    @future(callout=false)  
    public static void countContacts(List<Id> accountIds){  
        //Query all accounts in the list of Ids passed  
        Map<Id, Account> accountMap = new Map<Id, Account>([SELECT Id, (SELECT Id  
FROM Contacts) FROM Account WHERE Id IN:accountIds]);  
        List<Account> listName = new List<Account>();  
        //Loop through list of accounts  
        for(Account a: accountMap.values()){  
            //Assign field to number of contact  
            a.Number_of_Contacts__c=accountMap.get(a.Id).Contacts.size();  
        }  
        //Update Accounts  
        update accountMap.values();  
    }  
}
```

## 2.AccountProcessorTest.apxc

```
@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();
    }
}
```

### •USE BATCH APEX:

#### 1.LeadProcessor.apxc

global class LeadProcessor implements

```
Database.Batchable<sObject>, Database.Stateful {
    // instance member to retain state across transactions
    global Integer recordsProcessed = 0;
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, LeadSource FROM Lead');
    }
    global void execute(Database.BatchableContext bc, List<Lead> scope){
        // process each batch of records
        List<Lead> leads = new List<Lead>();
        for (Lead lead : scope) {
            lead.LeadSource = 'Dreamforce';
        }
    }
}
```

```

        // increment the instance member counter
        recordsProcessed = recordsProcessed + 1;
    }
    update leads;
}
global void finish(Database.BatchableContext bc){
    System.debug(recordsProcessed + ' records processed. Shazam!');
}
}

```

## 2.LeadProcessorTest.apxc

```

@isTest
public class LeadProcessorTest {
    @testSetup
    static void setup() {
        List<Lead> leads = new List<Lead>();
        // insert 200 leads
        for (Integer i=0;i<200;i++) {
            leads.add(new Lead(LastName='Lead '+i,
                Company='Lead', Status='Open - Not Contacted'));
        }
        insert leads;
    }
    static testmethod void test() {
        Test.startTest();
        LeadProcessor lp = new LeadProcessor();
        Id batchId = Database.executeBatch(lp, 200);
        Test.stopTest();
        // after the testing stops, assert records were updated properly
        System.assertEquals(200, [select count() from lead where LeadSource =
'Dreamforce']);
    }
}

```

## •CONTROL PROCESSES WITH QUEUEABLE APEX:

## 1.AddPrimaryContact.apxc

```
public class AddPrimaryContact implements Queueable {
    private Contact contactObj;
    private String state_code;
    public AddPrimaryContact(Contact c, String s) {
        this.contactObj = c;
        this.state_code = s;
    }
    public void execute(QueueableContext context) {
        List<Account> accounts = [SELECT Id
                                FROM Account
                                WHERE BillingState = :this.state_code
                                LIMIT 200];
        List<Contact> contacts = new List<Contact>();
        for (Account a : accounts) {
            Contact c = this.contactObj.clone(false, false, false, false);
            c.AccountId = a.Id;
            contacts.add(c);
        }
        if (contacts.size() > 0) {
            insert contacts;
        }
    }
}
```

## 2.AddPrimaryContactTest.apxc

```
@isTest
public class AddPrimaryContactTest{
    @testSetup
    static void setup(){
        List<Account> lstOfAcc = new List<Account>();
        for(Integer i = 1; i <= 100; i++){
            if(i <= 50)
                lstOfAcc.add(new Account(name='AC'+i, BillingState = 'NY'));
            else
                lstOfAcc.add(new Account(name='AC'+i, BillingState = 'CA'));
        }
        INSERT lstOfAcc;
    }
}
```

```

    }
    static testmethod void testAddPrimaryContact(){
        Contact con = new Contact(LastName = 'TestCont');
        AddPrimaryContact addPCIns = new AddPrimaryContact(CON,'CA');
        Test.startTest();
        System.enqueueJob(addPCIns);
        Test.stopTest();
        System.assertEquals(50, [select count() from Contact]);
    }
}

```

## •SCHEDULE JOBS USING APEX SCHEDULER:

### 1.DailyLeadProcessor.apxc

```

public class DailyLeadProcessor implements Schedulable {
    Public void execute(SchedulableContext SC){
        List<Lead> LeadObj=[SELECT Id from Lead where LeadSource=null limit 200];
        for(Lead l:LeadObj){
            l.LeadSource='Dreamforce';
            update l;
        }
    }
}

```

### 2.DailyLeadProcessorTest.apxc

```

@isTest
private class DailyLeadProcessorTest {
    static testMethod void testDailyLeadProcessor() {
        String CRON_EXP = '0 0 1 * * ?';
        List<Lead> IList = new List<Lead>();
        for (Integer i = 0; i < 200; i++) {
            IList.add(new Lead(LastName='Dreamforce'+i, Company='Test1 Inc.',
            Status='Open - Not Contacted'));
        }
        insert IList;
        Test.startTest();
        String jobId = System.schedule('DailyLeadProcessor', CRON_EXP, new
        DailyLeadProcessor());
    }
}

```

```
}
```

## APEX INTEGRATION SERVICES

### •APEX REST CALLOUTS:

#### 1. AnimalLocator.apxc

```
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');
    }
}
```

#### 2. AnimalLocatorMock.apxc

```
@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.getStatusCode(200);
        return response;
    }
}
```

#### 3. AnimalLocatorTest.apxc



```

@isTest
private class AnimalLocatorTest{
    @isTest static void AnimalLocatorMock1() {
        Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
        string result = AnimalLocator.getAnimalNameById(3);
        String expectedResult = 'chicken';
        System.assertEquals(result,expectedResult );
    }
}

```

## •APEX SOAP CALLOUTS:

### 1.ParkLocator.apxc

```

public class ParkLocator {
    public static string[] country(string theCountry) {
        ParkService.ParksImplPort parkSvc = new ParkService.ParksImplPort(); // remove
space
        return parkSvc.byCountry(theCountry);
    }
}

```

### 2.ParkLocatorTest.apxc

```

@isTest
private class ParkLocatorTest {
    @isTest static void testCallout() {
        Test.setMock(WebServiceMock.class, new ParkServiceMock ());
        String country = 'United States';
        List<String> result = ParkLocator.country(country);
        List<String> parks = new List<String>{'Yellowstone', 'Mackinac National Park',
'Yosemite'};
        System.assertEquals(parks, result);
    }
}

```

### 3.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> {'USA'};
    response.put ('response_x', response_x);
}
}

```

## •APEX WEB SERVICES:

### 1.AccountManager.apxc

```

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

```

### 2.AccountManagerTest.apxc

```

@IsTest
private class AccountManagerTest{
    @isTest static void testAccountManager(){
        Id recordId = getTestAccountId();
        // Set up a test request
        RestRequest request = new RestRequest();
        request.requestUri =
            'https://ap5.salesforce.com/services/apexrest/Accounts/'+ recordId +'/contacts';
        request.httpMethod = 'GET';
    }
}

```

```
RestContext.request = request;
// Call the method to test
Account acc = AccountManager.getAccount();
// Verify results
System.assert(acc != null);
}
private static Id getTestAccountId(){
    Account acc = new Account(Name = 'TestAcc2');
    Insert acc;
    Contact con = new Contact(LastName = 'TestCont2', AccountId = acc.Id);
    Insert con;
    return acc.Id;
}
}
```