Apex Triggers

-

ClosedOpportunityTrigger.apxt

```
trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {

    List<Task> taskList = new List<Task>();

    for(Opportunity opp : Trigger.new) {

    //Only create Follow Up Task only once when Opp StageName is to 'Closed Won' on Create
    if(Trigger.isInsert) {
      if(Opp.StageName == 'Closed Won') {
        taskList.add(new Task(Subject = 'Follow Up Test Task', WhatId = opp.Id));
      }
    }

    //Only create Follow Up Task only once when Opp StageName changed to 'Closed Won' on Update
    if(Trigger.isUpdate) {
      if(Opp.StageName == 'Closed Won'
      && Opp.StageName != Trigger.oldMap.get(opp.Id).StageName) {
        taskList.add(new Task(Subject = 'Follow Up Test Task', WhatId = opp.Id));
      }
    }
    }

    if(taskList.size()>0) {
        insert taskList;
    }
}
```

AccountAddressTrigger.apxt

```
trigger AccountAddressTrigger on Account (before insert, before update) {
    for(Account a:Trigger.New){
        if(a.Match_Billing_Address__c == True){
            a.ShippingStreet = a.BillingStreet;
            a.ShippingCity = a.BillingCity;
            a.ShippingState = a.BillingState;
            a.ShippingPostalCode = a.BillingPostalCode;
            a.ShippingCountry = a.BillingCountry;
        }
    }
}
```

RestrictContactByName.apxt

```
       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

      }

 }

}
```

AccountDeletion.apxt

```
trigger AccountDeletion on Account (before delete) {
    // Prevent the deletion of accounts if they have related opportunities.
    for (Account a : [SELECT Id FROM Account
                     WHERE Id IN (SELECT AccountId FROM Opportunity) AND
```

```
                        Id IN :Trigger.old]) {
            Trigger.oldMap.get(a.Id).addError(
                'Cannot delete account with related opportunities.');
        }
}
```

Apex Classes

VerifyDate.apxc

```
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
	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; }
	}

	/m
```

```
  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.apxc

```
@IsTest
public class TestVerifyDate {
    @isTest static void date2within31daydate1(){
        Date returnDate1 =
VerifyDate.CheckDates(date.valueOf('2022-05-

        //This should return may 16 2022 because this date is
WITHIN 31 Days of May 16 2022
        System.assertEquals(date.valueOf('2022-05-16'),
returnDate1);
    }
    @isTest static void date2NOTwithin31daydate(){
        Date returnDate2 =
VerifyDate.CheckDates(date.valueOf('2022-05-

        //This should return may 31 2022 because May 16 2022 is
NOT WITHIN 31 Days of May 16 2022
        System.assertEquals(date.valueOf('2022-05-31'),
returnDate2);
    }
}
```

TestRestrictContactByName.apxc

```
@IsTest
public class TestRestrictContactByName {
    @IsTest static void CreateBadContact(){
        Contact c = new
Contact(FirstName='Jeet',LastName='INVALIDNAME');
Test.startTest();
        Database.SaveResult result = Database.insert(c, false);
        Test.stopTest();
System.assert(!result.isSuccess());
    }
}
```

RandomContactFactory.apxc

```
public class RandomContactFactory {
    public static List<Contact> generateRandomContacts (Integer numOfContacts, string lastName){
        List<Contact> contacts = new List<Contact>();

        for(Integer i=0;i<numOfContacts;i++){
            Contact c = new Contact(FirstName = 'Test ' + i, LastName = lastName);
            contacts.add(c);
        }
        return contacts;
    }
}

```

AccountProcessor.apxc

```
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;
    }

}
```

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();
    }
}
```

LeadProcessor.apxc

```
public class LeadProcessor implements Database.Batchable<sObject>
{

     public Database.QueryLocator start(Database.BatchableContext bc) {
        // collect the batches of records or objects to be passed to execute
          return Database.getQueryLocator([Select LeadSource From Lead ]);
    }
    public void execute(Database.BatchableContext bc, List<Lead> leads){
         // process each batch of records
            for (Lead Lead : leads) {
                lead.LeadSource = 'Dreamforce';
            }
        update leads;
    }
    public void finish(Database.BatchableContext bc){
      }

}
```

LeadProcessorTest.apxc

```
@isTest
public class LeadProcessorTest {

        @testSetup
    static void setup() {
        List<Lead> leads = new List<Lead>();
        for(Integer counter=0 ;counter <200;counter++){
Lead lead = new Lead();
            lead.FirstName ='FirstName';
            lead.LastName ='LastName'+counter;
            lead.Company ='demo'+counter;
            leads.add(lead);
        }
        insert leads;
    }
@isTest static void test() {
        Test.startTest();
        LeadProcessor leadProcessor = new LeadProcessor();
        Id batchId = Database.executeBatch(leadProcessor);
        Test.stopTest();
    }

}
```

AddPrimaryContact.apxc

```
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.apxc

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

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;
        }
    }
}
```

DailyLeadProcessorTest.apxc

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

       Test.startTest();
       String jobId = System.schedule('DailyLeadProcessor',
CRON_EXP, new DailyLeadProcessor());
  }
}
```

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-

        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.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 );
    }
 }
```

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
        response.setStatusCode(200);
        return response;
    }
}
```

ParkLocator.apxc

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

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);
    }
}
```

ParkServiceMock.apxc

```
@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) {
        // start - specify the response you want to send
        ParkService.byCountryResponse response_x = new ParkService.byCountryResponse();
        response_x.return_x = new List<String>{'Yellowstone', 'Mackinac National Park', 'Yosemite'};
        // end
        response.put('response_x', response_x);
   }
}
```

AsyncParkService.apxc

```
public class AsyncParkService {
    public class byCountryResponseFuture extends System.WebServiceCalloutFuture {
        public String[] getValue() {
            ParkService.byCountryResponse response = (ParkService.byCountryResponse)System.WebServiceCallout.endInvoke(this);
            return response.return_x;
        }
    }
    public class AsyncParksImplPort {
        public String endpoint_x = 'https://th-apex-soap-

        public Map<String,String> inputHttpHeaders_x;
        public String clientCertName_x;
        public Integer timeout_x;
        private String[] ns_map_type_info = new String[]{'http://parks.services/', 'ParkService'};
        public AsyncParkService.byCountryResponseFuture
beginByCountry(System.Continuation continuation,String arg0) {
            ParkService.byCountry request_x = new ParkService.byCountry();
            request_x.arg0 = arg0;
            return (AsyncParkService.byCountryResponseFuture) System.WebServiceCallout.beginInvoke(
              this,
              request_x,
              AsyncParkService.byCountryResponseFuture.class,
              continuation,
              new String[]{endpoint_x,
              '',
              'http://parks.services/',
              'byCountry',
              'http://parks.services/',
              'byCountryResponse',
              'ParkService.byCountryResponse'}
            );
        }
    }
}
```

AccountManager.apxc

```
@RestResource(urlMapping='/Accounts/*/contacts')
global 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;
    }
}
```

AccountManagerTest.apxc

```
@isTest
global class AccountManagerTest {

    global static testMethod void getAccountTest1() {
        Id recordId = createTestRecord();
        // 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 thisAccount = AccountManager.getAccount();
        // Verify results
        System.assert(thisAccount != null);
        System.assertEquals('Test record', thisAccount.Name);

    }
    // Helper method
        static Id createTestRecord() {
        // Create test record
        Account TestAcc = new Account(
          Name='Test record');
        insert TestAcc;
        Contact TestCon= new Contact(
        LastName='Test',
        AccountId = TestAcc.id);
        return TestAcc.Id;
    }
}
```

Apex Pages

-

HeatMap.vfp

```
<apex:page applyBodyTag="false" applyHtmlTag="false"
standardStylesheets="false" showHeader="false">

    <apex:slds />

    <apex:remoteObjects >
        <apex:remoteObjectModel name="Property__c"
jsShorthand="Property">
            <apex:remoteObjectField name="Name"
jsShorthand="address"/>
            <apex:remoteObjectField name="City__c"
jsShorthand="city"/>
            <apex:remoteObjectField name="State__c"
jsShorthand="state"/>
            <apex:remoteObjectField name="Price__c"
jsShorthand="price"/>
            <apex:remoteObjectField name="Location__Latitude__s"
jsShorthand="lat"/>
            <apex:remoteObjectField name="Location__Longitude__s"
jsShorthand="long"/>
        </apex:remoteObjectModel>
    </apex:remoteObjects>


    <html>

        <head>
            <link rel="stylesheet"
href="{!URLFOR($Resource.leaflet1,'/leaflet.css')}" />
            <style>
                .map {
                    height: 480px;
                }

                .new-view {
                    background-color: #8B85F9;
                }

                .new-favorite {
                    background-color: #53B6D7;
                }

                .new-appointment {
                    background-color: #E260AB;
                }

                .right {
                    text-align: right;
                }

                .event-col {
                    width: 140px;
                }
            </style>
        </head>

        <body>

            <div id="app" class="slds-scope"></div>

            <script src="{!URLFOR($Resource.leaflet1,'/leaflet.js')}"></script>
            <script>

                function getSLDSPath() {
           return "{!URLFOR($Asset.SLDS)}";
                }

           function getRandomNumber(min, max) {
                    return Math.floor(Math.random() * (max - min + 1)) + min;
                }
                function getProperties(callback) {

                    var property = new SObjectModel.Property();
                    var properties;

                    property.retrieve({limit: 20}, function(error, records, event) {
                        if (error) {
                            alert(error.message);
                        } else {
                            properties = [];
                            console.log(records);
                            records.forEach(function(property) {
                                properties.push({
                                    id: property.get("Id"),
                                    address: property.get("address"),
                                    city: property.get("city"),
                                    price: property.get("price"),
                                    state: property.get("state"),
                                    lat: property.get("lat"),
                                    long: property.get("long"),
                                    view: getRandomNumber(100, 900),
                                    favorite: getRandomNumber(10, 60),
                                    appointment: getRandomNumber(0,8)
                                });
                            });
                            console.log(properties);
                            callback(properties);

                        }
                    });

                }

            </script>

            <script src="https://cdn.socket.io/socket.io-

            <script src="{!URLFOR($Resource.heatmap)}"></script>

                   </body>
               </html>
           </apex:page>
```

HeapMapMock.vfp

```
<apex:page applyBodyTag="false" applyHtmlTag="false" standardStylesheets="false" showHeader="false">

    <apex:slds />

    <apex:remoteObjects >
        <apex:remoteObjectModel name="Property__c" jsShorthand="Property">
            <apex:remoteObjectField name="Name" jsShorthand="address"/>
            <apex:remoteObjectField name="City__c" jsShorthand="city"/>
            <apex:remoteObjectField name="State__c" jsShorthand="state"/>
            <apex:remoteObjectField name="Price__c" jsShorthand="price"/>
            <apex:remoteObjectField name="Location__Latitude__s" jsShorthand="lat"/>
            <apex:remoteObjectField name="Location__Longitude__s" jsShorthand="long"/>
        </apex:remoteObjectModel>
    </apex:remoteObjects>


    <html>

        <head>

```

```
            <link rel="stylesheet" href="{!URLFOR($Resource.leaflet,'/leaflet.css')}" />
            <style>
                .map {
                    height: 480px;
                }

                .new-view {
                    background-color: #8B85F9;
                }

                .new-favorite {
                    background-color: #53B6D7;
                }

                .new-appointment {
                    background-color: #E260AB;
                }

                .right {
                    text-align: right;
                }

                .event-col {
                    width: 140px;
                }
            </style>
        </head>

        <body>

            <div id="app" class="slds-scope"></div>

            <script src="{!URLFOR($Resource.leaflet,'/leaflet.js')}"></script>
            <script>

                function getSLDSPath() {
         return "{!URLFOR($Asset.SLDS)}";
         }
```

```

                function getRandomNumber(min, max) {
                    return Math.floor(Math.random() * (max - min + 1)) + min;
                }

                function getProperties(callback) {

                    var property = new SObjectModel.Property();
                    var properties;

                    property.retrieve({limit: 20}, function(error, records, event) {
                        if (error) {
                            alert(error.message);
                        } else {
                            properties = [];
                            console.log(records);
                            records.forEach(function(property) {
                                properties.push({
                                    id: property.get("Id"),
                                    address: property.get("address"),
                                    city: property.get("city"),
                                    price: property.get("price"),
                                    state: property.get("state"),
                                    lat: property.get("lat"),
                                    long: property.get("long"),
                                    view: getRandomNumber(100, 900),
                                    favorite: getRandomNumber(10, 60),
                                    appointment: getRandomNumber(0,8)
                                });
                            });
                            console.log(properties);
                            callback(properties);

                        }
                    });

                }

            </script>

            <script
src="{!URLFOR($Resource.heatmapmock)}"></script>

                 </body>
             </html>
         </apex:page>
```

DisplayImage.vfp

```
<apex:page showHeader="false" sidebar="false" >
<apex:image
url="https://developer.salesforce.com/files/salesforce-developer-

</apex:page>
```

DisplayUserInfo.vfp

```
<apex:page >
    <apex:pageBlock title="User Status">
        <apex:pageBlockSection columns="1">
            {! $User.FirstName }
        </apex:pageBlockSection>
    </apex:pageBlock>
</apex:page>
```

ContactView.vfp

```
<apex:page standardController="Contact">
    <apex:pageBlock title="Contact Summary">
        <apex:pageBlockSection >
            First Name: {! Contact.FIrstName } <br/>
            Last Name: {! Contact.LastName } <br/>
            Owner Email: {! Contact.Owner.Email } <br/>
        </apex:pageBlockSection>
    </apex:pageBlock>
</apex:page>
```

OppView.vfp

```
<apex:page standardController="Opportunity" >
    <apex:outputField Value="{! Opportunity.Name}"/>
    <apex:outputField Value="{! Opportunity.Amount}"/>
    <apex:outputField Value="{! Opportunity.CloseDate}"/>
    <apex:outputField Value="{! Opportunity.Account.Name}"/>
</apex:page>
```

CreateContact.vfp

```
<apex:page standardController="Contact">
    <apex:form >
        <apex:pageBlock title="Edit Contact">
            <apex:pageBlockSection >
                <apex:inputField value="{! Contact.FirstName }"
/>
                <apex:inputField value="{! Contact.LastName }" />
                <apex:inputField value="{! Contact.Email }" />
            </apex:pageBlockSection>
<apex:pageBlockButtons >
                <apex:commandButton action="{! save }"
value="Save" />
            </apex:pageBlockButtons>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

AccountList.vfp

```
<apex:page standardController="Account"
recordSetVar="accounts">
    <apex:repeat var="a" value="{!accounts}">
        <li>
            <apex:outputLink value="/{!a.Id}">
                <apex.outputText value="{!a.Name}">
                </apex.outputText>
            </apex:outputLink>
        </li>
    </apex:repeat>
</apex:page>

```

ShowImage.vfp

```
<apex:page >
    <apex:image alt="eye" title="eye"
url="{!URLFOR($Resource.vfimagetest, 'cats/kitten1.jpg')}"/>
</apex:page>
```

NewCaseList.vfp

```
<apex:page controller="NewCaseListController" >
    <apex:repeat var="case" value="{!newCases}">
        <apex:outputLink value="/{!case.ID}">
            <apex:outputText value="{!case.CaseNumber}">
            </apex:outputText>
        </apex:outputLink>
    </apex:repeat>
</apex:page>
```

ContactForm.vfp

```
<apex:page standardController="Contact">
    <head>
      <meta charset="utf-8" />
      <meta name="viewport" content="width=device-width,

      <title>Quick Start: Visualforce</title>
      <!-- Import the Design System style sheet -->
<apex:slds />
 </head>
    <body>
        <apex:form >
      <apex:pageBlock title="New Contact">
        <!--Buttons -->
         <apex:pageBlockButtons >
            <apex:commandButton action="{!save}"
value="Save"/>
         </apex:pageBlockButtons>
         <!--Input form -->
         <apex:pageBlockSection columns="1">
         <apex:inputField value="{!Contact.Firstname}"/>
         <apex:inputField value="{!Contact.Lastname}"/>
         <apex:inputField value="{!Contact.Email}"/>
        </apex:pageBlockSection>
      </apex:pageBlock>
      </apex:form>
    </body>
</apex:page>
```