SALESFORCE APEX CODE

1. APEX TRIGGERS

Get Started with Apex Triggers

- AccountAddressTrigger.apxt

```
trigger AccountAddressTrigger on Account (before insert,before update)
    {
      List<Account> acclst=new List<Account>();
      for(account a:trigger.new)
      {
        if(a.Match_Billing_Address__c==true && a.BillingPostalCode!=null)
        {
          a.ShippingPostalCode=a.BillingPostalCode;
        }
      }
    }
```

Bulk Apex Triggers

- ClosedOpportunityTrigger.apx

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

      List <Task> todoList = new List <Task>()

     for (Opportunity opp :Trigger.new){

        if(Trigger.isInsert || Trigger.isUpdate) {

          if(opp.StageName == 'Closed Won') {

            todoList.add(new Task(Subject = 'Follow Up Test Task', WhatId = opp.Id));

          }

        }

      }

      if(todoList.size()>0) {
```

```
        insert todoList;
    }
}
```

2. APEX TESTING

Get Started With Apex Unittest

- 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
   @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.apxc

```
@isTest
public class TestVerifyDate {
   @isTest static void Test1() {
      Date d =
VerifyDate.CheckDates(Date.parse('05/17/2022'),Date.parse('05/21/2022'));
      System.assertEquals(Date.parse('05/21/2022'),d);
   }
   @isTest static void Test2() {
      Date d =
VerifyDate.CheckDates(Date.parse('05/17/2022'),Date.parse('06/21/2022'));
      System.assertEquals(Date.parse('06/21/2022'),d);
   }
}
```

TEST APEX TRIGGERS

- RestrictContactName.apxt

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

- TestRestrictContactByName.apx

```
CheckDates(Date.parse
@isTest
public class TestRestrictContactByName {
   @isTest
```

```
    public static void testContact(){
        Contact ct = new Contact();
        ct.LastName = 'INVALIDNAME' ;
        Database.SaveResult res = Database.insert(ct,false);
        System.assertEquals('the last name "INVALIDNAME" is not
allowed',res.getErrors()[0].getMessage());
    }
}
```

Create Test Data for Apex Tests

- RandomContactFactory.apxc

```
public class RandomContactFactory {
    public static List<Contact> generateRandomContacts(Integer numContactsToGenerate,
String FName) {
        List<Contact> contactList = new List<Contact>();
        for(Integer i=0;i<numContactsToGenerate;i++) {
            Contact c = new Contact(FirstName=FName + ' ' + i, LastName = 'Contact '+i);
            contactList.add(c);
            System.debug(c);
        }
        //insert contactList;
        System.debug(contactList.size());
        return contactList;
    }
}
```

3. ASYNCHRONUS APEX

Use Future Methods

- AccountProcessor.apxc

```
public class AccountProcessor
  {
    @future
    public static void countContacts(Set<id> setId)
    {
```

```
    List<Account> lstAccount = [select id,Number_of_Contacts__c , (select id from
contacts ) from account where id in :setId ];
    for( Account acc : lstAccount )
    {
       List<Contact> lstCont = acc.contacts ;

       acc.Number_of_Contacts__c = lstCont.size();
    }
    update lstAccount;
 }
}
```

- AccoutProcessorTest.apxc

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

      Contact cont = New Contact();
      cont.FirstName ='Bob';
      cont.LastName ='Masters';
      cont.AccountId = a.Id;
      Insert cont;

      set<Id> setAccId = new Set<ID>();
      setAccId.add(a.id);

      Test.startTest();
        AccountProcessor.countContacts(setAccId);
      Test.stopTest();

      Account ACC = [select Number_of_Contacts__c from Account where id = :a.id
LIMIT 1];
      System.assertEquals ( Integer.valueOf(ACC.Number_of_Contacts__c) ,1);
 }
```

```
}
```

Uses Batch Apex

- LeadProcessor.apxc

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

global Integer count = 0;


global Database.QueryLocator start(Database.BatchableContext bc){

    return Database.getQueryLocator('SELECT ID, LeadSource FROM Lead');

}


global void execute (Database.BatchableContext bc, List<Lead> L_list){

    List<lead> L_list_new = new List<lead>();


    for(lead L:L_list){

        L.leadsource = 'Dreamforce';

        L_list_new.add(L);

        count += 1;

    }

    update L_list_new;

}

global void finish(Database.BatchableContext bc){

    system.debug('count =' + count);

}
```

}

- LeadProcessorTest.apxc

```
@isTest
  public class LeadProcessorTest {

      @isTest
      public static void testit(){
         List<lead> L_list = new List<lead>();

         for(Integer i=0; i<200; i++){
            Lead L = new lead();
            L.LastName = 'name' + i;
            L.Company = 'Company';
            L.Status = 'Random Status';
            L_list.add(L);
         }
         insert L_list;

         Test.startTest();
         LeadProcessor lp = new LeadProcessor();
         Id batchId = Database.executeBatch(lp);
         Test.stopTest();
      }
  }
```

Control Processes with Queueable Apex

- AddPrimaryContact.apcx

```
public class AddPrimaryContact implements Queueable{

    private Contact con;
    private String state;

    public AddPrimaryContact(Contact con,String state){
       this.con = con;
       this.state = state;
    }
```

```
    public void execute(QueueableContext context){
      List<Account> accounts = [Select Id, Name, (Select FirstName, LastName, Id from contacts)
                    from Account where BillingState = :state Limit 200];
      List<Contact> primaryContacts = new List<Contact>();

      for(Account acc:accounts){
        contact c = con.clone();
        c.AccountId = acc.Id;
        primaryContacts.add(c);
      }
      if(primaryContacts.size() > 0){
        insert primaryContacts;
      }
    }
  }
```

- AddPrimaryContactTest.apcx

```
@isTest
  public class AddPrimaryContactTest {

    static testmethod void testQueueable(){
      List<Account> testAccounts = new List<Account>();
      for(Integer i=0; i<50;i++){
        testAccounts.add(new Account(Name='Account' +i,BillingState='CA'));
      }
      for(Integer j=0;j<50;j++){
        testAccounts.add(new Account(Name='Account'+j,BillingState='NY'));
      }
      insert testAccounts;

      Contact testContact = new Contact(FirstName = 'JOHN',LastName='Doe');
      insert testContact;

      AddPrimaryContact addit = new addPrimaryContact(testContact,'CA');

      Test.startTest();
      system.enqueueJob(addit);
      Test.stopTest();

      System.assertEquals(50, [Select count() from Contact where accountId in (Select Id from
  Account where BillingState='CA')]);
```

```
    }
  }
```

Schedule Jobs Using The Apex Schedule

- DailyLeadProcessor.apxc

```
global class DailyLeadProcessor implements Schedulable{
    global void execute(SchedulableContext ctx){
        List<Lead> lList = [Select Id, LeadSource from Lead where LeadSource = null];

        if(!lList.isEmpty()){
            for(Lead l:lList){
                l.LeadSource = 'Dreamforce';
            }
            update lList;
        }
    }
}
```

- DailyLeadProcessorTest.apxc

```
@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 4 ? 2023';

    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();
    }
}
```

4. APEX INTEGRATION SERVICES

Apex REST Callouts

- AnimalLocator.apxc

```
public class AnimalLocator
 {
    public static String getAnimalNameById(Integer id)
    {
       Http http = new Http();
       HttpRequest request = new HttpRequest();
       request.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/'+id)
       request.setMethod('GET');
       HttpResponse response = http.send(request);
       String strResp = '';
       system.debug('******response '+response.getStatusCode());
       system.debug('******response '+response.getBody());
       // If the request is successful, parse the JSON response.
       if (response.getStatusCode() == 200)
       {
          // Deserializes the JSON string into collections of primitive data types.
          Map<String, Object> results = (Map<String, Object>)
             JSON.deserializeUntyped(response.getBody());
          // Cast the values in the 'animals' key as a list
          Map<string,object> animals = (map<string,object>) results.get('animal');
          System.debug('Received the following animals:' + animals );
          strResp = string.valueof(animals.get('name'));
          System.debug('strResp >>>>>>' + strResp );
       }
       return strResp ;
    }
}
```

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

Apex SOAP Callouts

- ParkService.apxc

```
//Generated by wsdl2apex

  public class ParkService {
    public class byCountryResponse {
      public String[] return_x;
      private String[] return_x_type_info = new String[]{'return','http://parks.services/',null,'0','-
  1','false'};
      private String[] apex_schema_type_info = new
  String[]{'http://parks.services/','false','false'};
      private String[] field_order_type_info = new String[]{'return_x'};
    }
    public class byCountry {
      public String arg0;
      private String[] arg0_type_info = new
  String[]{'arg0','http://parks.services/',null,'0','1','false'};
```

```
        private String[] apex_schema_type_info = new
String[]{'http://parks.services/','false','false'};
        private String[] field_order_type_info = new String[]{'arg0'};
    }
    public class ParksImplPort {
        public String endpoint_x = 'https://th-apex-soap-service.herokuapp.com/service/parks';
        public Map<String,String> inputHttpHeaders_x;
        public Map<String,String> outputHttpHeaders_x;
        public String clientCertName_x;
        public String clientCert_x;
        public String clientCertPasswd_x;
        public Integer timeout_x;
        private String[] ns_map_type_info = new String[]{'http://parks.services/', 'ParkService'};
        public String[] byCountry(String arg0) {
            ParkService.byCountry request_x = new ParkService.byCountry();
            request_x.arg0 = arg0;
            ParkService.byCountryResponse response_x;
            Map<String, ParkService.byCountryResponse> response_map_x = new Map<String,
ParkService.byCountryResponse>();
            response_map_x.put('response_x', response_x);
            WebServiceCallout.invoke(
              this,
              request_x,
              response_map_x,
              new String[]{endpoint_x,
              '',
              'http://parks.services/',
              'byCountry',
              'http://parks.services/',
              'byCountryResponse',
              'ParkService.byCountryResponse'}
            );
            response_x = response_map_x.get('response_x');
            return response_x.return_x;
        }
    }
}
```

ParkLocator.apxc

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

ParkLocatorTest.apxc

```
@isTest
  private class ParkLocatorTest {
     @isTest
     static void testParkLocator() {
        Test.setMock(WebServiceMock.class, new ParkServiceMock());
        String[] arrayOfParks = ParkLocator.country('India');

        System.assertEquals('Park1', arrayOfParks[0]);
  }
  }
```

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) {
        ParkService.byCountryResponse response_x = new ParkService.byCountryResponse();
        List<String> lstOfDummyParks = new List<String> {'Park1','Park2','Park3'};
        response_x.return_x = lstOfDummyParks;

        response.put('response_x', response_x);
```

```
    }
}
```

- AsyncParkServices.apxc

```
//Generated by wsdl2apex

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-service.herokuapp.com/service/parks';
        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'}
            );
        }
    }
}
```

Apex Web Services

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;
   }
}
```

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;
    }
}
```