Salesforce Developer Catalyst Programs

submitted By : Amar Shree Bhardwaj

CLASS: Discount On Property

```
Public class DiscountOnProperty{
    public static void Discount5 ( list<Realtor_c> VarPropertyList ){for(
        Realtor_c VarP : VarPropertyList){
            if(VarP.Type_c == 'Row House'){
              VarP.Price_c = VarP.Price__c - (VarP.Price__c * 0.05 ) ;
                }
            else if(VarP.Type_c == 'Villa')
                {
              VarP.Price_c = VarP.Price__c - (VarP.Price__c * 0.1 ) ;
            }
             else if(VarP.Type_c == 'Apartment')
                {
              VarP.Price_c = VarP.Price__c - (VarP.Price__c * 0.2 );


        }
    }
    }
}
```

TRIGGER: Discount On Property

```
trigger DiscountTrigger on Realtor_c (before insert , Before Update) {
    DiscountOnProperty.Discount5(Trigger.New);
}
```

TEST CLASS: Discount On Property

```
@isTest
Class DiscountOnPropertyTest {

 //define test method
 @IsTest
  static Void DiscountTest1(){

  //take input by DML Code
  Realtor_c VarData = new Realtor_c();
  VarData.Type_c = 'Row House' ;
  VarData.Price_c = 70000;

  Insert VarData;
```

```
Realtor_c VarFB ;
VarFB = [SELECT Price_c FROM Realtor__c WHERE Id =: VarData.Id];

system.assertEquals (66500, VarFB.Price_c);
}

@IsTest
static Void DiscountTest2 (){

Realtor_c VarData = new Realtor_c();
VarData.Type_c = 'Villa' ;
VarData.Price_c = 80000;

Insert VarData;

Realtor_c VarFB ;
VarFB = [SELECT Price_c FROM Realtor__c WHERE Id =: VarData.Id];

system.assertEquals (72000, VarFB.Price_c);
}
```

```
@IsTest
   static Void DiscountTest3 (){

   Realtor_c VarData = new Realtor_c();
   VarData.Type_c = 'Apartment' ;
   VarData.Price_c = 90000;

   Insert VarData;

   Realtor__c VarFB ;
   VarFB = [SELECT Price_c FROM Realtor__c WHERE Id =: VarData.Id];

   system.assertEquals (72000, VarFB.Price_c);
   }
   }
```

BULKIFIED TRIGER:

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

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

    for(Opportunity Opp:Trigger.New){
        if(Trigger.isInsert || Trigger.isUpdate)
        if(opp.StageName=='Closed Won')
            taskList.add(new task(Subject='Follow Up Test Task',
                        WhatId=opp.Id));
    }

    if(taskList.size()>0)
        insert taskList;

}
```

Create A Unit Test For A Simple Apex Class Module 1

```
@isTest
private class TestVerifyDate {
   static testMethod void TestVerifyDate() {
     VerifyDate.CheckDates(System.today(),System.today().addDays(10));
     VerifyDate.CheckDates(System.today(),System.today().addDays(78));
   }
}



/ FOR CLASS


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; }
    }

    //method to return the end of the month of a given date
    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;
    }

}
```

Create A Unit Test For A Simple Apex Trigger Module 2 In Testing

```
@isTest
private class TestRestrictContactByName {


   static testMethod void  metodoTest()
   {


      List<Contact> listContact= new List<Contact>();
      Contact c1 = new Contact(FirstName='Francesco', LastName='Riggio' ,
email='Test@test.com');
      Contact c2 = new Contact(FirstName='Francesco1', LastName =
'INVALIDNAME',email='Test@test.com');
      listContact.add(c1);
      listContact.add(c2);


      Test.startTest();
         try
         {
            insert  listContact;
         }
         catch(Exception ee)
         {
         }
```

```
        Test.stopTest();

    }

}



//trigger from github

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');
                }

        }



}
```

Create Test Data Module 3 In Testing

```
public with sharing class RandomContactFactory
{
    public static List<Contact> generateRandomContacts( Integer noOfContacts,
String lastName )
    {
        List<Contact> contacts = new List<Contact>();

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

        return contacts;
    }
}
```

Apex Class That Calls A REST Endpoint And Test Class Module 1 Integration

```
//ANIMAL LOCATOR CLASS
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 ;
    }

}


//ANIMAL LOCATOR MOCK TEST 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.setStatusCode(200);
        return response;
    }
}


/ ANIMAL LOCATOR TEST CLASS
 @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 Class Using WSDL2Apex And Test Class Module 2 In Integration

```
/ 1/4 Generated by WSDL2APEX apex class


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;
        }
    }}
```

```
/ 2/4 Test Class For Park Service Class


@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);
   }
}
```

```
/ 3/4 Parklocator Class

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

/  4/4 PARKLOCATOR TEST CLASS
@isTest
private  class  ParkLocatorTest{
    @isTest
    static void testParkLocator() {
        Test.setMock(WebServiceMock.class, new ParkServiceMock());
        String[] arrayOfParks = ParkLocator.country('India');

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

Apex Class That Calls A REST Endpoint And Test Class Module 2 Integration

```
//ANIMAL LOCATOR CLASS


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 ;
    }

}

//Animal Locator Mock Test 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.setStatusCode(200);
        return response;
    }
}


/ Animal Locator Test Class

@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 Class Using WSDL2Apex And Test Class Module 3 In Integration

```
/ 1/4 Generated by WSDL2APEX apex class


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;
        }
    }
}
```

/ 2/4 TEST CLASS FOR PARK SERVICE CLASS

```
@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);
   }
}
```

/ 3/4 PARKLOCATOR CLASS

```
public class ParkLocator {
```

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

/  4/4 PARKLOCATOR TEST CLASS
@isTest
private  class  ParkLocatorTest{
    @isTest
    static void testParkLocator() {
        Test.setMock(WebServiceMock.class, new ParkServiceMock());
        String[] arrayOfParks = ParkLocator.country('India');

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

Apex REST Service That Returns An Account And Its Contacts Module 4 In Integration

/ apex class for web service

```
@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;
   }
}
```

/ test class

```
@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);
    }
}
```

Apex Class That Uses The @Future Annotation To Update Account Records. Module 2 In Batch Apex

```
/  Apex class @FUTURE ANNOTATION


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;
  }
}


//TEST CLASS


@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);
    }

}
```

Apex class that uses the @future annotation to update Account records. MODULE 2 IN ASYNCHRONOUS APEX

```
/ Apex class @FUTURE ANNOTATION


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;
  }
}


//TEST CLASS


@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);
    }

}
```

Apex Class That Uses Batch Apex To Update Lead Records. Asynchronous Apex Module 2

```
// Apex class


 @FUTURE ANNOTATION
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;

  }

}


//TEST CLASS


@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);
    }

}
```

Apex Class That Uses Batch Apex To Update Lead Records. Asynchronous Apex Module 3

```
// Apex class


@FUTURE ANNOTATION
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;
  }
}


//TEST CLASS


@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);
    }

}
```

Queueable Apex Class That Inserts Contacts For Accounts. Async Apex Module 4

```
// Apex class

@FUTURE ANNOTATION
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;

  }

}


//TEST CLASS


@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);
    }

}
```

Apex Class That Uses Scheduled Apex To Update Lead Records. Asynch Apex Module 5

```
// Apex class

@FUTURE ANNOTATION
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;
  }
}


//TEST CLASS


@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);
    }

}
```

VISUALFORCE PAGES:

VFP Oppview

```
<apex:page  standardController="Opportunity">
   <apex:pageBlock >
   <apex:pageBlockSection  title="Opportunity  info"  columns="1">

      <apex:outputField  Value="{!  Opportunity.Account.name  }"  />
       <apex:outputField Value="{!  Opportunity.amount  }"  />
       <apex:outputField  Value="{!  Opportunity.CloseDate}"  />
       <apex:outputField Value="{!  Opportunity.name}"  />

      </apex:pageBlockSection>
   </apex:pageBlock>
</apex:page>
```

VFP List Of Account

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

vfp display image

```
<apex:page  showHeader="false"  title="DisplayImage"  sidebar="false">
   <apex:form >
      <table>
         <tr>
            <td  width="1000px"  height="600px;"  align="center">
               <apex:image url="https://developer.salesforce.com/files/salesforce-
developer-network-logo.png" />
            </td>
         </tr>
      </table>
   </apex:form>
</apex:page>
```

VFP View Contact

```
<apex:page  standardController="Contact">
   <apex:form>
   <apex:pageBlock title="Contacts Modifyer Page" >

      <apex:pageBlockSection  title="demo  title"  columns="1">

         <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>
```

VFP Display Image Static Resource

```
<apex:page showHeader="false" title="DisplayImage" sidebar="false">
   <apex:form >
      <table>
         <tr>
            <td width="1000px" height="600px;" align="center">
               <apex:image url="{!URLFOR($Resource.vfimagetest, 'cats/kitten1.jpg')}" />
            </td>
         </tr>
      </table>
   </apex:form>
</apex:page>
```

VFP With Custom Controller

```
//vfp with custom controllr

<apex:page controller="NewCaseListController">
  <apex:repeat value="{!NewCases}" var="Case">

<apex:outputLink   onclick="/?id={!Case.Id}">
  {!Case.CaseNumber}
  </apex:outputLink>
  </apex:repeat>

</apex:page>



//Controller class

public class NewCaseListController {

list<case> newcase = new list<case>();
    public list<case> GetNewCases()
    {
    newcase = [Select Id,CaseNumber from case where status='New'];

        return newcase;
    }
}
```

APEX SPECIALIST CHALLENGE 1

```
//CHALLENGE 1 APEX SPECIALIST

public with sharing class MaintenanceRequestHelper {

    public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case>
nonUpdCaseMap) {

        Set<Id> validIds = new Set<Id>();



        For (Case c : updWorkOrders){

            if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){

                if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){

                    validIds.add(c.Id);



                }

            }

        }


        if (!validIds.isEmpty()){

            List<Case> newCases = new List<Case>();

            Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id, Vehicle_c,
Equipment_c, Equipment_r.Maintenance_Cycle_c,(SELECT
Id,Equipment__c,Quantity__c FROM Equipment_Maintenance_Items__r)

                                                FROM Case WHERE Id IN :validIds]);

            Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
```

```
        AggregateResult[] results = [SELECT Maintenance_Request_c,
MIN(Equipment_r.Maintenance_Cycle_c)cycle FROM
Equipment_Maintenance_Item_c WHERE Maintenance_Request_c IN :ValidIds GROUP
BY Maintenance_Request_c];


      for (AggregateResult ar : results){
        maintenanceCycles.put((Id) ar.get('Maintenance_Request_c'), (Decimal)
ar.get('cycle'));
      }


        for(Case cc : closedCasesM.values()){
          Case nc = new Case (
            ParentId = cc.Id,
          Status = 'New',
            Subject = 'Routine Maintenance',
            Type = 'Routine Maintenance',
            Vehicle_c = cc.Vehicle_c,
            Equipment_c =cc.Equipment_c,
            Origin = 'Web',
            Date_Reported_c = Date.Today()


          );


          If (maintenanceCycles.containskey(cc.Id)){
            nc.Date_Due_c = Date.today().addDays((Integer)
maintenanceCycles.get(cc.Id));
          } else {
            nc.Date_Due_c = Date.today().addDays((Integer)
cc.Equipment_r.maintenance_Cycle_c);
```

```
            }

            newCases.add(nc);
        }

        insert newCases;

        List<Equipment_Maintenance_Item_c> clonedWPs = new
List<Equipment_Maintenance_Item_c>();
        for (Case nc : newCases){
            for (Equipment_Maintenance_Item_c wp :
closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items_r){
                Equipment_Maintenance_Item_c wpClone = wp.clone();
                wpClone.Maintenance_Request_c = nc.Id;
                ClonedWPs.add(wpClone);

            }
        }
        insert ClonedWPs;
    }
  }
}

 //TRIGGER

 trigger MaintenanceRequest on Case (before update, after update) {
```

```
    if(Trigger.isUpdate && Trigger.isAfter){

        MaintenanceRequestHelper.updateWorkOrders(Trigger.New, Trigger.OldMap);

    }

}
```

APEX SPECIALIST CHALLENGE 2

```
public with sharing class WarehouseCalloutService implements Queueable {
    private static final String WAREHOUSE_URL = 'https://th-superbadge-
apex.herokuapp.com/equipment';



    @future(callout=true)
    public static void runVarwarehouseEquipmentSync(){
        Http http = new Http();
        HttpRequest request = new HttpRequest();


        request.setEndpoint(WAREHOUSE_URL);
        request.setMethod('GET');
        HttpResponse response = http.send(request);


        List<Product2> VarwarehouseEq = new List<Product2>();


        if (response.getStatusCode() == 200){
            List<Object> jsonResponse =
(List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());


            for (Object eq : jsonResponse){
                Map<String,Object> mapJson = (Map<String,Object>)eq;
                Product2 myEq = new Product2();
                myEq.Replacement_Part_c = (Boolean) mapJson.get('replacement');
                myEq.Name = (String) mapJson.get('name');
```

```
            myEq.Maintenance_Cycle_c = (Integer) mapJson.get('maintenanceperiod');
            myEq.Lifespan_Months_c = (Integer) mapJson.get('lifespan');
            myEq.Cost_c = (Integer) mapJson.get('cost');
            myEq.Warehouse_SKU_c = (String) mapJson.get('sku');
            myEq.Current_Inventory_c = (Double) mapJson.get('quantity');
            myEq.ProductCode = (String) mapJson.get('_id');
            VarwarehouseEq.add(myEq);
        }

        if (VarwarehouseEq.size() > 0){
            upsert VarwarehouseEq;
            System.debug('Your equipment was synced with the warehouse one');
        }
    }
}

public static void execute (QueueableContext context){
    runVarwarehouseEquipmentSync();
}

}
```

APEX SPECIALIST CHALLENGE 3

```
global with sharing class WarehouseSyncSchedule implements Schedulable{
    global void execute(SchedulableContext ctx){
        System.enqueueJob(new WarehouseCalloutService());
    }
}
```

APEX SPECIALIST CHALLENGE 4

```
@istest
public with sharing class MaintenanceRequestHelperTest {

    private static final string STATUS_NEW = 'New';
    private static final string WORKING = 'Working';
    private static final string CLOSED = 'Closed';
    private static final string REPAIR = 'Repair';
    private static final string REQUEST_ORIGIN = 'Web';
    private static final string REQUEST_TYPE = 'Routine Maintenance';
    private static final string REQUEST_SUBJECT = 'Testing subject';

    PRIVATE STATIC Vehicle_c createVehicle(){
        Vehicle_c Vehicle = new Vehicle_C(name = 'SuperTruck');
        return Vehicle;
    }

    PRIVATE STATIC Product2 createEq(){
        product2 equipment = new product2(name = 'SuperEquipment',
                                          lifespan_months_C = 10,
                                          maintenance_cycle_C = 10,
                                          replacement_part_c = true);
        return equipment;
    }

    PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id equipmentId){
```

```
    case cs = new case(Type=REPAIR,
                       Status=STATUS_NEW,
                       Origin=REQUEST_ORIGIN,
                       Subject=REQUEST_SUBJECT,
                       Equipment_c=equipmentId,
                       Vehicle_c=vehicleId);
    return cs;
  }

  PRIVATE STATIC Equipment_Maintenance_Item_c createWorkPart(id equipmentId,id
requestId){
    Equipment_Maintenance_Item_c wp = new
Equipment_Maintenance_Item_c(Equipment_c = equipmentId,
                                                Maintenance_Request_c = requestId);
    return wp;
  }


  @istest
  private static void testMaintenanceRequestPositive(){
    Vehicle_c vehicle = createVehicle();
    insert vehicle;
    id vehicleId = vehicle.Id;

    Product2 equipment = createEq();
    insert equipment;
    id equipmentId = equipment.Id;
```

```
case somethingToUpdate = createMaintenanceRequest(vehicleId,equipmentId);insert
somethingToUpdate;

Equipment_Maintenance_Item_c workP =
createWorkPart(equipmentId,somethingToUpdate.id);
insert workP;

test.startTest();
somethingToUpdate.status = CLOSED;
update somethingToUpdate;
test.stopTest();

Case newReq = [Select id, subject, type, Equipment__c, Date_Reported__c,
Vehicle__c, Date_Due__c
        from case
        where status =:STATUS_NEW];

Equipment_Maintenance_Item__c workPart = [select id
                    from Equipment_Maintenance_Item__c
                    where Maintenance_Request_c =:newReq.Id];

system.assert(workPart != null); system.assert(newReq.Subject
!= null); system.assertEquals(newReq.Type, REQUEST_TYPE);
SYSTEM.assertEquals(newReq.Equipment_c, equipmentId);
SYSTEM.assertEquals(newReq.Vehicle_c, vehicleId);
```

```
        SYSTEM.assertEquals(newReq.Date_Reported_c, system.today());
    }

    @istest
    private static void testMaintenanceRequestNegative(){
        Vehicle_C vehicle = createVehicle();
        insert vehicle;
        id vehicleId = vehicle.Id;

        product2 equipment = createEq();
        insert equipment;
        id equipmentId = equipment.Id;

        case emptyReq = createMaintenanceRequest(vehicleId,equipmentId);insert
        emptyReq;

        Equipment_Maintenance_Item_c workP = createWorkPart(equipmentId,
emptyReq.Id);
        insert workP;

        test.startTest();
        emptyReq.Status = WORKING;
        update emptyReq;
        test.stopTest();

        list<case> allRequest = [select id
                                 from case];
```

```
        Equipment_Maintenance_Item_c workPart = [select id
                                                 from Equipment_Maintenance_Item_c
                                                 where Maintenance_Request_c = :emptyReq.Id];

        system.assert(workPart != null);
        system.assert(allRequest.size() == 1);
    }

    @istest
    private static void testMaintenanceRequestBulk(){
        list<Vehicle_C> vehicleList = new list<Vehicle_C>();
        list<Product2>  equipmentList  =  new  list<Product2>();
        list<Equipment_Maintenance_Item_c> workPartList = new
list<Equipment_Maintenance_Item_c>();
        list<case> requestList = new list<case>();
        list<id> oldRequestIds = new list<id>();

        for(integer i = 0; i < 300; i++){
          vehicleList.add(createVehicle());
          equipmentList.add(createEq());
        }
        insert vehicleList;
        insert  equipmentList;

        for(integer i = 0; i < 300; i++){
           requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
```

```
equipmentList.get(i).id));
        }
        insert requestList;

        for(integer i = 0; i < 300; i++){
            workPartList.add(createWorkPart(equipmentList.get(i).id, requestList.get(i).id));
        }
        insert  workPartList;

        test.startTest();
        for(case req : requestList){
            req.Status = CLOSED;
            oldRequestIds.add(req.Id);
        }
        update requestList;
        test.stopTest();

        list<case> allRequests = [select id
                                  from case
                                  where status =: STATUS_NEW];

        list<Equipment_Maintenance_Item_c> workParts = [select id
                                                        from Equipment_Maintenance_Item_c
                                                        where Maintenance_Request_c in: oldRequestIds];

        system.assert(allRequests.size()  ==  300);
    }}
```

APEX SPECIALIST CHALLENGE 5

```
//MOCK TEST
@isTest
global class WarehouseCalloutServiceMock implements HttpCalloutMock {
    global static HttpResponse respond(HttpRequest request){

        System.assertEquals('https://th-superbadge-apex.herokuapp.com/equipment',
request.getEndpoint());
        System.assertEquals('GET',   request.getMethod());

        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5
,"name":"Generator 1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"}]');
        response.setStatusCode(200);
        return response;
    }
}

//UNIT TEST FOR CLASS


@isTest

private class WarehouseCalloutServiceTest {
```

```
    @isTest
    static void testWareHouseCallout(){
        Test.startTest();
        Test.setMock(HTTPCalloutMock.class, new WarehouseCalloutServiceMock());
        WarehouseCalloutService.runVarwarehouseEquipmentSync();
        Test.stopTest();
        System.assertEquals(1, [SELECT count() FROM Product2]);
    }
}
```

APEX SPECIALIST CHALLENGE 6

```
@isTest
public class WarehouseSyncScheduleTest {


    @isTest static void WarehousescheduleTest(){
        String scheduleTime = '00 00 01 * * ?';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
        String jobID=System.schedule('Warehouse Time To Schedule to Test',
scheduleTime, new WarehouseSyncSchedule());
        Test.stopTest();
        //Contains schedule information for a scheduled job. CronTrigger is similar to a
cron job on UNIX systems.
        /  This object is available in API version 17.0 and later.
        CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime > today];
        System.assertEquals(jobID, a.Id,'Schedule ');



    }
}
```