APEX SPECIALIST SUPERBADGE

AUTOMATE RECORD CREATION:

MaintenanceRequest.apxt

```
@isTest
private class MaintenanceRequestTest{
    @testSetup
    static void setup(){
        //Equipment SETUP
        List<Product2> lstOfEqpmnts = new List<Product2>();

        Product2 eqip = new Product2(Name = 'Test Equipment',
                                     Maintenance_Cycle__c = 10,
                                     Cost__c = 100,
                                     Current_Inventory__c = 10,
                                     Lifespan_Months__c = 10,
                                     Replacement_Part__c = true,
                                     Warehouse_SKU__c = 'abc');
        lstOfEqpmnts.add(eqip);
        INSERT lstOfEqpmnts;
    }

    @isTest
    static void testMaintenanceRequest(){
        List<Case> lstOfInsertMRs = new List<Case>();

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

        Id equipId = [SELECT Id FROM Product2 LIMIT 1].get(0).Id;

        Case newCase = new Case(Type = 'Routine Maintenance',Status =
'New', Origin = 'Phone');
        newCase.Equipment__c = equipId;
        lstOfInsertMRs.add(newCase);

       Case mrInsert = new Case(Type = 'Routine Maintenance2', Status =
'New', Origin = 'Phone');
       mrInsert.Equipment__c = equipId;
       lstOfInsertMRs.add(mrInsert);

```

```
35
36          Test.startTest();
37              INSERT lstOfInsertMRs;
38
39              System.assertEquals(2, lstOfInsertMRs.size());
40
41              for(Case mrUpdt : lstOfInsertMRs){
42                  mrUpdt.Status = 'Closed';
43                  lstOfUpdtMRs.add(mrUpdt);
44              }
45
46              UPDATE lstOfUpdtMRs;
47
48            System.assertEquals(2, lstOfUpdtMRs.size());
49          Test.stopTest();
50      }
51  }
```

MaintenanceRequestHelper.apxc

```
1   public with sharing class MaintenanceRequestHelper {
2       public static void updateworkOrders(List<Case> updWorkOrders,
    Map<Id,Case> nonUpdCaseMap) {
3           Set<Id> validIds = new Set<Id>();
4
5
6           For (Case c : updWorkOrders){
7               if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status
    == 'Closed'){
8                   if (c.Type == 'Repair' || c.Type == 'Routine

9                       validIds.add(c.Id);
10
11
12                  }
13              }
14          }
15
16          if (!validIds.isEmpty()){
17              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;
        }
    }
}
```

SYNCHRONIZATION SALESFORCE DATA WITH AN EXTERNAL SYSTEM:

WarehouseCalloutService.apxc

```
public with sharing class WarehouseCalloutService {

    private static final String endpoint = 'https://th-superbadge-


    @future(callout = true)     //need this so that it knows to call to
external source
    public static void runWarehouseEquipmentSync(){
        Http http = new Http();
        HttpRequest httpRequest = new HttpRequest();
        httpRequest.setEndpoint(endpoint);
        httpRequest.setMethod('GET');
        HttpResponse httpResponse = http.send(httpRequest);

        //if successfully get the JSON file, need to parse out to
different equipment objects
        if (httpResponse.getStatusCode() == 200){ //status = "OK" (this
is for GET or HEAD requests)
            List<Object> equipmentList = (List<Object>)
JSON.deserializeUntyped(httpResponse.getBody());
            List<Product2> products = new List<Product2>();

            for(Object item: equipmentList){
```

```
                Map<String, Object> productMap = (Map<String,Object>)
item;      //map of item(s) in JSON file
                Product2 product = new Product2();      //list of
products to insert/update in system

                product.Replacement_Part__c = (Boolean)
productMap.get('replacement');
                product.Cost__c = (Integer) productMap.get('cost');
                product.Current_Inventory__c = (Integer)
productMap.get('quantity');
                product.Lifespan_Months__c = (Integer)
productMap.get('lifespan');
                product.Maintenance_Cycle__c = (Integer)
productMap.get('maintenanceperiod');
                product.Warehouse_SKU__c = (String)
productMap.get('sku');
                product.Name = (String) productMap.get('name');
                product.ProductCode = (String) productMap.get('_id');
                products.add(product);
            }

            if(products.size() > 0){  //only need to upsert if items
actually exist
                System.debug(products);
                upsert products;
            }
        }
    }
}
```

SCHEDULE SYNCHRONIZATION USING APEX CODE:

WarehouseSyncSchedule.apxc

```
global class WarehouseSyncSchedule implements Schedulable
{
    global void execute ( SchedulableContext sc )
    {
        WarehouseCalloutService.runWarehouseEquipmentSync();
    }
}
```

TEST AUTOMATION LOGIC:

MaintenanceRequestHelperTest.apxc

```
@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;
```

```
33      }
34
35      PRIVATE STATIC Equipment_Maintenance_Item__c createWorkPart(id
    equipmentId,id requestId){
36          Equipment_Maintenance_Item__c wp = new
    Equipment_Maintenance_Item__c(Equipment__c = equipmentId,
37
    Maintenance_Request__c = requestId);
38          return wp;
39      }
40
41
42      @istest
43      private static void testMaintenanceRequestPositive(){
44          Vehicle__c vehicle = createVehicle();
45          insert vehicle;
46          id vehicleId = vehicle.Id;
47
48          Product2 equipment = createEq();
49          insert equipment;
50          id equipmentId = equipment.Id;
51
52          case somethingToUpdate =
    createMaintenanceRequest(vehicleId,equipmentId);
53          insert somethingToUpdate;
54
55          Equipment_Maintenance_Item__c workP =
    createWorkPart(equipmentId,somethingToUpdate.id);
56          insert workP;
57
58          test.startTest();
59          somethingToUpdate.status = CLOSED;
60          update somethingToUpdate;
61          test.stopTest();
62
63          Case newReq = [Select id, subject, type, Equipment__c,
    Date_Reported__c, Vehicle__c, Date_Due__c
64                         from case
65                         where status =:STATUS_NEW];
66
67          Equipment_Maintenance_Item__c workPart = [select id
68                                                    from
    Equipment_Maintenance_Item__c
69                                                    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);
    }
```

```
110
111     @istest
112     private static void testMaintenanceRequestBulk(){
113         list<Vehicle__C> vehicleList = new list<Vehicle__C>();
114         list<Product2> equipmentList = new list<Product2>();
115         list<Equipment_Maintenance_Item__c> workPartList = new
    list<Equipment_Maintenance_Item__c>();
116         list<case> requestList = new list<case>();
117         list<id> oldRequestIds = new list<id>();
118 
119         for(integer i = 0; i < 300; i++){
120            vehicleList.add(createVehicle());
121             equipmentList.add(createEq());
122         }
123         insert vehicleList;
124         insert equipmentList;
125 
126         for(integer i = 0; i < 300; i++){
127 
    requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
    equipmentList.get(i).id));
128         }
129         insert requestList;
130 
131         for(integer i = 0; i < 300; i++){
132             workPartList.add(createWorkPart(equipmentList.get(i).id,
    requestList.get(i).id));
133         }
134         insert workPartList;
135 
136         test.startTest();
137         for(case req : requestList){
138             req.Status = CLOSED;
139             oldRequestIds.add(req.Id);
140         }
141         update requestList;
142         test.stopTest();
143 
144         list<case> allRequests = [select id
145                                   from case
146                                   where status =: STATUS_NEW];
147 
148         list<Equipment_Maintenance_Item__c> workParts = [select id
149                                                          from
```

```
    Equipment_Maintenance_Item__c
150                                                         where
    Maintenance_Request__c in: oldRequestIds];
151
152         system.assert(allRequests.size() == 300);
153     }
154}
```

MaintenanceRequestHelper.apxc

```
1   public with sharing class MaintenanceRequestHelper {
2       public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case>
    nonUpdCaseMap) {
3           Set<Id> validIds = new Set<Id>();
4
5
6           For (Case c : updWorkOrders){
7               if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){
8                   if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
9                       validIds.add(c.Id);
10
11
12                  }
13              }
14          }
15
16          if (!validIds.isEmpty()){
17              List<Case> newCases = new List<Case>();
18              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)
19                                                  FROM Case WHERE Id IN :validIds]);
20              Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
21              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];
22
23          for (AggregateResult ar : results){
24              maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
25          }
```

```
26
27          for(Case cc : closedCasesM.values()){
28              Case nc = new Case (
29                  ParentId = cc.Id,
30              Status = 'New',
31                  Subject = 'Routine Maintenance',
32                  Type = 'Routine Maintenance',
33                  Vehicle__c = cc.Vehicle__c,
34                  Equipment__c =cc.Equipment__c,
35                  Origin = 'Web',
36                  Date_Reported__c = Date.Today()
37
38              );
39
40              If (maintenanceCycles.containskey(cc.Id)){
41                  nc.Date_Due__c = Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
42              }
43
44              newCases.add(nc);
45          }
46
47          insert newCases;
48
49          List<Equipment_Maintenance_Item__c> clonedWPs = new
    List<Equipment_Maintenance_Item__c>();
50          for (Case nc : newCases){
51              for (Equipment_Maintenance_Item__c wp :
    closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
52                  Equipment_Maintenance_Item__c wpClone = wp.clone();
53                  wpClone.Maintenance_Request__c = nc.Id;
54                  ClonedWPs.add(wpClone);
55
56              }
57          }
58          insert ClonedWPs;
59      }
60    }
61 }
```

MaintenanceRequest.apxt

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

```
    if(Trigger.isUpdate && Trigger.isAfter){
        MaintenanceRequestHelper.updateWorkOrders(Trigger.New, Trigger.OldMap);
    }
}
```

TEST CALLOUT LOGIC:

WarehouseCalloutService.apxc

```
public with sharing class WarehouseCalloutService {

    private static final String endpoint = 'https://th-superbadge-


    @future(callout = true)     //need this so that it knows to call to external source
    public static void runWarehouseEquipmentSync(){
        Http http = new Http();
        HttpRequest httpRequest = new HttpRequest();
        httpRequest.setEndpoint(endpoint);
        httpRequest.setMethod('GET');
        HttpResponse httpResponse = http.send(httpRequest);

        //if successfully get the JSON file, need to parse out to different equipment objects
        if (httpResponse.getStatusCode() == 200){ //status = "OK" (this is for GET or HEAD requests)
            List<Object> equipmentList = (List<Object>) JSON.deserializeUntyped(httpResponse.getBody());
            List<Product2> products = new List<Product2>();

            for(Object item: equipmentList){
                Map<String, Object> productMap = (Map<String,Object>) item;      //map of item(s) in JSON file
                Product2 product = new Product2();      //list of products to insert/update in system

                product.Replacement_Part__c = (Boolean) productMap.get('replacement');
```

```
                product.Cost__c = (Integer) productMap.get('cost');
                product.Current_Inventory__c = (Integer)
productMap.get('quantity');
                product.Lifespan_Months__c = (Integer)
productMap.get('lifespan');
                product.Maintenance_Cycle__c = (Integer)
productMap.get('maintenanceperiod');
                product.Warehouse_SKU__c = (String)
productMap.get('sku');
                product.Name = (String) productMap.get('name');
                product.ProductCode = (String) productMap.get('_id');
                products.add(product);
            }

            if(products.size() > 0){  //only need to upsert if items
actually exist
                System.debug(products);
                upsert products;
            }
        }
    }
}
```

WarehouseCalloutServiceTest.apxc

```
public with sharing class WarehouseCalloutService {

    private static final String WAREHOUSE_URL = 'https://th-superbadge-


    //@future(callout=true)
    public static void runWarehouseEquipmentSync(){

        Http http = new Http();
        HttpRequest request = new HttpRequest();

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

```

```
15
16          List<Product2> warehouseEq = new List<Product2>();
17
18          if (response.getStatusCode() == 200){
19              List<Object> jsonResponse =
    (List<Object>)JSON.deserializeUntyped(response.getBody());
20              System.debug(response.getBody());
21
22              for (Object eq : jsonResponse){
23                  Map<String,Object> mapJson = (Map<String,Object>)eq;
24                  Product2 myEq = new Product2();
25                  myEq.Replacement_Part__c = (Boolean)
    mapJson.get('replacement');
26                  myEq.Name = (String) mapJson.get('name');
27                  myEq.Maintenance_Cycle__c = (Integer)
    mapJson.get('maintenanceperiod');
28                  myEq.Lifespan_Months__c = (Integer)
    mapJson.get('lifespan');
29                  myEq.Cost__c = (Decimal) mapJson.get('lifespan');
30                  myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
31                  myEq.Current_Inventory__c = (Double)
    mapJson.get('quantity');
32                  warehouseEq.add(myEq);
33              }
34
35              if (warehouseEq.size() > 0){
36                  upsert warehouseEq;
37                  System.debug('Your equipment was synced with the

38                  System.debug(warehouseEq);
39              }
40
41          }
42      }
43  }
44
45
```

WarehouseCalloutServiceMock.apxc

```
@isTest
global class WarehouseCalloutServiceMock implements
HttpCalloutMock {
    // implement http mock callout
    global static HttpResponse respond(HttpRequest request){

        System.assertEquals('https://th-superbadge-
));
        System.assertEquals('GET', request.getMethod());

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

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement




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

TEST SCHEDULING LOGIC:

WarehouseSyncSchedule.apxc

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

WarehouseSyncScheduleTest.apxc

```
@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

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 ');


    }
}
```

APPEX TRIGGERS

GET STARTED WITH APEX TRIGGERS:

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:

ClosedOpportunityTrigger.apxt

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

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

    //Interate through opps in this trigger (updated or created opps)
    for (Opportunity opp : Trigger.New) {

        //Check if stage is 'Closed Won'
        if (opp.StageName=='Closed Won') {

            //If stage is 'Closed won', create task and associate it with Opportunity via the WhatId field
            newTasks.add(new Task(
                OwnerId=opp.OwnerId,
                Subject='Follow up Task',
                WhatId = opp.Id,
                ActivityDate= System.Today().addDays(2)
            ));

        }
    }

    //Check if there were any new tasks created
    if(newTasks.size()>0){
```

```
        insert newTasks;
    }

}
```

APPEX TESTING

GET STARTED WITH APEX UNIT TEST:

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; }
 }

 //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());
27          Date lastDay = Date.newInstance(date1.year(), date1.month(),
    totalDays);
28          return lastDay;
29      }
30
31  }
```

TestVerifyDate.apxc

```
1   @isTest
2   private class TestVerifyDate {
3
4       //testing that if date2 is within 30 days of date1, should return date 2
5       @isTest static void testDate2within30daysofDate1() {
6           Date date1 = date.newInstance(2018, 03, 20);
7           Date date2 = date.newInstance(2018, 04, 11);
8           Date resultDate = VerifyDate.CheckDates(date1,date2);
9           Date testDate = Date.newInstance(2018, 04, 11);
10          System.assertEquals(testDate,resultDate);
11      }
12
13      //testing that date2 is before date1. Should return "false"
14      @isTest static void testDate2beforeDate1() {
15          Date date1 = date.newInstance(2018, 03, 20);
16          Date date2 = date.newInstance(2018, 02, 11);
17          Date resultDate = VerifyDate.CheckDates(date1,date2);
18          Date testDate = Date.newInstance(2018, 02, 11);
19          System.assertNotEquals(testDate, resultDate);
20      }
21
22      //Test date2 is outside 30 days of date1. Should return end of month.
23      @isTest static void testDate2outside30daysofDate1() {
24          Date date1 = date.newInstance(2018, 03, 20);
25          Date date2 = date.newInstance(2018, 04, 25);
26          Date resultDate = VerifyDate.CheckDates(date1,date2);
27          Date testDate = Date.newInstance(2018, 03, 31);
28          System.assertEquals(testDate,resultDate);
29      }
30  }
```

TEST APEX TRIGGERS:

RestrictContactByName.apxt

```
@isTest
public class TestRestrictContactByName {
    @isTest static void TestInsertContact_INVALIDNAME(){
       Contact contact = new Contact(LastName='INVALIDNAME');
        Test.startTest();
        Database.SaveResult result = Database.insert(contact);
        Test.stopTest();
        System.assert(!result.isSuccess());
        System.assert(result.getErrors().size() > 0);
        System.assertEquals('The Last Name "'+contact.LastName+'" is

                            result.getErrors()[0].getMessage());
    }

    @isTest static void TestInsertContact_VALIDNAME(){
       Contact contact = new Contact(LastName='VALIDNAME');
        Test.startTest();
        Database.SaveResult result = Database.insert(contact);
        Test.stopTest();
        System.assert(result.isSuccess());
    }

    @isTest(SeeAllData=true) static void
TestUpdateContact_INVALIDNAME(){
       Contact contact = [SELECT Id FROM Contact WHERE LastName =
'Test Contact' LIMIT 1];
        contact.LastName = 'INVALIDNAME';
        Test.startTest();
        Database.SaveResult result = Database.update(contact);
        Test.stopTest();
        System.assert(!result.isSuccess());
        System.assert(result.getErrors().size() > 0);
        System.assertEquals('The Last Name "'+contact.LastName+'" is

                            result.getErrors()[0].getMessage());
    }

```

```
    @isTest(SeeAllData=true) static void TestUpdateContact_VALIDNAME(){
        Contact contact = [SELECT Id FROM Contact WHERE LastName = 'Test Contact' LIMIT 1];
         contact.LastName = 'VALIDNAME';
         Test.startTest();
         Database.SaveResult result = Database.update(contact);
         Test.stopTest();
         System.assert(result.isSuccess());
    }
}
```

CREATE TEST DATA FOR APEX TESTS:

RandomContactFactory.apxc

```
public class RandomContactFactory{
   public static List<Contact> generateRandomContacts(integer n,string lastname){
        integer n1= n;
        list<contact> c=[select  FirstName from contact limit : n1];
        return c;
    }

}


but for me it was giving error of class not created as static or either the method is not returning the first name. so for me it worked as follows:

public class RandomContactFactory{

    public static List<Contact> generateRandomContacts(integer n,string LastName){
    integer n1=n;
    List<contact> c1 = new list<contact>();
    list<contact> c2 =new list<contact>();
     c1 = [select FirstName from Contact Limit : n1];
     integer i=0;
     for(contact cnew : c1){
```

```
22      contact cnew1 = new contact();
23      cnew1.firstname = cnew.firstname + i;
24
25      c2.add(cnew1);
26      i++;
27      }
28     return c2;
29
30     }
31 }
```

ASYNCHRONOUS APEX

USE FUTURE METHODS:

AccountProcessor.apxc

```
1  global class AccountProcessor {
2   @future
3      public static void countContacts(Set<id> setId){
4          List<Account> lstAccount = [SELECT Id,
   Number_of_Contacts__c, (SELECT Id FROM Contacts) FROM Account where
   id in : setid];
5          for(Account acc: lstAccount){
6              List<Contact> lstCont=acc.contacts;
7    acc.Number_of_Contacts__c = lstCont.size();
8          }
9          update lstAccount;
10     }
11 }
```

AccountProcessorTest.apxc

```
1  @isTest private class AccountProcessorTest {
2      @isTest static void countContacts() {
3          Test.setMock(AccountProcessorTest.class, new Account());
4          Test.startTest();
5    AccountProcessor.countContacts();
```

```
        Test.stopTest();

    }
}
```

USE BATCH APEX:

LeadProcessor.apxc

```
global class LeadProcessor implements Database.Batchable<Sobject>
{
    global Database.QueryLocator start(Database.BatchableContext bc)
    {
        Database.QueryLocator qObj;
        qObj = Database.getQueryLocator('select Id, Name,

        return qObj;
    }

    global void execute(Database.BatchableContext bc, List<sObject>
records)
    {

        List<Lead> leadRecords = (List<Lead>) records;
        List<Lead> updatingLeadList = new List<Lead>();
            for (Lead leadObj : leadRecords)
            {
                leadObj.LeadSource = 'Dreamforce';
                updatingLeadList.add(leadObj);
            }
        if(!updatingLeadList.isEmpty()){
            update updatingLeadList;
        }

    }

    global void finish(Database.BatchableContext bc){   }
}
```

LeadProcessorTest.apxc

```
@isTest
private class LeadProcessorTest
{
    private static testMethod void LeadProcess()
    {
        List<Lead> lstLead = new List<Lead>();
        for(Integer i=0 ;i <200;i++)
        {
            lstLead.add(new Lead(LastName ='LastName'+i, Company
='demo'+i, City='New York', Country='US', LeadSource='Phone

        }

        insert lstLead;

        Test.startTest();

            LeadProcessor obj = new LeadProcessor();
            DataBase.executeBatch(obj);

        Test.stopTest();
    }
}
```

CONTROL PROCESSES WITH QUEUEABLE APEX:

AddPrimaryContact.apxc

```
public class AddPrimaryContact implements Queueable {
public contact c;
public String state;

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

public void execute(QueueableContext qc) {
```

```
system.debug('this.c = '+this.c+' this.state = '+this.state);
List<Account> accList = new List<account>([select id, name,
BillingState from account where account.BillingState = :this.state
limit 200]);
List<contact> insertContact = new List<contact>();
for(account a: accList) {
contact c = new contact();
c = this.c.clone(false, false, false, false);
c.AccountId = a.Id;
insertContact.add(c);
}
insert insertContact;
}

}
```

AddPrimaryContactTest.apxc

```
@isTest
public class AddPrimaryContactTest {

@testSetup
static void setup() {
List<Account> insertAccount = new List<Account>();
for(integer i=0; i<=100; i++) {
if(i <=50) {
insertAccount.add(new Account(Name='Acc'+i, BillingState = 'NY'));
} else {
insertAccount.add(new Account(Name='Acc'+i, BillingState = 'CA'));
}
}
insert insertAccount;
}

static testMethod void testAddPrimaryContact() {
Contact con = new Contact(LastName = 'LastName');
AddPrimaryContact addPC = new AddPrimaryContact(con, 'CA');
Test.startTest();
system.enqueueJob(addPC);
Test.stopTest();

```

```
24 system.assertEquals(50, [select count() from Contact]);
25 }
26
27 }
```

SCHEDULE JOBS USING APEX SCHEDULER:
DailyLeadProcessor.apxc

```
1  global class DailyLeadProcessor implements Schedulable{
2      global void execute(SchedulableContext ctx){
3          List<Lead> leads = [SELECT Id, LeadSource FROM Lead WHERE
   LeadSource = ''];
4
5          if(leads.size() > 0){
6              List<Lead> newLeads = new List<Lead>();
7
8              for(Lead lead : leads){
9                  lead.LeadSource = 'DreamForce';
10                 newLeads.add(lead);
11             }
12
13             update newLeads;
14         }
15     }
16 }
```

DailyLeadProcessorTest.apxc

```
1
2  @isTest
3  private class DailyLeadProcessorTest{
4      //Seconds Minutes Hours Day_of_month Month Day_of_week
   optional_year
5      public static String CRON_EXP = '0 0 0 2 6 ? 2022';
6
7      static testmethod void testScheduledJob(){
8          List<Lead> leads = new List<Lead>();
9
10         for(Integer i = 0; i < 200; i++){
11             Lead lead = new Lead(LastName = 'Test ' + i, LeadSource
```

```
    = '', Company = 'Test Company ' + i, Status = 'Open - Not

            leads.add(lead);
        }

        insert leads;

        Test.startTest();
        // Schedule the test job
        String jobId = System.schedule('Update LeadSource to


        // Stopping the test will run the job synchronously
        Test.stopTest();
    }
}
```

APEX INTEGRATION SERVICES

APEX REST CALLOUTS:

AnimalLocator.apxc

```
public class AnimalLocator {
    public static string getAnimalNameById(integer numSubmitted) {
        Http http = new Http();
        HttpRequest request = new HttpRequest();
        request.setEndpoint('https://th-apex-http-

        request.setMethod('GET');
        HttpResponse response = http.send(request);
        string replyName = 'None returned';
        if (response.getStatusCode() == 200) {
         replyName = response.getBody();
        }
            return replyName;
    }
}
```

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('Charles H Bones Esquire');
        response.setStatusCode(200);
        return response;
    }
}
```

AnimalLocatorTest.apxc

```
@IsTest
private class AnimalLocatorTest {
    @isTest static  void  testGetCallout() {
    // Set mock callout class
    Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
    // This causes a fake response to be sent
    // from the class that implements HttpCalloutMock.
    String animalname = AnimalLocator.getAnimalNameById(2);
    // Verify that the response received contains fake values
    String expectedValue = 'Charles H Bones Esquire';
    System.assertEquals(animalname, expectedValue);
    }
}
```

APEX SOAP CALLOUTS:

ParkService.apxc

```
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'};
6           private String[] field_order_type_info = new
String[]{'return_x'};
7       }
8       public class byCountry {
9           public String arg0;
10          private String[] arg0_type_info = new
String[]{'arg0','http://parks.services/',null,'0','1','false'};
11          private String[] apex_schema_type_info = new
String[]{'http://parks.services/','false','false'};
12          private String[] field_order_type_info = new
String[]{'arg0'};
13      }
14      public class ParksImplPort {
15          public String endpoint_x = 'https://th-apex-soap-

16          public Map<String,String> inputHttpHeaders_x;
17          public Map<String,String> outputHttpHeaders_x;
18          public String clientCertName_x;
19          public String clientCert_x;
20          public String clientCertPasswd_x;
21          public Integer timeout_x;
22          private String[] ns_map_type_info = new
String[]{'http://parks.services/', 'ParkService'};
23          public String[] byCountry(String arg0) {
24              ParkService.byCountry request_x = new
ParkService.byCountry();
25              request_x.arg0 = arg0;
26              ParkService.byCountryResponse response_x;
27              Map<String, ParkService.byCountryResponse>
response_map_x = new Map<String, ParkService.byCountryResponse>();
28              response_map_x.put('response_x', response_x);
29              WebServiceCallout.invoke(
30                this,
31                request_x,
32                response_map_x,
33                new String[]{endpoint_x,
34                '',
35                'http://parks.services/',
36                'byCountry',
37                'http://parks.services/',
38                'byCountryResponse',
```

```
                'ParkService.byCountryResponse'}
            );
            response_x = response_map_x.get('response_x');
            return response_x.return_x;
        }
    }
}
```

ParkLocatorTest.apxc

```
@isTest
private class ParkLocatorTest {
    @isTest static void testCallout() {
        // This causes a fake response to be generated
        Test.setMock(WebServiceMock.class, new ParkServiceMock());
        // Call the method that invokes a callout
        //Double x = 1.0;
        //Double result = AwesomeCalculator.add(x, y);

        String country = 'Germany';
        String[] result = ParkLocator.Country(country);


        // Verify that a fake result is returned
        System.assertEquals(new List<String>{'Hamburg Wadden Sea
l


    }
}
```

APEX WEB SERVICES:

AccountManager.apxc

```
@RestResource(urlMapping='/Accounts/*/contacts')
global with sharing class AccountManager {
    @HttpGet
    global static Account getAccount() {
        RestRequest request = RestContext.request;
        // grab the accountId from the end of the URL
```

```
        String accountId =
request.requestURI.substringBetween('Accounts/','/contacts');
        System.debug('Account ID: ' + accountId);
        Account result =  [SELECT Id, Name, (SELECT Id, Name FROM
Contacts) FROM Account WHERE Id = :accountId LIMIT 1];
        System.debug ('Account: ' + result);
        return result;
    }
}
```

AccountManagerTest.apxc

```
@IsTest
private class AccountManagerTest {
 @IsTest static void testGetAccount() {
        Id recordId = createTestRecord();
        //Id recordId = '0015J000002v8JjQAI';
        System.debug('Test ID: ' + recordId);
        // Set up a test request
        RestRequest request = new RestRequest();
        request.requestUri =
'https://myInstance.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);
    }

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

```
29     }
30 }
```