Apex Specialist Super Badge

In this project doc the code for the apex specialist super badge has been provided which was completed during the virtual internship program of salesforce using the trailblazer platform.

The code has been divided into two types. The files with the extension of .apxc are the apex classes and the one with the extension of .apxtare apex trigger classes.

The naming convention of the files has been done in accordance with the requirements of the super badge.

Apex Trigger Classes

1. **MaintenanceRequest.apxt**

```
trigger MaintenanceRequest on Case (before update, after update)
{
   if(trigger.isAfter)
   {
      MaintenanceRequestHelper.updateWorkOrders();
   }
}
```

Apex Classes

1. **MaintenanceRequestHelper.apxc**

```
public with sharing class MaintenanceRequestHelper {
   public static void updateWorkOrders() {
      List<case> newCaseList = new List<case>();
      Integer avgAmount=10000;
```

```
        List<Equipment_Maintenance_Item__c> newEMI = new
List<Equipment_Maintenance_Item__c>();
        List<case> caseList = [SELECT id,Vehicle__c,Subject,ProductID,Product__c, (SELECT
id from Equipment_Maintenance_Items__r) from case where status='closed' and Type
IN ('Repair', 'Routine Maintenance') and ID IN :Trigger.new LIMIT 200];
        Map<id,Equipment_Maintenance_Item__c> equip = new
map<id,Equipment_Maintenance_Item__c>([Select ID, Equipment__c,
Quantity__c,Equipment__r.id,Equipment__r.Maintenance_Cycle__c from
Equipment_Maintenance_Item__c ]);
        for(case c: caseList){
            case newCase = new Case();
            newCase.Type = 'Routine Maintenance';
            newCase.Status = 'New';
            newCase.Vehicle__c = c.Vehicle__c;
            newCase.Subject =  String.isBlank(c.Subject) ? 'Routine Maintenance Request' :
c.Subject;
            newCase.Date_Reported__c = Date.today();
            newCase.ProductId = c.ProductId;
            newCase.Product__c = c.Product__c;
            newCase.parentID = c.Id;


            for(Equipment_Maintenance_Item__c emi : c.Equipment_Maintenance_Items__r ){
                avgAmount =
Math.min(avgAmount,Integer.valueOf(equip.get(emi.id).Equipment__r.Maintenance_Cyc
le__c));
                newEMI.add(new Equipment_Maintenance_Item__c(
                    Equipment__c = equip.get(emi.id).Equipment__c,
                    Maintenance_Request__c = c.id,
                    Quantity__c = equip.get(emi.id).Quantity__c));
            }
            Date dueDate = date.TODAY().adddays(avgAmount);
            newCase.Date_Due__c =dueDate;
            newCaseList.add(newCase);

        }
        if(newCaseList.size()>0){
            Database.insert(newCaseList);
        }

        for(Case c2: newCaseList){
            for(Equipment_Maintenance_Item__c emi2 : newEmi){
                if(c2.parentID == emi2.Maintenance_Request__c){
                    emi2.Maintenance_Request__c = c2.id;
                }
            }
        }

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

2. **<u>MaintenanceRequestHelperTest.apxc</u>**

```
@istest
public with sharing class MaintenanceRequestHelperTest {
    @istest
    public static void BulkTesting(){
        product2 pt2 = new product2(Name = 'tester',Maintenance_Cycle__c = 10,
Replacement_Part__c = true);
        Database.insert(pt2);
        List<case> caseList = new List<case>();
        for(Integer i=0;i<300;i++){
            caseList.add(new case(
                Type = 'Routine Maintenance',
                Status = 'Closed',
                Subject = 'testing',
                Date_Reported__c = Date.today(),
                ProductId = pt2.id
            ));
        }
        if(caseList.size()>0){
            Database.insert(caseList);
            System.debug(pt2.id);
            System.debug(caseList.size());
        }
        List<Equipment_Maintenance_Item__c> newEMI = new
List<Equipment_Maintenance_Item__c>();
        for(Integer i=0;i<5;i++){
            newEMI.add(new Equipment_Maintenance_Item__c(
                Equipment__c = pt2.id,
                Maintenance_Request__c = caseList[1].id,
                Quantity__c = 10));
        }
        if(newEmi.size()>0){
            Database.insert(newEmi);
        }
        for(case c :caseList){
            c.Subject = 'For Testing';
        }
        Database.update(caseList);
        Integer newcase = [Select count() from case where ParentId = :caseList[0].id];
        System.assertEquals(1, newcase);
    }
    @istest
    public static void positive(){
        product2 pt2 = new product2(Name = 'tester',Maintenance_Cycle__c = 10);
        insert pt2;

        Case cParent = new Case(Type = 'Repair',status = 'Closed',Date_Reported__c =
Date.today(), ProductId = pt2.id);
        insert cParent;
        Case cChild = new Case(Type = 'Repair',status = 'Closed',Date_Reported__c =
Date.today(), ProductId = pt2.id,parentID = cParent.ParentId);
        insert cChild;
        cParent.subject = 'child refrecer record';
        update cParent;
        Integer newcase = [Select count() from case where ParentId = :cParent.id];
        System.assertEquals(1, newcase);
    }
    @istest public static void negetive(){
        product2 pt2 = new product2(Name = 'tester',Maintenance_Cycle__c = 10);
        insert pt2;
        Case c = new Case(Type = 'Repair',status = 'New',Date_Reported__c = Date.today(),
                          ProductId = pt2.id);
        insert c;
        c.Status = 'Working';
        update c;
        Integer newcase = [Select count() from case where ParentId = :c.id];
        System.assertEquals(0, newcase);
    }
}
```

3. **WarehouseCalloutService.apxc**

```
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 runWarehouseEquipmentSync(){
        System.debug('go into runWarehouseEquipmentSync');
        Http http = new Http();
        HttpRequest request = new HttpRequest();

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

        List<Product2> product2List = new List<Product2>();
        System.debug(response.getStatusCode());
        if (response.getStatusCode() == 200){
            List<Object> jsonResponse =
(List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());
            for (Object jR : jsonResponse){
                Map<String,Object> mapJson = (Map<String,Object>)jR;
                Product2 product2 = new Product2();
                //replacement part (always true),
                product2.Replacement_Part__c = (Boolean) mapJson.get('replacement');
                //cost
                product2.Cost__c = (Integer) mapJson.get('cost');
                //current inventory
                product2.Current_Inventory__c = (Double) mapJson.get('quantity');
                //lifespan
                product2.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
                //maintenance cycle
                product2.Maintenance_Cycle__c = (Integer)mapJson.get('maintenanceperiod');
                //warehouse SKU
                product2.Warehouse_SKU__c = (String) mapJson.get('sku');
                product2.Name = (String) mapJson.get('name');
                product2.ProductCode = (String) mapJson.get('_id');
                product2List.add(product2);
            }
            if (product2List.size() > 0){
                upsert product2List;
                System.debug('Your equipment was synced with the warehouse one');
            }
        }
    }
    public static void execute (QueueableContext context){
        System.debug('start runWarehouseEquipmentSync');
        runWarehouseEquipmentSync();
        System.debug('end runWarehouseEquipmentSync');
    }
}
```

4. **<u>WarehouseCalloutServiceMock.apxc</u>**

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

        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"},{"_id":"55d66226
726b611100aaf742","replacement":true,"quantity":183,"name":"Cooling
Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004"},{"_id":"55d66226726b6
11100aaf743","replacement":true,"quantity":143,"name":"Fuse
20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"}]');
        response.setStatusCode(200);
        return response;
    }
}
```

5. **WarehouseCalloutServiceTest.apxc**

```
@IsTest
private class WarehouseCalloutServiceTest {
            @isTest
    static void testWarehouseCallout() {
        test.startTest();
        test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
        WarehouseCalloutService.execute(null);
        test.stopTest();

        List<Product2> product2List = new List<Product2>();
        product2List = [SELECT ProductCode FROM Product2];

        System.assertEquals(3, product2List.size());
        System.assertEquals('55d66226726b611100aaf741',
product2List.get(0).ProductCode);
        System.assertEquals('55d66226726b611100aaf742',
product2List.get(1).ProductCode);
        System.assertEquals('55d66226726b611100aaf743',
product2List.get(2).ProductCode);
    }
}
```

6. **<u>WarehouseSyncSchedule.apxc</u>**

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

7. **<u>WarehouseSyncScheduleTest.apxc</u>**

```
@isTest
public with sharing class WarehouseSyncScheduleTest {

   @isTest static void test() {
      String scheduleTime = '00 00 00 * * ? *';
      Test.startTest();
      Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
      String jobId = System.schedule('Warehouse Time to Schedule to test',
scheduleTime, new WarehouseSyncSchedule());
      CronTrigger c = [SELECT State FROM CronTrigger WHERE Id =: jobId];
      System.assertEquals('WAITING', String.valueOf(c.State), 'JobId does not match');

      Test.stopTest();
   }
}
```