

# Apex Specialist Superbadge:

In this superbadge, initial step is to create a new playground. Now the steps which are mentioned in 'set up development org' has to be done. Then according to the given process, write the code for each step mentioned below:

Step 1 : Answering the multiple choice questions.

Step 2 - Automate Record Creation :

Automate record creation using apex triggers.

Go to developer console and edit the apex class and the triggers for below:

MaintenanceRequestHelper

```
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     For (Case c : updWorkOrders){
5         if (nonUpdCaseMap.get(c.Id).Status != 'Closed' &&
      c.Status == 'Closed'){
6             if (c.Type == 'Repair' || c.Type ==
      'Routine
7                 validIds.add(c.Id);
8     }
9     }
10 }
11         if (!validIds.isEmpty()){
12             List<Case> newCases = new
      List<Case>();
13 Map<Id,Case> closedCasesM = new
      Map<Id,Case>([SELECT
14 Id, Vehicle_c, Equipment_c,
15 Equipment_r.Maintenance_Cycle_c,(SELECT
16 Id,Equipment_c,Quantity_c FROM Equipment_Maintenance_Items_r)
17 FROM
```

Case            Id

```
19     Map<Id,Decimal> maintenanceCycles = new
20 Map<ID,Decimal>();
21     AggregateResult[] results = [SELECT
    Maintenance_Request_c,
22 MIN(Equipment_r.Maintenance_Cycle    c)cycle FROM
    Equipment_Maintenance_Item_c WHERE Maintenance_Request_c IN
23 :ValidIds GROUP BY Maintenance_Request_c];
24
25     for (AggregateResult ar : results){
26         maintenanceCycles.put((Id) ar.get('Maintenance_Request
    c'), (Decimal) ar.get('cycle'));
27     }
28
29     for(Case cc : closedCasesM.values()){
30         Case nc = new Case (
31             ParentId = cc.Id,
32             Status = 'New',
33             Subject = 'Routine Maintenance',
34             Type = 'Routine Maintenance',
35             Vehicle_c = cc.Vehicle_c,
36             Equipment_c =cc.Equipment_c,
37             Origin = 'Web',
38             Date_Reported_c = Date.Today()
39         );
40
41         If (maintenanceCycles.containsKey(cc.Id)){
42             nc.Date_Due_c =
43             Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
44         }
45         newCases.add(nc);    }
46
47     insert newCases;
48     List<Equipment_Maintenance_Item    c> clonedWPs = new
```

## List

```
49     for (Case nc : newCases){
50         for (Equipment_Maintenance_Item_c wp ;
51 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 insert ClonedWPs;
58 }
59 }
```

## MaintenanceRequestHelperTest

```
1 @istest
2 public with sharing class MaintenanceRequestHelperTest {
3     private static final string STATUS_NEW = 'New';
4     private static final string WORKING = 'Working';
5     private static final string CLOSED = 'Closed';
6     private static final string REPAIR = 'Repair';
7     private static final string REQUEST_ORIGIN = 'Web';
8     private static final string REQUEST_TYPE = 'Routine
9     private static final string REQUEST_SUBJECT = 'Testing
10 PRIVATE STATIC Vehicle_c createVehicle(){
11 Vehicle_c Vehicle = new Vehicle_C(name = 'SuperTruck');
12     return Vehicle;
13 }
14 PRIVATE STATIC Product2 createEq(){ product2 equipment = new
15     product2(name =
16 'SuperEquipment',
17 lifespan_months C = 10, maintenance_cycle C =
18 10,
19 replacement_partc =
20 true);
21     return equipment;
22 }
23 PRIVATE STATIC Case createMaintenanceRequest(id
24 vehicleId, id equipmentId){
25     case cs = new case(Type=REPAIR,
26     Status=STATUS_NEW,
27     Origin=REQUEST_ORIGIN,
28     Subject=REQUEST_SUBJECT,
29     Equipment_c=equipmentId,
30     Vehicle_c=vehicleId);
31     return cs;
32 }
33 PRIVATE STATIC Equipment_Maintenance_Item_c
34 createWorkPart(id equipmentId,id requestId){
35     Equipment_Maintenance_Item_c wp = new
```

## Equipment\_Maintenance\_Item\_c Equipment\_c

```
32
33 Maintenance_Request_c = requestId);
34     return wp;
35 }
36
37
38 @istest
39 private static void testMaintenanceRequestPositive(){
40     Vehicle_c vehicle = createVehicle();
41     insert vehicle;
42     id vehicleId = vehicle.Id;
43     Product2 equipment = createEq();
44     insert equipment;
45     id equipmentId = equipment.Id;
46     case somethingToUpdate =
        createMaintenanceRequest(vehicleId,equipmentId);
47     insert somethingToUpdate;
48     Equipment_Maintenance_Item_c workP =
        createWorkPart(equipmentId,somethingToUpdate.id);
49     insert workP;
50     test.startTest();
51     somethingToUpdate.status = CLOSED;
52     update somethingToUpdate;
53     test.stopTest();
54     Case newReq = [Select id, subject, type, Equipment_c,
        Date_Reported_c, Vehicle_c, Date_Due-c
55     from case
56     where status =:STATUS_NEW];
57     Equipment_Maintenance_Item_c workPart = [select id
58     from
59     Equic
60     where
61     Maintenance_Request_c =:newReq.Id];
62
63     system.assert(workPart != null);
```

assert                      Subject                      null

```
65  system.assertEquals(newReq.Type, REQUEST_TYPE);
66  SYSTEM.assertEquals(newReq.Equipment_c, equipmentId);
67  SYSTEM.assertEquals(newReq.Vehicle_c, vehicleId);
68  SYSTEM.assertEquals(newReq.Date_Reported_c,
    system.today());
69  }
70
71  @istest
72  private static void testMaintenanceRequestNegative(){
73  Vehicle-C vehicle = createVehicle();
74  insert vehicle;
75  id vehicleId = vehicle.Id;
76  product2 equipment = createEq();
77  insert equipment;
78  id equipmentId = equipment.Id;
79      case emptyReq =
    createMaintenanceRequest(vehicleId,equipmentId);
80  insert emptyReq;
81  Equipment_Maintenance_Item_c workP =
    createWorkPart(equipmentId, emptyReq.Id);
82  insert workP;
83  test.startTest();
84  emptyReq.Status = WORKING;
85  update emptyReq;
86  test.stopTest();
87  list<case> allRequest = [select id
88  from
89  case];
90  Equipment_Maintenance_Item_
91  workPart = [select id
92  from Equipment_Maintenance_Item_c
93  where Maintenance_Request_c = :emptyReq.Id];
94  system.assert(workPart != null);
95  system.assert(allRequest.size()
```

```
97  }
98
99  @istest
100     private static void
101 testMaintenanceRequestBulk(){
102     list<Vehicle_C> vehicleList = new list<Vehicle-C>();
103     list<Product2> equipmentList = new list<Product2>();
104
105 list<Equipment_Maintenance_Item_c> workPartList = new
    list<Equipment_Maintenance_Item_c>();
106     list<case> requestList = new
107 list<case>();
108     list<id> oldRequestIds = new
109
110 list<id>();
111     for(integer i = 0; i < 300;
112 i++){
113 vehicleList.add(createVehicle());
114 equipmentList.add(createEq());
115     }
116     insert vehicleList;
117     insert equipmentList;
118
119     for(integer i = 0; i < 300;
120 i++){
121 requestList.add(createMaintenanceRequest(vehicleList.get(i)
    .id, equipmentList.get(i).id));
122     }
123     insert requestList;
124
125     for(integer i = 0; i < 300;
126 i++){
127 workPartList.add(createWorkPart(equipmentList.get(i).id,
    requestList.get(i).id));
```

```

129     insert workPartList;
130
131     test.startTest();
132     for(case req : requestList){
133         req.Status = CLOSED;
134         oldRequestIds.add(req.Id);
135     }
136     update requestList;
137     test.stopTest();
138
139     list<case> allRequests = [select id
140         from
141     case
142         where
143 status =: STATUS_NEW];
144
145 list<Equipment_Maintenance_Item  c> workParts = [select id
146 from Equipment_Maintenance_Item  c
147 where Maintenance_Request      c in: oldRequestIds];
148 system.assert(allRequests.size()== 300);
149 }
150 }

```

### Step 3 - Synchronize the salesforce data with an external system:

Modify the Apex Classes as below, save and run all.

WarehouseCalloutService

```

1  public with sharing class WarehouseCalloutService implements
    Queueable {
2  private static final String WAREHOUSE_URL = 'https://th-
3      //class that makes a REST callout to an external warehouse

```



```
system to get a list of equipment that needs to be updated.
4      //The callout's JSON response returns the equipment
      records that you upsert in Salesforce.
5      @future(callout=true)
6      public static void runWarehouseEquipmentSync(){
7          Http http = new Http();
8          HttpRequest request = new HttpRequest();
9          request.setEndpoint(WAREHOUSE_URL);
10         request.setMethod('GET');
11         HttpResponse response = http.send(request);
12         List<Product2> warehouseEq = new List<Product2>();
13         if (response.getStatusCode() == 200){
14             List<Object> jsonResponse =
                (List<Object>)JSON.deserializeUntyped(response.getBody());
15             System.debug(response.getBody());
16             //class maps the following fields: replacement part
                (always true), cost, current inventory, lifespan, maintenance
                cycle, and warehouse SKU
17             //warehouse SKU will be external ID for identifying which
                equipment records to update within Salesforce
18             for (Object eq : jsonResponse){
19                 Map<String,Object> mapJson = (Map<String,Object>)eq;
20                 Product2 myEq = new Product2();
21                 myEq.Replacement_Part c = (Boolean)
                    mapJson.get('replacement');
22                 myEq.Name = (String) mapJson.get('name');
23                 myEq.Maintenance_Cycle_c = (Integer)
                    mapJson.get('maintenanceperiod');
24                 myEq.Lifespan_Months_c = (Integer)
                    mapJson.get('lifespan');
25                 myEq.Cost_c = (Integer) mapJson.get('cost');
26                 myEq.Warehouse_SKU_c = (String) mapJson.get('sku');
27                 myEq.Current_Inventory_c = (Double)
                    mapJson.get('quantity');
28                 myEq.ProductCode = (String) mapJson.get('_id');
```

```

29 warehouseEq.add(myEq);      }
30
31 if (warehouseEq.size() > 0){
32 upsert warehouseEq;
33 System.debug('Your equipment was synced with the
34
35 }
36 }
37 }
38
39 public static void execute (QueueableContext context){
40 runWarehouseEquipmentSync();
41 }

```

#### Step 4 - Schedule Synchronization:

Modify the Apex Classes as below, save and run all.

WarehouseSyncSchdeule

```

1  global with sharing class WarehouseSyncSchedule implements
    Schedulable{
2  global void execute(SchedulableContext ctx){
3  System.enqueueJob(new WarehouseCalloutService());      }
4  }
5

```

## Step 5 - Test automationlogic :

Modify the Apex Classes as below, save and run all.

### MaintenanceRequestHelper

```
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      For (Case c : updWorkOrders){
5          if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status ==
            'Closed'){
6              if (c.Type == 'Repair' || c.Type == 'Routine
7              validIds.add(c.Id);
8          }
9      }
10 }
11 if (!validIds.isEmpty()){
12     List<Case> newCases = new List<Case>();
13     Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id,
        Vehicle_c, Equipment_c,
```

```

14 Equipment    r.Maintenance_Cycle    c,(SELECT
15 Id,Equipment_c,Quantity_c FROM Equipment_Maintenance_Items_r)
16 FROM
17 Case WHERE Id IN :validIds]);
18     Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
19     AggregateResult[] results = [SELECT Maintenance_Request_c,
20 MIN(Equipment_r.Maintenance_Cycle_c)cycle FROM
    Equipment_Maintenance_Item_c WHERE Maintenance_Request_c IN
21 :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
37 );
38
39 If (maintenanceCycles.containsKey(cc.Id)){
40     nc.Date_Due_c =
41 Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
42 }
43 newCases.add(nc);
44 }
45 insert newCases;
46 List<Equipment_Maintenance_Item_c> clonedWPs = new
    List<Equipment_Maintenance_Item_c>();
47 for (Case nc : newCases){
48 for (Equipment_Maintenance_Item_c wp :
    closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items_r){

```

## Equipment\_Maintenance\_Item\_c

```
50 wp.clone();
51 wpClone.Maintenance_Request_c = nc.Id; ClonedWPs.add(wpClone);
52
53 }
54 }
55 insert ClonedWPs;
56 }
```

## MaintenanceRequestHelperTest

```
1 @istest
2 public with sharing class MaintenanceRequestHelperTest {
3     private static final string STATUS_NEW = 'New';
4     private static final string WORKING = 'Working';
5     private static final string CLOSED = 'Closed';
6     private static final string REPAIR = 'Repair';
7     private static final string REQUEST_ORIGIN = 'Web';
8     private static final string REQUEST_TYPE = 'Routine
9     private static final string REQUEST_SUBJECT = 'Testing
10 PRIVATE STATIC Vehicle_c createVehicle(){
11 Vehicle_c Vehicle = new Vehicle_C(name = 'SuperTruck'); return
```

```

    Vehicle;
12 }
13 PRIVATE STATIC Product2 createEq(){ product2 equipment = new
    product2(name =
14 'SuperEquipment',
15 lifespan_months_C = 10, maintenance_cycle_C =
16 10,
17 replacement_part_c =
18 true);
19 return equipment;
20 }
21 PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id
    equipmentId){
22 case cs = new case(Type=REPAIR,
23     Status=STATUS_NEW,
24     Origin=REQUEST_ORIGIN,
25     Subject=REQUEST_SUBJECT,
26     Equipment_c=equipmentId,
27     Vehicle_c=vehicleId);
28 return cs;    }
29 PRIVATE STATIC Equipment_Maintenance_Item_c
    createWorkPart(id equipmentId,id requestId){
30     Equipment_Maintenance_Item_c wp = new
    Equipment_Maintenance_Item_c(Equipment_c = equipmentId,
31
32 Maintenance_Request_c = requestId);
33     return wp;    }
34
35
36 @istest
37 private static void testMaintenanceRequestPositive(){
38 Vehicle    c vehicle = createVehicle();
39 insert vehicle;
40 id vehicleId = vehicle.Id; 47
41 Product2 equipment = createEq();
42 insert equipment;
43 id equipmentId = equipment.Id; 51
44     case somethingToUpdate =
    createMaintenanceRequest(vehicleId,equipmentId);
45 insert somethingToUpdate; 54
46     Equipment_Maintenance_Item_c workP =

```

```

        createWorkPart(equipmentId,somethingToUpdate.id);
47  insert workP;
48  test.startTest();
49  somethingToUpdate.status = CLOSED;
50  update somethingToUpdate;
51  test.stopTest();
52      Case newReq = [Select id, subject, type, Equipment_c,
    Date_Reported_c, Vehicle_c, Date_Due_
53  from case
54  where status =:STATUS_NEW];
55  Equipment_Maintenance_Item_c workPart = [select id
56  from
57  Equipment_Maintenance_Item_c
58  where
59  Maintenance_Request_c =:newReq.Id];
60
61  system.assert(workPart != null);
62  system.assert(newReq.Subject != null);
63  system.assertEquals(newReq.Type, REQUEST_TYPE);
64  SYSTEM.assertEquals(newReq.Equipment_c, equipmentId);
65  SYSTEM.assertEquals(newReq.Vehicle_c, vehicleId);
66  SYSTEM.assertEquals(newReq.Date_Reported_c, system.today());
67  }
68
69  @istest
70  private static void testMaintenanceRequestNegative(){
71  Vehicle_C vehicle = createVehicle();
72  insert vehicle;
73  id vehicleId = vehicle.Id;
74  product2 equipment = createEq();
75  insert equipment;
76  id equipmentId = equipment.Id;
77      case emptyReq =
    createMaintenanceRequest(vehicleId,equipmentId);
78  insert emptyReq;
79      Equipment_Maintenance_Item_c workP =
    createWorkPart(equipmentId, emptyReq.Id);
80  insert workP;
81  test.startTest();
82  emptyReq.Status = WORKING;
83  update emptyReq;

```

```

84 test.stopTest();
85 list<case> allRequest = [select id
86 from
87 case];
88 Equipment_Maintenance_Item_
89 workPart = [select id
90 from Equipment_Maintenance_Item_
91 where Maintenance_Request_c = :emptyReq.Id];
92 system.assert(workPart != null);
93 system.assert(allRequest.size()
94 == 1);
95 }
96 @istest
97 private static void
98 testMaintenanceRequestBulk(){
99 list<Vehicle C> vehicleList = new list<Vehicle_C>();
100 list<Product2> equipmentList = new list<Product2>();
101
102 list<Equipment_Maintenance_Item_c> workPartList = new
103 list<Equipment_Maintenance_Item_c>();
104 list<case> requestList = new
105 list<case>();
106 list<id> oldRequestIds = new
107 list<id>();
108 for(integer i = 0; i < 300;
109 i++){
110 vehicleList.add(createVehicle());
111 equipmentList.add(createEq());
112 }
113 insert vehicleList;
114 insert equipmentList;
115
116 for(integer i = 0; i < 300;
117 i++){
118 requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
119 equipmentList.get(i).id));
120 }
121 insert requestList;
122
123 for(integer i = 0; i < 300;

```



```

122 i++){
123 workPartList.add(createWorkPart(equipmentList.get(i).id,
    requestList.get(i).id));
124     }
125     insert workPartList;
126 test.startTest();
127 for(case req : requestList){ req.Status = CLOSED;
    oldRequestIds.add(req.Id);
128 }
129 update requestList;
130 test.stopTest();
131
132 list<case> allRequests = [select id
133 from
134 case
135 where
136 status =: STATUS_NEW];
137 list<Equipment_Maintenance_Item c> workParts = [select id
138 from Equipment_Maintenance_Item_
139 where Maintenance_Request_c in: oldRequestIds];
140 system.assert(allRequests.size()== 300);
141 }
142 }
143

```

**1**

## Step 6 - Test callout logic :

Modify the Apex Classes as below, save and run all.

## WarehouseCalloutServiceTest

```
1 @isTest
2 private class WarehouseCalloutServiceTest { @isTest
3 static void testWareHouseCallout(){ Test.startTest();
4 // implement mock callout test here
5 Test.setMock(HTTPCalloutMock.class, new
6 WarehouseCalloutServiceMock());
7 WarehouseCalloutService.execute(null);
8 test.stopTest();
9 System.assertEquals(1, [SELECT count() FROM Product2]);
```

## WarehouseCalloutServiceMock

```
1  @isTest
2  2global class WarehouseCalloutServiceMock implements
   HttpCalloutMock {
3    // implement http mock callout
4    global static HttpResponse respond(HttpRequest request){
5      System.assertEquals('https://th-superbadge-
6    });
7      System.assertEquals('GET', request.getMethod());
8      // Create a fake response
9      HttpResponse response = new HttpResponse();
10     response.setHeader('Content-Type', 'application/json');
11     response.setBody('{"_id":"55d66226726b611100aaf741","replacement
12     response.setStatusCode(200);
13     return response;
14 }
```

## Step 7 - Test scheduling logic :

Modify the Apex Classes as below, save and run all.

WarehouseSyncSchedule

```
1  global with sharing class WarehouseSyncSchedule implements
    Schedulable{
2  global void execute(SchedulableContext ctx){
3  System.enqueueJob(new WarehouseCalloutService());
4  }
5  }
```

## WarehouseSyncScheduleTest

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

System

Id 'Schedule '

```
12 }  
13 System.assertEquals(jobID, a.Id, 'Schedule ');  
14
```







