

Apex Specialist Super Badge

In this super badge, initial step is to create a new playground. Now the steps which are mentioned in "set up development org" has to be done. The step by step process is mentioned below :

step 1 : The first step is answering the multiple choice questions.

step 2 : Automate Record Creation :

Create automate record using apex triggers. Go to the setup and find the developer console and edit the apex class and the triggers for the below :

MaintenanceRequestHelper

```
1  public with sharing class MaintenanceRequestHelper {
2      public static void updateWorkOrders(List<Case> updWorkOrders,
3      Map<Id,Case> nonUpdCaseMap) {
4
5
6          For (Case c : updWorkOrders){
7              if (nonUpdCaseMap.get(c.Id).Status != 'Closed' &&
8              c.Status == 'Closed'){
9                  if (c.Type == 'Repair' || c.Type == 'Routine
9
10                     validIds.add(c.Id);
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
19         Id, Vehicle__c, Equipment__c,
20         Equipment__r.Maintenance_Cycle__c, (SELECT
21         Id,Equipment__c,Quantity__c FROM Equipment_Maintenance_Items__r)
22         FROM
23         Case WHERE Id IN :validIds]);
24         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 }

```

MaintenanceRequestHelperTest

```

1  @istest
2  public with sharing class MaintenanceRequestHelperTest {
3
4      private static final string STATUS_NEW = 'New';
5      private static final string WORKING = 'Working';
6      private static final string CLOSED = 'Closed';
7      private static final string REPAIR = 'Repair';
8      private static final string REQUEST_ORIGIN = 'Web';
9      private static final string REQUEST_TYPE = 'Routine'
10
11      private static final string REQUEST_SUBJECT = 'Testing'
12
13      PRIVATE STATIC Vehicle__c createVehicle(){
14          Vehicle__c Vehicle = new Vehicle__C(name = 'SuperTruck');
15          return Vehicle;
16      }
17
18      PRIVATE STATIC Product2 createEq(){
19          product2 equipment = new product2(name =
20              'SuperEquipment',
21              lifespan_months__C = 10,
22              maintenance_cycle__C =
23              10,
24              replacement_part__c =
25              true);

```

```

22         return equipment;
23     }
24
25     PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id
equipmentId){
26         case cs = new case(Type=REPAIR,
27                             Status=STATUS_NEW,
28                             Origin=REQUEST_ORIGIN,
29                             Subject=REQUEST_SUBJECT,
30                             Equipment__c=equipmentId,
31                             Vehicle__c=vehicleId);
32         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];
70
71         system.assert(workPart != null);
72         system.assert(newReq.Subject != null);
73         system.assertEquals(newReq.Type, REQUEST_TYPE);
74         SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
75         SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
76         SYSTEM.assertEquals(newReq.Date_Reported__c,
system.today());
77     }
78
79     @istest
80     private static void testMaintenanceRequestNegative(){
81         Vehicle__C vehicle = createVehicle();
82         insert vehicle;
83         id vehicleId = vehicle.Id;
84
85         product2 equipment = createEq();
86         insert equipment;
87         id equipmentId = equipment.Id;
88
89         case emptyReq =
createMaintenanceRequest(vehicleId,equipmentId);
90         insert emptyReq;

```

```

91
92     Equipment_Maintenance_Item__c workP =
    createWorkPart(equipmentId, emptyReq.Id);
93     insert workP;
94
95     test.startTest();
96     emptyReq.Status = WORKING;
97     update emptyReq;
98     test.stopTest();
99
100     list<case> allRequest = [select id
101                             from case];
102
103     Equipment_Maintenance_Item__c workPart = [select id
104                                                from
105                                                Equipment_Maintenance_Item__c
106                                                where
107                                                Maintenance_Request__c = :emptyReq.Id];
108
109     system.assert(workPart != null);
110     system.assert(allRequest.size() == 1);
111 }
112
113 @istest
114 private static void testMaintenanceRequestBulk(){
115     list<Vehicle__C> vehicleList = new list<Vehicle__C>();
116     list<Product2> equipmentList = new list<Product2>();
117     list<Equipment_Maintenance_Item__c> workPartList = new
118     list<Equipment_Maintenance_Item__c>();
119     list<case> requestList = new list<case>();
120     list<id> oldRequestIds = new list<id>();
121
122     for(integer i = 0; i < 300; i++){
123         vehicleList.add(createVehicle());
124         equipmentList.add(createEq());
125     }
126     insert vehicleList;
127     insert equipmentList;
128
129     for(integer i = 0; i < 300; i++){

```

```

127     requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
128         equipmentList.get(i).id));
129     insert requestList;
130
131     for(integer i = 0; i < 300; i++){
132         workPartList.add(createWorkPart(equipmentList.get(i).id,
133             requestList.get(i).id));
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
149             id
150                                                         from
151             Equipment_Maintenance_Item__c
152                                                         where
153             Maintenance_Request__c in: oldRequestIds];
154     }

```

step 3 : Synchronize the sales force data with an external system :
 Modify the apex as below save and run all

WarehouseCalloutService

```
1 public with sharing class WarehouseCalloutService {
2
3     private static final String WAREHOUSE_URL =
4         'https://th-superbadge-
5
6         //@future(callout=true)
7     public static void runWarehouseEquipmentSync(){
8
9         Http http = new Http();
10        HttpRequest request = new HttpRequest();
11
12        request.setEndpoint(WAREHOUSE_URL);
13        request.setMethod('GET');
14        HttpResponse response = http.send(request);
15
16        List<Product2> warehouseEq = new
17        List<Product2>();
18
19        if (response.getStatusCode() == 200){
20            List<Object> jsonResponse =
21            (List<Object>)JSON.deserializeUntyped(response.getBody());
22
23            System.debug(response.getBody());
24
25            for (Object eq : jsonResponse){
26                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
38
39         System.debug(warehouseEq);
40
41     }
42 }
43 }
```

step 4 : Schedule Synchronization :

Modify the Apex Classes as below save and run all

WarehouseSyncSchedule

```
1 global class WarehouseSyncSchedule implements
  Schedulable {
2     global void execute(SchedulableContext ctx) {
3
4
5     WarehouseCalloutService.runWarehouseEquipmentSync(
6     );
7     }
8 }
```

step 5: Test automation logic :

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
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
32
33             Type = 'Routine Maintenance',
34             Vehicle__c = cc.Vehicle__c,
35             Equipment__c = cc.Equipment__c,
36             Origin = 'Web',
37             Date_Reported__c = Date.Today()
38         );
39
40         If
41         (maintenanceCycles.containsKey(cc.Id)){
42             nc.Date_Due__c =
43             Date.today().addDays((Integer)
44             maintenanceCycles.get(cc.Id));
45         }
46
47         newCases.add(nc);
48     }
49     insert newCases;
50
51     List<Equipment_Maintenance_Item__c>
52     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

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}

```

MaintenanceRequestHelperTest

```

1  @istest
2  public with sharing class
MaintenanceRequestHelperTest {
3
4      private static final string STATUS_NEW = 'New';
5      private static final string WORKING =
'Working';
6      private static final string CLOSED = 'Closed';
7      private static final string REPAIR = 'Repair';

```

```
8     private static final string REQUEST_ORIGIN =  
    'Web';  
9     private static final string REQUEST_TYPE =  
    'Routine Maintenance';  
10    private static final string REQUEST_SUBJECT =  
    'Testing subject';  
11  
12    PRIVATE STATIC Vehicle__c createVehicle(){  
13        Vehicle__c Vehicle = new Vehicle__C(name =  
    'SuperTruck');  
14        return Vehicle;  
15    }  
16  
17    PRIVATE STATIC Product2 createEq(){  
18        product2 equipment = new product2(name =  
    'SuperEquipment',  
19    lifespan_months__C = 10,  
20    maintenance_cycle__C = 10,  
21    replacement_part__c = true);  
22        return equipment;  
23    }  
24  
25    PRIVATE STATIC Case createMaintenanceRequest(id  
    vehicleId, id equipmentId){  
26        case cs = new case(Type=REPAIR,  
27            Status=STATUS_NEW,  
28            Origin=REQUEST_ORIGIN,  
29            Subject=REQUEST_SUBJECT,
```

```
30         Equipment__c=equipmentId,
31         Vehicle__c=vehicleId);
32     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];  
70  
71      system.assert(workPart != null);  
72      system.assert(newReq.Subject != null);  
73      system.assertEquals(newReq.Type,  
      REQUEST_TYPE);  
74      SYSTEM.assertEquals(newReq.Equipment__c,  
      equipmentId);  
75      SYSTEM.assertEquals(newReq.Vehicle__c,  
      vehicleId);  
76
```



```

        SYSTEM.assertEquals(newReq.Date_Reported__c,
        system.today());
77     }
78
79     @istest
80     private static void
        testMaintenanceRequestNegative(){
81         Vehicle__C vehicle = createVehicle();
82         insert vehicle;
83         id vehicleId = vehicle.Id;
84
85         product2 equipment = createEq();
86         insert equipment;
87         id equipmentId = equipment.Id;
88
89         case emptyReq =
            createMaintenanceRequest(vehicleId,equipmentId);
90         insert emptyReq;
91
92         Equipment_Maintenance_Item__c workP =
            createWorkPart(equipmentId, emptyReq.Id);
93         insert workP;
94
95         test.startTest();
96         emptyReq.Status = WORKING;
97         update emptyReq;
98         test.stopTest();
99
100         list<case> allRequest = [select id
101                                 from case];
102
103         Equipment_Maintenance_Item__c workPart =

```

```
    [select id
104
    from Equipment_Maintenance_Item__c
105
    where Maintenance_Request__c = :emptyReq.Id];
106
107     system.assert(workPart != null);
108     system.assert(allRequest.size() == 1);
109 }
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(vehicleLis

128     }
129     insert requestList;
130
131     for(integer i = 0; i < 300; i++){
132
133         workPartList.add(createWorkPart(equipmentList.get(i)
134             .id, requestList.get(i).id));
135     }
136     insert workPartList;
137
138     test.startTest();
139     for(case req : requestList){
140         req.Status = CLOSED;
141         oldRequestIds.add(req.Id);
142     }
143     update requestList;
144     test.stopTest();
145
146     list<case> allRequests = [select id
147         from case
148         where status =:
149         STATUS_NEW];
150
151     list<Equipment_Maintenance_Item__c>
152     workParts = [select id
153         from Equipment_Maintenance_Item__c
154         where Maintenance_Request__c in: oldRequestIds];

```

```
151
152     system.assert(allRequests.size() == 300);
153 }
154 }
```

step 6 : Test callout logic :

Modify the apex classes as below save and run all

WarehouseCalloutServiceTest

```
1 @isTest
2
3 private class WarehouseCalloutServiceTest {
4     @isTest
5     static void testWareHouseCallout(){
6         Test.startTest();
7         // implement mock callout test here
8         Test.setMock(HTTPCalloutMock.class, new
WarehouseCalloutServiceMock());
9
WarehouseCalloutService.runWarehouseEquipmentSync(
);
10         Test.stopTest();
11         System.assertEquals(1, [SELECT count() FROM
Product2]);
12     }
13 }
```

WarehouseCalloutServiceMock

```

1 @isTest
2 global class WarehouseCalloutServiceMock implements
   HttpCalloutMock {
3     // implement http mock callout
4     global static HttpResponse respond(HttpRequest
   request){
5
6         System.assertEquals('https://th-superbadge-
   request.getEndpoint());
7         System.assertEquals('GET',
   request.getMethod());
8
9         // Create a fake response
10        HttpResponse response = new HttpResponse();
11        response.setHeader('Content-Type',
   'application/json');
12        response.setBody(' [{"_id": "55d66226726b611100aaf741
13
14        response.setStatusCode(200);
15        return response;
16    }

```

step 7 : Test Scheduling logic :

Modify the apex classes as below save and run all

WarehouseSyncSchedule :

```
1 global class WarehouseSyncSchedule implements
  Schedulable {
2     global void execute(SchedulableContext ctx) {
3
4
5     WarehouseCalloutService.runWarehouseEquipmentSync(
6     );
7     }
8 }
```

WarehouseSyncScheduleTest :

```
1 @isTest
2 public class WarehouseSyncScheduleTest {
3
4     @isTest static void WarehouseSyncScheduleTest(){
5         String scheduleTime = '00 00 01 * * ?';
6         Test.startTest();
7         Test.setMock(HttpCalloutMock.class, new
8         WarehouseCalloutServiceMock());
9         String jobID=System.schedule('Warehouse Time To Schedule
10
11         Test.stopTest();
12         //Contains schedule information for a scheduled job.
13         CronTrigger is similar to a cron job on UNIX systems.
14         // This object is available in API version 17.0 and
15         later.
16         CronTrigger a=[SELECT Id FROM CronTrigger where
17         NextFireTime > today];
18         System.assertEquals(jobID, a.Id,'Schedule ');
19
20     }
```

```
16     }
```

```
17 }
```