

# APEX SPECIALISTSUPERBADGE

## AUTOMATE RECORD CREATION:

### 1. MaintenanceRequest.apxt

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

```

25 for(Case cc : closedCasesM.values()){ Case nc = new Case (
26
27 ParentId = cc.Id, Status = 'New',
28 Subject = 'Routine Maintenance', Type = 'Routine Maintenance',
   Vehicle c = cc.Vehicle c, Equipment c =cc.Equipment c, Origin =
   'Web',
29 Date_Reported c = Date.Today()
30 );
31 If (maintenanceCycles.containsKey(cc.Id)){
32
33 nc.Date_Due c = Date.today().addDays((Integer)
   maintenanceCycles.get(cc.Id));
34
35 }
36 newCases.add(nc);
37 }
38
39 insert newCases;
40
41 List<Equipment_Maintenance_Item c> clonedWPs = new
   List<Equipment_Maintenance_Item c>();
42
43 for (Case nc : newCases){
44
45 for (Equipment_Maintenance_Item c wp :
   closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items r){
46
47 Equipment_Maintenance_Item c wpClone = wp.clone();
   wpClone.Maintenance_Request c = nc.Id;
48 ClonedWPs.add(wpClone);
49 }
50 }
51 insert ClonedWPs;
52 }
53 }
54 }

```

## **2. MaintenanceRequestHelper.apxc**

```

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

```

```

32
33 Maintenance_Request c = requestId);
34 return wp;
35
36 }
37
38 @istest
39 private static void testMaintenanceRequestPositive(){
40
41 Vehicle c vehicle = createVehicle(); insert vehicle;
42 id vehicleId = vehicle.Id;
43
44 Product2 equipment = createEq(); insert equipment;
45 id equipmentId = equipment.Id;
46 case somethingToUpdate =
    createMaintenanceRequest(vehicleId,equipmentId);
47
48 insert somethingToUpdate; Equipment_Maintenance_Item c workP =
49 createWorkPart(equipmentId,somethingToUpdate.id);
50
51 insert workP; test.startTest();
52
53 somethingToUpdate.status = CLOSED; update somethingToUpdate;
54
55
56 test.stopTest();
57 Case newReq = [Select id, subject, type, Equipment c,
    Date_Reported c, Vehicle c,
58 Date_Due c
59
60 from case
61
62 where status =:STATUS_NEW];
63
64
65 Equipment_Maintenance_Item c workPart = [select id
66
67 from Equipment_Maintenance_Item c
68
69 where Maintenance_Request c =:newReq.Id];

```

```

70
71 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);
72 SYSTEM.assertEquals(newReq.Date_Reported c, system.today());
73
74 }
75 @istest
76 private static void testMaintenanceRequestNegative(){ Vehicle C
    vehicle = createVehicle();
77 insert vehicle;
78 id vehicleId = vehicle.Id;
79 product2 equipment = createEq();
80
81 insert equipment;
82 id equipmentId = equipment.Id;
83 case emptyReq = createMaintenanceRequest(vehicleId,equipmentId);
84
85 insert emptyReq;
86 Equipment_Maintenance_Item c workP = createWorkPart(equipmentId,
    emptyReq.Id);
87
88 insert workP; test.startTest();
89
90 emptyReq.Status = WORKING; update emptyReq; test.stopTest();
91
92 list<case> allRequest = [select id from case];
93
94 Equipment_Maintenance_Item c workPart = [select id
95
96 from Equipment_Maintenance_Item c
97
98 where Maintenance_Request c = :emptyReq.Id];
99
100
101 system.assert(workPart != null); system.assert(allRequest.size()
    == 1);
102 }
103

```

```

104 @istest
105 private static void testMaintenanceRequestBulk(){
106
107 list<Vehicle C> vehicleList = new list<Vehicle C>();
    list<Product2> equipmentList = new list<Product2>();
    list<Equipment_Maintenance_Item c> workPartList = new
108 list<Equipment_Maintenance_Item c>(); list<case> requestList =
    new list<case>(); list<id> oldRequestIds = new list<id>();
109
110 for(integer i = 0; i < 300; i++){
111
112 vehicleList.add(createVehicle()); equipmentList.add(createEq());
113 }
114 insert vehicleList; insert equipmentList;
115 for(integer i = 0; i < 300; i++){
    requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
    equipmentList.get(i).id));
116 }
117
118 insert requestList;
119 for(integer i = 0; i < 300; i++){
    workPartList.add(createWorkPart(equipmentList.get(i).id,
    requestList.get(i).id));
120 }
121
122 insert workPartList;
123 test.startTest();
124
125 for(case req : requestList){ req.Status = CLOSED;
    oldRequestIds.add(req.Id);
126 }
127
128 update requestList; test.stopTest();
129
130 list<case> allRequests = [select id from case where status =:
    STATUS_NEW]; list<Equipment_Maintenance_Item c> workParts =
    [select id from
131 Equipment_Maintenance_Item c
132
133 where Maintenance_Request c in: oldRequestIds];

```

```
134
135 system.assert(allRequests.size() == 300);
136
137 }
138 }
139
```

## **\*SYNCHRONIZATION SALESFORCEDATA WITH AN EXTERNAL SYSTEM:**

### **1. WarehouseCalloutService.apxc**

```
1 public with sharing class WarehouseCalloutService {
2
3 private static final String WAREHOUSE_URL = 'https:// th-
4
```

```

5 / @future(callout=true)
6 public static void runWarehouseEquipmentSync(){
7
8     Http http = new Http();
9     HttpRequest request = new HttpRequest();
    request.setEndpoint(WAREHOUSE_URL); request.setMethod('GET');

10     HttpResponse response = http.send(request); List<Product2>
    warehouseEq = new List<Product2>();

11     if (response.getStatusCode() == 200){ List<Object>
    jsonResponse =

12         (List<Object>)JSON.deserializeUntyped(response.getBody());
13         System.debug(response.getBody());
14
15         for (Object eq : jsonResponse){
16             Map<String,Object> mapJson = (Map<String,Object>)eq;
            Product2 myEq = new Product2();

17             myEq.Replacement_Part c = (Boolean)
            mapJson.get('replacement'); myEq.Name = (String)

            mapJson.get('name');

18             myEq.Maintenance_Cycle c = (Integer)
            mapJson.get('maintenanceperiod'); myEq.Lifespan_Months c =

            (Integer) mapJson.get('lifespan');

```



```

19      myEq.Cost c = (Decimal) mapJson.get('lifespan');
      myEq.Warehouse_SKU c = (String) mapJson.get('sku');

20
21      myEq.Current_Inventory c = (Double)
      mapJson.get('quantity'); warehouseEq.add(myEq);

22
23    }

24
25      if (warehouseEq.size() > 0){ upsert warehouseEq;
26      System.debug('Your equipment was synced with the warehouse

27    }
28  }
29 }
30 }
31

```

## **\*SCHEDULE SYNCHRONIZATION USING APEX CODE:**

### **1. WarehouseSyncSchedule.apxc**

```

1 global class WarehouseSyncSchedule implements
  Schedulable { global void
  execute(SchedulableContext ctx) {

```

```
2 WarehouseCalloutService.runWarehouseEquipmentSync(  
    );  
3 }  
4 }  
5
```

## \*TEST AUTOMATION LOGIC:

### 1. MaintenanceRequestHelperTest.apxc

```
1 @istest
2 public with sharing class MaintenanceRequestHelperTest {
3
4     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';
5
6     private static final string REQUEST_ORIGIN = 'Web';
7
8     private static final string REQUEST_TYPE = 'Routine
        OBJECT =
        'Testing subject';
9
10
11 }
12 PRIVATE STATIC Product2 createEq(){
```

```
13 product2 equipment = new product2(name = 'SuperEquipment',
    lifespan_months C = 10, maintenance_cycle C = 10,
14
15 replacement_part c = true);
16 return equipment;
17 }
18
19 PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id
    equipmentId){
20
21 case cs = new case(Type=REPAIR,Status=STATUS_NEW,
    Origin=REQUEST_ORIGIN,Subject=REQUEST_SUBJECT,
22 Equipment c=equipmentId,Vehicle c=vehicleId);
23
24 return cs;
25 }
26 PRIVATE STATIC Equipment_Maintenance_Item c createWorkPart(id
    equipmentId,id requestId){
27 Equipment_Maintenance_Item c wp = new
    Equipment_Maintenance_Item c(Equipment c = equipmentId,
28 Maintenance_Request c = requestId);
29
30 return wp;
```

```
31 }

32 @istest

33 private static void testMaintenanceRequestPositive(){ Vehicle
    c vehicle = createVehicle();

34 insert vehicle;

35 id vehicleId = vehicle.Id;

36

37 Product2 equipment = createEq(); insert equipment;

38

39 id equipmentId = equipment.Id;

40

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

42

43 Equipment_Maintenance_Item c workP =
    createWorkPart(equipmentId,somethingToUpdate.id);

44 insert workP; test.startTest();

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

46

47

48
```

```
49 Case newReq = [Select id, subject, type, Equipment c,
    Date_Reported c, Vehicle c, Date_Due c from case where status
    =:STATUS_NEW];

50

51 Equipment_Maintenance_Item c workPart = [select id from
    Equipment_Maintenance_Item c where Maintenance_Request c
    =:newReq.Id];

52 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);

53 SYSTEM.assertEquals(newReq.Date_Reported c, system.today());

54

55 }

56

57 @istest

58 private static void testMaintenanceRequestNegative(){

59

60 Vehicle C vehicle = createVehicle(); insert vehicle;

61 id vehicleId = vehicle.Id;

62

63 product2 equipment = createEq(); insert equipment;

64 id equipmentId = equipment.Id;

65
```

```

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

67

68 Equipment_Maintenance_Item c workP =
    createWorkPart(equipmentId, emptyReq.Id); insert workP;

69

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

71 list<case> allRequest = [select id from case];
    Equipment_Maintenance_Item c workPart = [select id from

72 Equipment_Maintenance_Item c where Maintenance_Request c =
    :emptyReq.Id];

73

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

75 }

76 @istest

77 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

78 list<Equipment_Maintenance_Item c>(); list<case> requestList
    = new list<case>(); list<id> oldRequestIds = new list<id>();

79

80 for(integer i = 0; i < 300; i++){

```

```
81
82 vehicleList.add(createVehicle());
   equipmentList.add(createEq());
83 }
84 insert vehicleList; insert equipmentList;
85
86 for(integer i = 0; i < 300; i++){
87
88 requestList.add(createMaintenanceRequest(vehicleList.get(i).i
89 }
90
91 insert requestList; for(integer i = 0; i < 300; i++){
92 workPartList.add(createWorkPart(equipmentList.get(i).id,
   requestList.get(i).id));
93
94 }
95
96 insert workPartList; test.startTest();
97
98 for(case req : requestList){
99
```



```
100 req.Status = CLOSED; oldRequestIds.add(req.Id);
101
102 }
103
104 update requestList; test.stopTest();
105
106 list<case> allRequests = [select id from case where status
    =: STATUS_NEW];
107
108 list<Equipment_Maintenance_Item c> workParts = [select id
    from Equipment_Maintenance_Item c where Maintenance_Request c
    in: oldRequestIds];
109
110 system.assert(allRequests.size() == 300);
111 }
112 }
```

## 2. MaintenanceRequestHelper.apxc

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

```

24
25
26 AggregateResult[] results = [SELECT Maintenance_Request c,
27 MIN(Equipment r.Maintenance_Cycle c)cycle FROM
    Equipment_Maintenance_Item c WHERE Maintenance_Request c IN
    :ValidIds GROUP BY Maintenance_Request c];
28
29 for (AggregateResult ar : results){
30
31 maintenanceCycles.put((Id) ar.get('Maintenance_Request c'),
    (Decimal) ar.get('cycle'));
32
33 }
34 for(Case cc : closedCasesM.values()){ Case nc = new Case (
35 ParentId = cc.Id, Status = 'New',
36 Subject = 'Routine Maintenance', Type = 'Routine Maintenance',
    Vehicle c = cc.Vehicle c, Equipment c =cc.Equipment c, Origin =
    'Web',
37 Date_Reported c = Date.Today()
38
39 );
40
41 If (maintenanceCycles.containsKey(cc.Id)){
42
43 nc.Date_Due c = Date.today().addDays((Integer)
    maintenanceCycles.get(cc.Id));
44
45 }
46
47
48
49 newCases.add(nc);
50
51 }
52
53 insert newCases;
54
55 List<Equipment_Maintenance_Item c> clonedWPs = new
    List<Equipment_Maintenance_Item c>();
56

```

```

57 for (Case nc : newCases){
58
59 for (Equipment_Maintenance_Item c wp :
    closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items r){
60
61 Equipment_Maintenance_Item c wpClone = wp.clone();
    wpClone.Maintenance_Request c = nc.Id; ClonedWPs.add(wpClone);
62
63
64 }
65
66 }
67
68 insert ClonedWPs;
69
70 }
71
72 }
73 }
74

```

### 3. MaintenanceRequest.apxt

```

1 trigger MaintenanceRequest on Case (before update, after update)
  { if(Trigger.isUpdate && Trigger.isAfter){
2 MaintenanceRequestHelper.updateWorkOrders(Trigger.New,
    Trigger.OldMap);
3
4 }
5
6 }
7

```

## \*TEST CALLOUTLOGIC:

### 1. WarehouseCalloutService.apxc

```
1 public with sharing class WarehouseCalloutService {
2     private static final String WAREHOUSE_URL = 'https:// th-
3 / @future(callout=true)
4
5     public static void runWarehouseEquipmentSync(){
6
7         Http http = new Http();
8
9         HttpRequest request = new HttpRequest();
10        request.setEndpoint(WAREHOUSE_URL); request.setMethod('GET');
11        HttpResponse response = http.send(request);
12        List<Product2> warehouseEq = new List<Product2>();
13        if (response.getStatusCode() == 200){ List<Object>
            jsonResponse =
14        (List<Object>)JSON.deserializeUntyped(response.getBody());
```

```
System.debug(response.getBody());

15

16 for (Object eq : jsonResponse){

17 Map<String,Object> mapJson = (Map<String,Object>)eq; Product2
    myEq = new Product2();

18 myEq.Replacement_Part c = (Boolean)
    mapJson.get('replacement'); myEq.Name = (String)
    mapJson.get('name');

19 myEq.Maintenance_Cycle c = (Integer)
    mapJson.get('maintenanceperiod'); myEq.Lifespan_Months c =
    (Integer) mapJson.get('lifespan');

20 myEq.Cost c = (Decimal) mapJson.get('lifespan');
    myEq.Warehouse_SKU c = (String) mapJson.get('sku');

21

22 myEq.Current_Inventory c = (Double) mapJson.get('quantity');
    warehouseEq.add(myEq);

23 }

24

25 if (warehouseEq.size() > 0){ upsert warehouseEq;

26 System.debug('Your equipment was synced with the warehouse

27 }

28

29 }

30 }
```

```
31 }
```

```
32
```

## 2. WarehouseCalloutServiceTest.apxc

```
1  @isTest
2
3
4
5  private class WarehouseCalloutServiceTest { @isTest
6  static void testWareHouseCallout(){ Test.startTest();
7  / implement mock callout test here
8
9  Test.setMock(HTTPCalloutMock.class, new
    WarehouseCalloutServiceMock());
    WarehouseCalloutService.runWarehouseEquipmentSync();
10 Test.stopTest();
11
12 System.assertEquals(1, [SELECT count() FROM Product2]);
13
14 }
15
16 }
17
```

## 3. WarehouseCalloutServiceMock.apxc

```
1  @isTest
2  global class WarehouseCalloutServiceMock implements
    HttpCalloutMock {
```

```
3
4 / implement http mock callout
5
6 global static HttpResponseMessage respond(HttpRequest request){
7
8     System.assertEquals('https:// th-superbadge-
        ndpoint());
9
10    System.assertEquals('GET', request.getMethod());
11
12    / Create a fake response
13
14    HttpResponseMessage response = new HttpResponseMessage();
        response.setHeader('Content-Type', 'application/json');
15
16    response.SetBody('["_id":"55d66226726b611100aaf741","replacement

17 response.StatusCode(200); return response;
18 }
19
20 }
```



## **\*TEST SCHEDULING LOGIC:**

### **1. WarehouseSyncSchedule.apxc**

```
1 global class WarehouseSyncSchedule implements Schedulable {
    global void execute(SchedulableContext ctx) {

2

3 WarehouseCalloutService.runWarehouseEquipmentSync();

4 }

5 }
```

### **2. WarehouseSyncScheduleTest.apxc**

```
1 @isTest
2 public class WarehouseSyncScheduleTest {
3
4 @isTest static void WarehousescheduleTest(){ String scheduleTime
    = '00 00 01 * * ?'; Test.startTest();
5 Test.setMock(HttpCalloutMock.class, new
    WarehouseCalloutServiceMock());
6 String jobID=System.schedule('Warehouse Time To Schedule to

7 Test.stopTest();
8 / Contains schedule information for a scheduled job. CronTrigger
    is similar to a cron job on UNIX systems.
9 / This object is available in API version 17.0 and later.
10 CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime >
    today]; System.assertEquals(jobID, a.Id,'Schedule ');
11 }
```

