

1. Apex Triggers

1.1. Get Started with Apex Triggers:

AccountAddressTrigger.apxt:

```
1 trigger AccountAddressTrigger on Account (before insert,  
  before update) {  
2  
3     for(Account account:Trigger.New){  
4         if(account.Match_Billing_Address__c == True){  
5             account.ShippingPostalCode =  
              account.BillingPostalCode;  
6         }  
7     }  
8 }
```

1.2. Bulk Apex Triggers:

ClosedOpportunityTrigger.apxt:

```
1 trigger ClosedOpportunityTrigger on Opportunity (after  
  insert, after update) {  
2     list<Task> tasklist = new List<Task>();  
3  
4     for(Opportunity opp: Trigger.New){  
5         if(opp.StageName == 'Closed Won'){  
6             tasklist.add(new Task(Subject = 'Follow Up  
7  
8         }  
9  
10    if(tasklist.size()>0){  
11        insert tasklist;  
12    }  
13}
```

2. Apex Testing

2.1. Get Started with Apex Unit Tests:

VerifyDate.apxc:

```
1 public class VerifyDate {
2
3
4     public static Date CheckDates(Date date1, Date date2) {
5
6         if(DateWithin30Days(date1,date2)) {
7             return date2;
8         } else {
9             return SetEndOfMonthDate(date1);
10        }
11    }
12
13
14    private static Boolean DateWithin30Days(Date date1, Date
    date2) {
15
16        if( date2 < date1) { return false; }
17
18
19        Date date30Days = date1.addDays(30);
20        if( date2 >= date30Days ) { return false; }
21        else { return true; }
22    }
23
24
25    private static Date SetEndOfMonthDate(Date date1) {
26        Integer totalDays = Date.daysInMonth(date1.year(),
    date1.month());
27        Date lastDay = Date.newInstance(date1.year(),
```

```
    date1.month(), totalDays);
28     return lastDay;
29 }
30 }
```

TestVerifyDate.apxc:

```
1  @isTest
2  private class TestVerifyDate {
3
4
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
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
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 }
```

2.2.Test Apex Triggers:

RestrictContactByName.apxt:

```
1 trigger RestrictContactByName on Contact (before insert,  
  before update) {  
2   For (Contact c : Trigger.New) {  
3       if(c.LastName == 'INVALIDNAME') {  
4           c.AddError('The Last Name "'+c.LastName+'" is  
  
5       }  
6   }  
7 }
```

TestRestrictContactByname.apxc:

```
1 @isTest  
2 public class TestRestrictContactByname {  
3  
4     @isTest static void  
    TestContactWithInvalidNameNotInserted(){  
5  
6  
7         String inputLastName = 'INVALIDNAME';  
8         Contact newContact = new Contact(LastName=  
    inputLastName);  
9  
10  
11        Test.startTest();  
12        try{  
13            insert newContact;  
14        }  
15        catch (DmlException dmlEx) {  
16  
17            String expectedMessage = 'The Last Name "'+  
    newContact.LastName+'" is not allowed for DML';  
18            System.assertEquals(expectedMessage,  
    dmlEx.getDmlMessage(0));
```

```

19     }
20     Test.stopTest();
21 }
22 }

```

2.3 Create Test Data for Apex Tests

RandomContactFactory.apxc:

```

1 public class RandomContactFactory {
2     public static List<Contact> generateRandomContacts
      (Integer numOfCon, String ConLastName){
3         List<Contact> conList = new List<Contact>();
4         for(Integer i=1; i<numOfCon;i++){
5             conList.add(new Contact(FirstName='Test ' + i,
      LastName = ConLastName));
6         }
7         return conlist;
8     }
9 }

```

3. Asynchronous Apex

3.1. Use Future Methods:

AccountProcessor.apxc:

```

1 public class AccountProcessor
2 {
3     @future
4     public static void countContacts(Set<id> setId)
5     {
6         List<Account> lstAccount = [select id,Number_of_Contacts__c ,(select id from

```

```

        contacts ) from account where id in :setId ];
7     for( Account acc : lstAccount )
8     {
9         List<Contact> lstCont = acc.contacts ;
10
11         acc.Number_of_Contacts__c = lstCont.size();
12     }
13     update lstAccount;
14 }
15 }

```

AccountProcessorTest.apxc:

```

1  @IsTest
2  public class AccountProcessorTest {
3      public static testmethod void TestAccountProcessorTest()
4      {
5          Account a = new Account();
6          a.Name = 'Test Account';
7          Insert a;
8
9          Contact cont = New Contact();
10         cont.FirstName = 'Bob';
11         cont.LastName = 'Masters';
12         cont.AccountId = a.Id;
13         Insert cont;
14
15         set<Id> setAccId = new Set<ID>();
16         setAccId.add(a.id);
17
18         Test.startTest();
19         AccountProcessor.countContacts(setAccId);
20         Test.stopTest();
21
22         Account ACC = [select Number_of_Contacts__c from Account
23             where id = :a.id LIMIT 1];
24         System.assertEquals (
25             Integer.valueOf(ACC.Number_of_Contacts__c) ,1);
26     }
27 }

```

3.2 Use Batch Apex:

LeadProcessor.apxc:

```
1  global class LeadProcessor implements
    Database.Batchable<sObject>, Database.Stateful {
2
3
4      global Integer recordsProcessed = 0;
5
6
7      global Database.QueryLocator
    start(Database.BatchableContext bc) {
8          return Database.getQueryLocator([SELECT ID,
        LeadSource FROM Lead]);
9      }
10
11
12     global void execute(Database.BatchableContext bc,
        List<Lead> scope) {
13         for (Lead lead : scope) {
14             lead.LeadSource = 'Dreamforce';
15             recordsProcessed = recordsProcessed + 1;
16             System.debug(lead.LeadSource);
17         }
18         update scope;
19     }
20
21
22     global void finish(Database.BatchableContext bc){
23         System.debug(recordsProcessed + ' records
24     }
25}
```

LeadProcessorTest.apxc:

```
1  @isTest
2  private class LeadProcessorTest {
3
4
5      @TestSetup
6      static void setup(){
7          List<Lead> leads = new List<Lead>();
8
9          for (Integer i = 0; i < 200; i++) {
10
11              leads.add(new Lead(LastName='Lead ' + i,
12 Company='Company Number ' + i, Status='Open - Not Contacted'));
13          }
14
15          insert leads;
16      }
17
18      static testMethod void test() {
19
20          Test.startTest();
21          LeadProcessor lp = new LeadProcessor();
22          Id batchId = Database.executeBatch(lp);
23          Test.stopTest();
24
25
26          System.assertEquals(200, [select count() from lead where
27 LeadSource = 'Dreamforce']);
28      }
29 }
```

3.3. Control Processes with Queueable Apex

AddPrimaryContact.apxc

```
1  public class AddPrimaryContact implements Queueable {
```



```

2     public contact c;
3     public String state;
4
5     public AddPrimaryContact(Contact c, String state) {
6         this.c = c;
7         this.state = state;
8     }
9
10    public void execute(QueueableContext qc) {
11        system.debug('this.c = '+this.c+' this.state =
12
13        List<Account> acc_lst = new List<account>([select
14        id, name, BillingState from account where
15        account.BillingState = :this.state limit 200]);
16
17        List<contact> c_lst = new List<contact>();
18        for(account a: acc_lst) {
19            contact c = new contact();
20            c = this.c.clone(false, false, false, false);
21            c.AccountId = a.Id;
22            c_lst.add(c);
23        }
24    }
25}

```

AddPrimaryContactTest.apxc:

```

1 @IsTest
2 public class AddPrimaryContactTest {
3
4     @IsTest
5     public static void testing() {
6         List<account> acc_lst = new List<account>();
7         for (Integer i=0; i<50;i++) {
8             account a = new
9             account(name=string.valueOf(i),billingstate='NY');
10        }
11    }
12}

```

```

9         system.debug('account a = '+a);
10        acc_lst.add(a);
11    }
12    for (Integer i=0; i<50;i++) {
13        account a = new
account(name=string.valueOf(50+i),billingstate='CA');
14        system.debug('account a = '+a);
15        acc_lst.add(a);
16    }
17    insert acc_lst;
18    Test.startTest();
19    contact c = new contact(lastname='alex');
20    AddPrimaryContact apc = new
AddPrimaryContact(c,'CA');
21    system.debug('apc = '+apc);
22    System.enqueueJob(apc);
23    Test.stopTest();
24    List<contact> c_lst = new List<contact>([select id
from contact]);
25    Integer size = c_lst.size();
26    system.assertEquals(50, size);
27 }
28
29 }

```

3.4 Schedule Jobs Using the 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  @isTest
2  private class DailyLeadProcessorTest{
3
4      public static String CRON_EXP = '0 0 0 2 6 ? 2022';
5
6      static testmethod void testScheduledJob(){
7          List<Lead> leads = new List<Lead>();
8
9          for(Integer i = 0; i < 200; i++){
10              Lead lead = new Lead(LastName = 'Test ' + i,
11              LeadSource = '', Company = 'Test Company ' + i, Status =
12              'Open - Not Contacted');
13              leads.add(lead);
14          }
15      }
16  }

```

```

13
14     insert leads;
15
16     Test.startTest();
17
18     String jobId = System.schedule('Update LeadSource

19
20
21     Test.stopTest();
22 }
23 }

```

4. Apex Integration Services

4.1 Apex REST Callouts:

AnimalLocator.apxc:

```

1 public class AnimalLocator{
2     public static String getAnimalNameById(Integer x){
3         Http http = new Http();
4         HttpRequest req = new HttpRequest();
5         req.setEndpoint('https://th-apex-http-

6         req.setMethod('GET');
7         Map<String, Object> animal= new Map<String,
Object>();
8         HttpResponse res = http.send(req);
9         if (res.getStatusCode() == 200) {
10            Map<String, Object> results = (Map<String,
Object>)JSON.deserializeUntyped(res.getBody());

```

```

11     animal = (Map<String, Object>) results.get('animal');
12     }
13     return (String)animal.get('name');
14     }
15 }

```

AnimalLocatorTest.apxc:

```

1  @isTest
2  public class AnimalLocatorTest {
3      @isTest public static void AnimalLocatorMock() {
4          Test.setMock(HttpCalloutMock.class, new
AnimalLocatorMock());
5          String result = AnimalLocator.getAnimalNameById(1);
6          System.debug(result);
7          String expectedResult = 'chicken';
8          System.assertEquals(result,expectedResult );
9      }
10 }

```

AnimalLocatorMock.apxc:

```

1  @isTest
2  global class AnimalLocatorMock implements HttpCalloutMock {
3
4      global HTTPResponse respond(HTTPRequest request) {
5
6          HttpResponse response = new HttpResponse();
7          response.setHeader('Content-Type',
'application/json');
8
9          response.setBody('{"animal":{"id":1,"name":"chicken","eats"
10
11          response.setStatusCode(200);
12          return response;
11      }
12 }

```

4.2. Apex SOAP Callouts

ParkLocator.apxc:

```
1 public class ParkLocator {
2     public static String[] country(String country){
3         ParkService.ParksImplPort parks = new
4         ParkService.ParksImplPort();
5         String[] parksname = parks.byCountry(country);
6         return parksname;
7     }
8 }
```

ParkLocatorTest.apxc:

```
1 @isTest
2 private class ParkLocatorTest{
3     @isTest
4     static void testParkLocator() {
5         Test.setMock(WebServiceMock.class, new
6         ParkServiceMock());
7         String[] arrayOfParks =
8         ParkLocator.country('India');
9         System.assertEquals('Park1', arrayOfParks[0]);
10    }
```

ParkService.apxc:

```
1 public class ParkService {
2     public class byCountryResponse {
3         public String[] return_x;
4         private String[] return_x_type_info = new
5         String[]{'return','http://parks.services/',null,'0','-
```

```

5         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', 'fals

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',
39             'ParkService.byCountryResponse'}
40     );
41     response_x = response_map_x.get('response_x');
42     return response_x.return_x;
43 }
44 }
45 }

```

ParkServiceMock.apxc:

```

1  @isTest
2  global class ParkServiceMock implements WebServiceMock {
3      global void doInvoke(
4          Object stub,
5          Object request,
6          Map<String, Object> response,
7          String endpoint,
8          String soapAction,
9          String requestName,
10         String responseNS,
11         String responseName,
12         String responseType) {
13         ParkService.byCountryResponse response_x = new
14         ParkService.byCountryResponse();
15         List<String> lstOfDummyParks = new List<String>
16         {'Park1', 'Park2', 'Park3'};
17         response_x.return_x = lstOfDummyParks;
18
19         response.put('response_x', response_x);

```



```
18     }  
19 }
```

4.3 Apex Web Services:

AccountManager.apxc:

```
1  @RestResource(urlMapping='/Accounts/*/contacts')  
2  global with sharing class AccountManager{  
3      @HttpGet  
4      global static Account getAccount(){  
5          RestRequest req = RestContext.request;  
6          String accId =  
7          req.requestURI.substringBetween('Accounts/', '/contacts');  
8          Account acc = [SELECT Id, Name, (SELECT Id, Name FROM  
9              Contacts)  
10              FROM Account WHERE Id = :accId];  
11      }  
12 }
```

AccountManagerTest.apxc:

```
1  @IsTest  
2  private class AccountManagerTest{  
3      @isTest static void testAccountManager(){  
4          Id recordId = getTestAccountId();  
5  
6          RestRequest request = new RestRequest();  
7          request.requestUri =  
8  
9          'https://ap5.salesforce.com/services/apexrest/Accounts/' +  
10         recordId + '/contacts';  
11         request.httpMethod = 'GET';  
12         RestContext.request = request;  
13     }
```

```

12
13     Account acc = AccountManager.getAccount();
14
15
16     System.assert(acc != null);
17 }
18
19 private static Id getTestAccountId(){
20     Account acc = new Account(Name = 'TestAcc2');
21     Insert acc;
22
23     Contact con = new Contact(LastName = 'TestCont2',
    AccountId = acc.Id);
24     Insert con;
25
26     return acc.Id;
27 }
28 }

```

Apex Specialist

1. Automated Record Creation:

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     if (!validIds.isEmpty()){
15         List<Case> newCases = new List<Case>();
16         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)
17 FROM Case WHERE Id IN :validIds]);
18         Map<Id,Decimal> maintenanceCycles = new
Map<ID,Decimal>();
19         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];
20
21         for (AggregateResult ar : results){
22             maintenanceCycles.put((Id)
ar.get('Maintenance_Request__c'), (Decimal)
ar.get('cycle'));
23         }
24
25         for(Case cc : closedCasesM.values()){
26             Case nc = new Case (
27                 ParentId = cc.Id,
28                 Status = 'New',
29                 Subject = 'Routine Maintenance',
30                 Type = 'Routine Maintenance',
31                 Vehicle__c = cc.Vehicle__c,
32                 Equipment__c =cc.Equipment__c,

```

```
33             Origin = 'Web',
34             Date_Reported__c = Date.Today()
35
36         );
37
38         If (maintenanceCycles.containsKey(cc.Id)){
39             nc.Date_Due__c =
34             Date.today().addDays((Integer)
35             maintenanceCycles.get(cc.Id));
40         } else {
41             nc.Date_Due__c =
36             Date.today().addDays((Integer)
37             cc.Equipment__r.maintenance_Cycle__c);
42         }
43
44         newCases.add(nc);
45     }
46
47     insert newCases;
48
49     List<Equipment_Maintenance_Item__c> clonedWPs =
38     new List<Equipment_Maintenance_Item__c>();
50     for (Case nc : newCases){
51         for (Equipment_Maintenance_Item__c wp :
39         closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__
52
53         Equipment_Maintenance_Item__c wpClone =
34         wp.clone();
54         wpClone.Maintenance_Request__c = nc.Id;
55         ClonedWPs.add(wpClone);
56     }
57     insert ClonedWPs;
58 }
59 }
60 }
```

MaintenanceRequest.apxt:

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

2. Synchronize Salesforce data with an external system:

WarehouseCalloutService.apxc :

```
1    public with sharing class WarehouseCalloutService
    implements Queueable {
2        private static final String WAREHOUSE_URL =
        'https://th-superbadge-apex.herokuapp.com/equipment';
3
4        @future(callout=true)
5        public static void runWarehouseEquipmentSync(){
6            Http http = new Http();
7            HttpRequest request = new HttpRequest();
8
9            request.setEndpoint(WAREHOUSE_URL);
10           request.setMethod('GET');
11           HttpResponse response = http.send(request);
12
13           List<Product2> warehouseEq = new List<Product2>();
14
15           if (response.getStatusCode() == 200){
16               List<Object> jsonResponse =
                (List<Object>)JSON.deserializeUntyped(response.getBody());
17               System.debug(response.getBody());
18
19           }
```

```
20         for (Object eq : jsonResponse){
21             Map<String,Object> mapJson =
22             (Map<String,Object>)eq;
23             Product2 myEq = new Product2();
24             myEq.Replacement_Part__c = (Boolean)
25             mapJson.get('replacement');
26             myEq.Name = (String) mapJson.get('name');
27             myEq.Maintenance_Cycle__c = (Integer)
28             mapJson.get('maintenanceperiod');
29             myEq.Lifespan_Months__c = (Integer)
30             mapJson.get('lifespan');
31             myEq.Cost__c = (Integer)
32             mapJson.get('cost');
33             myEq.Warehouse_SKU__c = (String)
34             mapJson.get('sku');
35             myEq.Current_Inventory__c = (Double)
36             mapJson.get('quantity');
37             myEq.ProductCode = (String)
38             mapJson.get('_id');
39             warehouseEq.add(myEq);
40         }
41     }
42     if (warehouseEq.size() > 0){
43         upsert warehouseEq;
44         System.debug('Your equipment was synced
45
46
47     }
48 }

49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

3. Schedule synchronization using Apex code:

WarehouseSyncShedule.apxc :

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

4. Test automation logic:

MaintenanceRequestHelperTest.apxc :

```
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 =
15         'SuperTruck');
16         return Vehicle;
17     }
18 }
```

```

16
17     PRIVATE STATIC Product2 createEq(){
18         product2 equipment = new product2(name =
19             'SuperEquipment',
20                                     lifespan_months__C
21         = 10,
22         maintenance_cycle__C = 10,
23         replacement_part__c = true);
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(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 =:
147     STATUS_NEW];
148     list<Equipment_Maintenance_Item__c> workParts =
149     [select id
150     from Equipment_Maintenance_Item__c
151     where Maintenance_Request__c in: oldRequestIds];
152     system.assert(allRequests.size() == 300);
153 }
154 }

```

MaintenanceRequestHelper.apxc :

```

1 public with sharing class MaintenanceRequestHelper {
2     public static void updateWorkOrders(List<Case>
3     updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
4         Set<Id> validIds = new Set<Id>();
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 =
42         Date.today().addDays((Integer)
43         maintenanceCycles.get(cc.Id));
44     }
45     newCases.add(nc);
46
47     insert newCases;
48
49     List<Equipment_Maintenance_Item__c> clonedWPs =
50     new List<Equipment_Maintenance_Item__c>();
51     for (Case nc : newCases){
52         for (Equipment_Maintenance_Item__c wp :
53         closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__
54
55         Equipment_Maintenance_Item__c wpClone =
56         wp.clone();
57         wpClone.Maintenance_Request__c = nc.Id;
58         ClonedWPs.add(wpClone);
59     }
60     insert ClonedWPs;
61 }

```

```
61}
```

MaintenanceRequest.apxt :

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

5. Test callout logic:

WarehouseCalloutService.apxc :-

```
1 public with sharing class WarehouseCalloutService {
2
3     private static final String WAREHOUSE_URL =
4     'https://th-superbadge-apex.herokuapp.com/equipment';
5
6     public static void runWarehouseEquipmentSync(){
7
8         Http http = new Http();
9         HttpRequest request = new HttpRequest();
10
11         request.setEndpoint(WAREHOUSE_URL);
12         request.setMethod('GET');
13         HttpResponse response = http.send(request);
14
15 }
```

```

16         List<Product2> warehouseEq = new List<Product2>();
17
18         if (response.getStatusCode() == 200){
19             List<Object> jsonResponse =
20             (List<Object>)JSON.deserializeUntyped(response.getBody());
21             System.debug(response.getBody());
22
23             for (Object eq : jsonResponse){
24                 Map<String,Object> mapJson =
25                 (Map<String,Object>)eq;
26                 Product2 myEq = new Product2();
27                 myEq.Replacement_Part__c = (Boolean)
28                 mapJson.get('replacement');
29                 myEq.Name = (String) mapJson.get('name');
30                 myEq.Maintenance_Cycle__c = (Integer)
31                 mapJson.get('maintenanceperiod');
32                 myEq.Lifespan_Months__c = (Integer)
33                 mapJson.get('lifespan');
34                 myEq.Cost__c = (Decimal)
35                 mapJson.get('lifespan');
36                 myEq.Warehouse_SKU__c = (String)
37                 mapJson.get('sku');
38                 myEq.Current_Inventory__c = (Double)
39                 mapJson.get('quantity');
40                 warehouseEq.add(myEq);
41             }
42         }

```



```
43 }
```

WarehouseCalloutServiceTest.apxc :

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

WarehouseCalloutServiceMock.apxc :

```
1  @isTest
2  global class WarehouseCalloutServiceMock implements
HttpCalloutMock {
3
4      global static HttpResponse respond(HttpRequest
request){
5
6          System.assertEquals('https://th-superbadge-
));
7          System.assertEquals('GET', request.getMethod());
8
9
10         HttpResponse response = new HttpResponse();
11         response.setHeader('Content-Type',
```

```

        'application/json');
12     response.setBody(' [{"_id":"55d66226726b611100aaf741","repla

13         response.setStatusCode(200);
14         return response;
15     }
16 }

```

6. Test scheduling logic:

WarehouseSyncSchedule.apxc :

```

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

```

WarehouseSyncScheduleTest.apxc :

```

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

WarehouseSyncSchedule());

```

```
9         Test.stopTest();
10
11
12         CronTrigger a=[SELECT Id FROM CronTrigger where
    NextFireTime > today];
13         System.assertEquals(jobID, a.Id,'Schedule ');
14     }
15 }
```