Apex Triggers

AccountAddressTrigger.apxt :

```
trigger AccountAddressTrigger on Account (before insert, before update)
{
    For(Account accountAddress: Trigger.new){
        if(accountAddress.BillingPostalCode !=null &&
accountAddress.Match_Billing_Address__c ==true){

accountAddress.ShippingPostalCode=accountAddress.BillingPostalCode;
        }
    }

}
```

ClosedOpportunityTrigger.apxt :

```
trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {

    List<Task> taskList = new List<Task>();



    for(opportunity opp: Trigger.New){

        if(opp.StageName!=trigger.oldMap.get(opp.id).stageName)
        {

            taskList.add(new Task(Subject = 'Follow Up Test Task',
                                  WhatId = opp.Id));
        }

    }


    if(taskList.size()>0){

        insert taskList;

    }

}
```

Apex Testing

RandomContactFactory.apxc :

```
public class RandomContactFactory {
    public static List<Contact> generateRandomContacts(Integer count, String name) {
        List<Contact> contactList = new List<Contact>();

        for(Integer index = 1; index <= count; index++) {
            Contact c = new Contact();
            c.FirstName = name + index;
            contactList.add(c);
        }

        return contactList;
    }
}
```

RestrictContactByName.apxt :

```
  trigger RestrictContactByName on Contact (before insert, before update) {

  //check contacts prior to insert or update for invalid data
  For (Contact c : Trigger.New) {
        if(c.LastName == 'INVALIDNAME') {   //invalidname is invalid
              c.AddError('The Last Name "'+c.LastName+'" is not allowed

        }
  }

}
```

TestRestrictContactByName.apxc :

```
@isTest
private class TestRestrictContactByName {
  @isTest static void test1(){
    contact c = new Contact();
    c.LastName = 'INVALIDNAME';
```

```
        // Perform test
        Test.startTest();
        insert c;
        Test.stopTest();
    }
     @isTest static void test2(){
        contact c = new Contact();
        c.LastName = 'Devi';
        // Perform test
        Test.startTest();
        insert c;
        Test.stopTest();
    }

}
```

TestVerifyDate.apxc :

```
@isTest
public class TestVerifyDate {
    @isTest static void testOldDate(){
        Date dateTest = VerifyDate.CheckDates(date.today(), date.today().addDays(-1));
        System.assertEquals(date.newInstance(2016, 4, 30), dateTest);
    }

    @isTest static void testLessThan30Days(){
        Date dateTest = VerifyDate.CheckDates(date.today(), date.today().addDays(20));
        System.assertEquals(date.today().addDays(20), dateTest);
    }

    @isTest static void testMoreThan30Days(){
        Date dateTest = VerifyDate.CheckDates(date.today(), date.today().addDays(31));
        System.assertEquals(date.newInstance(2016, 4, 30), dateTest);
    }

}
```

VerifyDate.apxc :

```
public class VerifyDate {

 //method to handle potential checks against two dates
 public static Date CheckDates(Date date1, Date date2) {
 //if date2 is within the next 30 days of date1, use date2. Otherwise use the end of the month
 if(DateWithin30Days(date1,date2)) {
       return date2;
 } else {
       return SetEndOfMonthDate(date1);
 }
 }

 //method to check if date2 is within the next 30 days of date1
 private static Boolean DateWithin30Days(Date date1, Date date2) {
 //check for date2 being in the past
 if( date2 < date1) { return false; }

 //check that date2 is within (>=) 30 days of date1
 Date date30Days = date1.addDays(30); //create a date 30 days away from date1
 if( date2 >= date30Days ) { return false; }
 else { return true; }
 }

 //method to return the end of the month of a given date
 private static Date SetEndOfMonthDate(Date date1) {
 Integer totalDays = Date.daysInMonth(date1.year(), date1.month());
 Date lastDay = Date.newInstance(date1.year(), date1.month(), totalDays);
 return lastDay;
 }

}
```

AsynchronousApex

AccountProcessor.apxc :

```
public class AccountProcessor
{
  @future
  public static void countContacts(Set<id> setId)
  {
      List<Account> lstAccount = [select id,Number_of_Contacts__c
, (select id from contacts ) from account where id in :setId ];
      for( Account acc : lstAccount )
      {
          List<Contact> lstCont = acc.contacts ;

          acc.Number_of_Contacts__c = lstCont.size();
      }
      update lstAccount;
  }
}
```

AccountProcessorTest.apxc :

```
@IsTest
public class AccountProcessorTest {
    public static testmethod void TestAccountProcessorTest()
    {
        Account a = new Account();
        a.Name = 'Test Account';
        Insert a;

        Contact cont = New Contact();
        cont.FirstName ='Bob';
        cont.LastName ='Masters';
        cont.AccountId = a.Id;
        Insert cont;

        set<Id> setAccId = new Set<ID>();
        setAccId.add(a.id);

        Test.startTest();
            AccountProcessor.countContacts(setAccId);
        Test.stopTest();
```

```
21
22          Account ACC = [select Number_of_Contacts__c from Account where
    id = :a.id LIMIT 1];
23          System.assertEquals ( Integer.valueOf(ACC.Number_of_Contacts__c)
    ,1);
24    }
25
26 }
```

AddPrimaryContact.apxc :

```
1  public class AddPrimaryContact implements Queueable
2  {
3      private Contact c;
4      private String state;
5      public  AddPrimaryContact(Contact c, String state)
6      {
7          this.c = c;
8          this.state = state;
9      }
10     public void execute(QueueableContext context)
11     {
12          List<Account> ListAccount = [SELECT ID, Name ,(Select
    id,FirstName,LastName from contacts ) FROM ACCOUNT WHERE
    BillingState = :state LIMIT 200];
13          List<Contact> lstContact = new List<Contact>();
14          for (Account acc:ListAccount)
15          {
16                  Contact cont = c.clone(false,false,false,false);
17                  cont.AccountId =  acc.id;
18                  lstContact.add( cont );
19          }
20
21          if(lstContact.size() >0 )
22          {
23              insert lstContact;
24          }
25
26     }
27
28 }
```

AddPrimaryContactTest.apxc :

```
@isTest
public class AddPrimaryContactTest
{
     @isTest static void TestList()
     {
          List<Account> Teste = new List <Account>();
          for(Integer i=0;i<50;i++)
          {
              Teste.add(new Account(BillingState = 'CA', name =
'Test'+i));
          }
          for(Integer j=0;j<50;j++)
          {
              Teste.add(new Account(BillingState = 'NY', name =
'Test'+j));
          }
          insert Teste;

          Contact co = new Contact();
          co.FirstName='demo';
          co.LastName ='demo';
          insert co;
          String state = 'CA';

           AddPrimaryContact apc = new AddPrimaryContact(co,
state);
           Test.startTest();
             System.enqueueJob(apc);
           Test.stopTest();
      }
 }
```

DailyLeadProcessor.apxc :

```
global class DailyLeadProcessor implements Schedulable {

    global void execute(SchedulableContext ctx) {
        List<Lead> lList = [Select Id, LeadSource from Lead where
LeadSource = null];
```

```
5
6           if(!lList.isEmpty()) {
7      for(Lead l: lList) {
8       l.LeadSource = 'Dreamforce';
9      }
10     update lList;
11    }
12      }
13 }
```

DailyLeadProcessorTest.apxc :

```
1  @isTest
2  private class DailyLeadProcessorTest {
3    static testMethod void testDailyLeadProcessor() {
4          String CRON_EXP = '0 0 1 * * ?';
5          List<Lead> lList = new List<Lead>();
6        for (Integer i = 0; i < 200; i++) {
7                lList.add(new Lead(LastName='Dreamforce'+i,
   Company='Test1 Inc.', Status='Open - Not Contacted'));
8          }
9          insert lList;
10
11         Test.startTest();
12         String jobId = System.schedule('DailyLeadProcessor',
   CRON_EXP, new DailyLeadProcessor());
13   }
14 }
```

LeadProcessor.apxc :

```
1  global class LeadProcessor implements
2  Database.Batchable<sObject>, Database.Stateful {
3
4      // instance member to retain state across transactions
5      global Integer recordsProcessed = 0;
6
7      global Database.QueryLocator start(Database.BatchableContext bc) {
8          return Database.getQueryLocator('SELECT Id, LeadSource FROM

9      }
10
```

```
    global void execute(Database.BatchableContext bc, List<Lead> scope){
        // process each batch of records
        List<Lead> leads = new List<Lead>();
        for (Lead lead : scope) {

                lead.LeadSource = 'Dreamforce';
                // increment the instance member counter
                recordsProcessed = recordsProcessed + 1;

        }
        update leads;
    }

    global void finish(Database.BatchableContext bc){
        System.debug(recordsProcessed + ' records processed. Shazam!');

    }
}
```

LeadProcessorTest.apxc :

```
@isTest
public class LeadProcessorTest {
 @testSetup
    static void setup() {
        List<Lead> leads = new List<Lead>();
        // insert 200 leads
        for (Integer i=0;i<200;i++) {
            leads.add(new Lead(LastName='Lead '+i,
                Company='Lead', Status='Open - Not Contacted'));
        }
        insert leads;
    }

    static testmethod void test() {
        Test.startTest();
        LeadProcessor lp = new LeadProcessor();
        Id batchId = Database.executeBatch(lp, 200);
        Test.stopTest();

        // after the testing stops, assert records were updated
properly
```

```
        System.assertEquals(200, [select count() from lead where
    LeadSource = 'Dreamforce']);
    }
}
```

Apex Integration Services

AccountManager.apxc :

```
@RestResource(urlMapping='/Accounts/*/contacts')
global with sharing class AccountManager{
    @HttpGet
    global static Account getAccount(){
        RestRequest req = RestContext.request;
        String accId =
    req.requestURI.substringBetween('Accounts/', '/contacts');
        Account acc = [SELECT Id, Name, (SELECT Id, Name FROM
    Contacts)
                       FROM Account WHERE Id = :accId];

        return acc;
    }
}
```

AccountManagerTest.apxc :

```
@IsTest
private class AccountManagerTest{
    @isTest static void testAccountManager(){
        Id recordId = getTestAccountId();
        // Set up a test request
        RestRequest request = new RestRequest();
        request.requestUri =

    'https://ap5.salesforce.com/services/apexrest/Accounts/'+
    recordId +'/contacts';
        request.httpMethod = 'GET';
        RestContext.request = request;
```

```
11
12          // Call the method to test
13          Account  acc = AccountManager.getAccount();
14
15          // Verify results
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  }
```

AnimalCallouts.apxc :

```
1   public class AnimalsCallouts {
2       public static HttpResponse makeGetCallout() {
3           Http http = new Http();
4           HttpRequest request = new HttpRequest();
5           request.setEndpoint('https://th-apex-http-

6           request.setMethod('GET');
7           HttpResponse response = http.send(request);
8           // If the request is successful, parse the JSON response.
9           if(response.getStatusCode() == 200) {
10              // Deserializes the JSON string into collections of
    primitive data types.
11              Map<String, Object> results = (Map<String, Object>)
    JSON.deserializeUntyped(response.getBody());
12              // Cast the values in the 'animals' key as a list
13              List<Object> animals = (List<Object>)
    results.get('animals');
14              System.debug('Received the following animals:');
15              for(Object animal: animals) {
16                  System.debug(animal);
```

```
            }
        }
        return response;
    }
    public static HttpResponse makePostCallout() {
        Http http = new Http();
        HttpRequest request = new HttpRequest();
        request.setEndpoint('https://th-apex-http-

        request.setMethod('POST');
       request.setHeader('Content-Type', 'application/json;charset=UTF-8');
        request.setBody('{"name":"mighty moose"}');
        HttpResponse response = http.send(request);
        // Parse the JSON response
        if(response.getStatusCode() != 201) {
            System.debug('The status code returned was not expected: ' +
                response.getStatusCode() + ' ' + response.getStatus());
        } else {
            System.debug(response.getBody());
        }
        return response;
    }
}
```

AnimalCalloutsTest.apxc :

```
@isTest
private class AnimalsCalloutsTest {
    @isTest static  void testGetCallout() {
        // Create the mock response based on a static resource
        StaticResourceCalloutMock mock = new
StaticResourceCalloutMock();
        mock.setStaticResource('GetAnimalResource');
        mock.setStatusCode(200);
        mock.setHeader('Content-Type',
'application/json;charset=UTF-8');
        // Associate the callout with a mock response
        Test.setMock(HttpCalloutMock.class, mock);
        // Call method to test
        HttpResponse result = AnimalsCallouts.makeGetCallout();
        // Verify mock response is not null
        System.assertNotEquals(null,result, 'The callout returned
```

```
        // Verify status code
        System.assertEquals(200,result.getStatusCode(), 'The

        // Verify content type
        System.assertEquals('application/json;charset=UTF-8',
          result.getHeader('Content-Type'),
          'The content type value is not expected.');
        // Verify the array contains 3 items
        Map<String, Object> results = (Map<String, Object>)
            JSON.deserializeUntyped(result.getBody());
        List<Object> animals = (List<Object>)
results.get('animals');
        System.assertEquals(3, animals.size(), 'The array should

    }
    @isTest
static void testPostCallout() {
    // Set mock callout class
    Test.setMock(HttpCalloutMock.class, new
AnimalsHttpCalloutMock());
    // This causes a fake response to be sent
    // from the class that implements HttpCalloutMock.
    HttpResponse response = AnimalsCallouts.makePostCallout();
    // Verify that the response received contains fake values
    String contentType = response.getHeader('Content-Type');
    System.assert(contentType == 'application/json');
    String actualValue = response.getBody();
    System.debug(response.getBody());
    String expectedValue = '{"animals": ["majestic badger",

    System.assertEquals(expectedValue, actualValue);
    System.assertEquals(200, response.getStatusCode());
}
}
```

AnimalLocator.apxc :

```
public class AnimalLocator
```

```
2  {
3
4    public static String getAnimalNameById(Integer id)
5     {
6          Http http = new Http();
7          HttpRequest request = new HttpRequest();
8          request.setEndpoint('https://th-apex-http-

9          request.setMethod('GET');
10         HttpResponse response = http.send(request);
11           String strResp = '';
12            system.debug('******response

13            system.debug('******response '+response.getBody());
14         // If the request is successful, parse the JSON response.
15         if (response.getStatusCode() == 200)
16         {
17             // Deserializes the JSON string into collections of
   primitive data types.
18            Map<String, Object> results = (Map<String, Object>)
   JSON.deserializeUntyped(response.getBody());
19             // Cast the values in the 'animals' key as a list
20            Map<string,object> animals = (map<string,object>)
   results.get('animal');
21             System.debug('Received the following animals:' +
   animals );
22             strResp = string.valueof(animals.get('name'));
23             System.debug('strResp >>>>>>' + strResp );
24         }
25         return strResp ;
26     }
27
28 }
```

AnimalLocatorMock.apxc :

```
1  @isTest
2  global class AnimalLocatorMock implements HttpCalloutMock {
3      global HTTPResponse respond(HTTPRequest request) {
4          HttpResponse response = new HttpResponse();
```

```
        response.setHeader('Content-Type', 'application/json');

response.setBody('{"animal":{"id":1,"name":"chicken","eats":"chic

        response.setStatusCode(200);
        return response;
    }
}
```

AnimalLocatorTest.apxc :

```
@isTest
private class AnimalLocatorTest{
    @isTest static  void AnimalLocatorMock1() {
        Test.SetMock(HttpCallOutMock.class, new
AnimalLocatorMock());
        string result=AnimalLocator.getAnimalNameById(3);
        string expectedResult='chicken';
        System.assertEquals(result, expectedResult);
    }
}
```

AsyncCalculatorServices.apxc :

```
//Generated by wsdl2apex

public class AsyncCalculatorServices {
    public class doDivideResponseFuture extends System.WebServiceCalloutFuture
{
        public Double getValue() {
            calculatorServices.doDivideResponse response =
(calculatorServices.doDivideResponse)System.WebServiceCallout.endInvoke(this);
            return response.return_x;
        }
    }
    public class doSubtractResponseFuture extends
System.WebServiceCalloutFuture {
        public Double getValue() {
            calculatorServices.doSubtractResponse response =
(calculatorServices.doSubtractResponse)System.WebServiceCallout.endInvoke(this
);
            return response.return_x;
        }
    }
```

```
16      public class doMultiplyResponseFuture extends
    System.WebServiceCalloutFuture {
17          public Double getValue() {
18              calculatorServices.doMultiplyResponse response =
    (calculatorServices.doMultiplyResponse)System.WebServiceCallout.endInvoke(this
    );
19              return response.return_x;
20          }
21      }
22      public class doAddResponseFuture extends System.WebServiceCalloutFuture {
23          public Double getValue() {
24              calculatorServices.doAddResponse response =
    (calculatorServices.doAddResponse)System.WebServiceCallout.endInvoke(this);
25              return response.return_x;
26          }
27      }
28      public class AsyncCalculatorImplPort {
29          public String endpoint_x = 'https://th-apex-soap-

30          public Map<String,String> inputHttpHeaders_x;
31          public String clientCertName_x;
32          public Integer timeout_x;
33          private String[] ns_map_type_info = new
    String[]{'http://calculator.services/', 'calculatorServices'};
34          public AsyncCalculatorServices.doDivideResponseFuture
    beginDoDivide(System.Continuation continuation,Double arg0,Double arg1) {
35              calculatorServices.doDivide request_x = new
    calculatorServices.doDivide();
36              request_x.arg0 = arg0;
37              request_x.arg1 = arg1;
38              return (AsyncCalculatorServices.doDivideResponseFuture)
    System.WebServiceCallout.beginInvoke(
39                this,
40                request_x,
41                AsyncCalculatorServices.doDivideResponseFuture.class,
42                continuation,
43                new String[]{endpoint_x,
44                '',
45                'http://calculator.services/',
46                'doDivide',
47                'http://calculator.services/',
48                'doDivideResponse',
49                'calculatorServices.doDivideResponse'}
50              );
51          }
52          public AsyncCalculatorServices.doSubtractResponseFuture
    beginDoSubtract(System.Continuation continuation,Double arg0,Double arg1) {
53              calculatorServices.doSubtract request_x = new
    calculatorServices.doSubtract();
```

```
54              request_x.arg0 = arg0;
55              request_x.arg1 = arg1;
56              return (AsyncCalculatorServices.doSubtractResponseFuture)
    System.WebServiceCallout.beginInvoke(
57                this,
58                request_x,
59                AsyncCalculatorServices.doSubtractResponseFuture.class,
60                continuation,
61                new String[]{endpoint_x,
62                '',
63                'http://calculator.services/',
64                'doSubtract',
65                'http://calculator.services/',
66                'doSubtractResponse',
67                'calculatorServices.doSubtractResponse'}
68              );
69          }
70          public AsyncCalculatorServices.doMultiplyResponseFuture
    beginDoMultiply(System.Continuation continuation,Double arg0,Double arg1) {
71              calculatorServices.doMultiply request_x = new
    calculatorServices.doMultiply();
72              request_x.arg0 = arg0;
73              request_x.arg1 = arg1;
74              return (AsyncCalculatorServices.doMultiplyResponseFuture)
    System.WebServiceCallout.beginInvoke(
75                this,
76                request_x,
77                AsyncCalculatorServices.doMultiplyResponseFuture.class,
78                continuation,
79                new String[]{endpoint_x,
80                '',
81                'http://calculator.services/',
82                'doMultiply',
83                'http://calculator.services/',
84                'doMultiplyResponse',
85                'calculatorServices.doMultiplyResponse'}
86              );
87          }
88          public AsyncCalculatorServices.doAddResponseFuture
    beginDoAdd(System.Continuation continuation,Double arg0,Double arg1) {
89              calculatorServices.doAdd request_x = new
    calculatorServices.doAdd();
90              request_x.arg0 = arg0;
91              request_x.arg1 = arg1;
92              return (AsyncCalculatorServices.doAddResponseFuture)
    System.WebServiceCallout.beginInvoke(
93                this,
94                request_x,
95                AsyncCalculatorServices.doAddResponseFuture.class,
```

```
                continuation,
                new String[]{endpoint_x,
                '',
                'http://calculator.services/',
                'doAdd',
                'http://calculator.services/',
                'doAddResponse',
                'calculatorServices.doAddResponse'}
            );
        }
    }
}
```

CalculatorServices.apxc :

```
//Generated by wsdl2apex

public class calculatorServices {
    public class doDivideResponse {
        public Double return_x;
        private String[] return_x_type_info = new
String[]{'return','http://calculator.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new
String[]{'http://calculator.services/','false','false'};
        private String[] field_order_type_info = new
String[]{'return_x'};
    }
    public class doMultiply {
        public Double arg0;
        public Double arg1;
        private String[] arg0_type_info = new
String[]{'arg0','http://calculator.services/',null,'0','1','false'};
        private String[] arg1_type_info = new
String[]{'arg1','http://calculator.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new
String[]{'http://calculator.services/','false','false'};
        private String[] field_order_type_info = new
String[]{'arg0','arg1'};
    }
    public class doAdd {
        public Double arg0;
        public Double arg1;
        private String[] arg0_type_info = new
String[]{'arg0','http://calculator.services/',null,'0','1','false'};
        private String[] arg1_type_info = new
String[]{'arg1','http://calculator.services/',null,'0','1','false'};
```

```
23          private String[] apex_schema_type_info = new
    String[]{'http://calculator.services/','false','false'};
24          private String[] field_order_type_info = new
    String[]{'arg0','arg1'};
25      }
26      public class doAddResponse {
27          public Double return_x;
28          private String[] return_x_type_info = new
    String[]{'return','http://calculator.services/',null,'0','1','false'};
29          private String[] apex_schema_type_info = new
    String[]{'http://calculator.services/','false','false'};
30          private String[] field_order_type_info = new
    String[]{'return_x'};
31      }
32      public class doDivide {
33          public Double arg0;
34          public Double arg1;
35          private String[] arg0_type_info = new
    String[]{'arg0','http://calculator.services/',null,'0','1','false'};
36          private String[] arg1_type_info = new
    String[]{'arg1','http://calculator.services/',null,'0','1','false'};
37          private String[] apex_schema_type_info = new
    String[]{'http://calculator.services/','false','false'};
38          private String[] field_order_type_info = new
    String[]{'arg0','arg1'};
39      }
40      public class doSubtract {
41          public Double arg0;
42          public Double arg1;
43          private String[] arg0_type_info = new
    String[]{'arg0','http://calculator.services/',null,'0','1','false'};
44          private String[] arg1_type_info = new
    String[]{'arg1','http://calculator.services/',null,'0','1','false'};
45          private String[] apex_schema_type_info = new
    String[]{'http://calculator.services/','false','false'};
46          private String[] field_order_type_info = new
    String[]{'arg0','arg1'};
47      }
48      public class doSubtractResponse {
49          public Double return_x;
50          private String[] return_x_type_info = new
    String[]{'return','http://calculator.services/',null,'0','1','false'};
51          private String[] apex_schema_type_info = new
    String[]{'http://calculator.services/','false','false'};
```

```
52          private String[] field_order_type_info = new
    String[]{'return_x'};
53      }
54      public class doMultiplyResponse {
55          public Double return_x;
56          private String[] return_x_type_info = new
    String[]{'return','http://calculator.services/',null,'0','1','false'};
57          private String[] apex_schema_type_info = new
    String[]{'http://calculator.services/','false','false'};
58          private String[] field_order_type_info = new
    String[]{'return_x'};
59      }
60      public class CalculatorImplPort {
61          public String endpoint_x = 'https://th-apex-soap-

62          public Map<String,String> inputHttpHeaders_x;
63          public Map<String,String> outputHttpHeaders_x;
64          public String clientCertName_x;
65          public String clientCert_x;
66          public String clientCertPasswd_x;
67          public Integer timeout_x;
68          private String[] ns_map_type_info = new
    String[]{'http://calculator.services/', 'calculatorServices'};
69          public Double doDivide(Double arg0,Double arg1) {
70              calculatorServices.doDivide request_x = new
    calculatorServices.doDivide();
71              request_x.arg0 = arg0;
72              request_x.arg1 = arg1;
73              calculatorServices.doDivideResponse response_x;
74              Map<String, calculatorServices.doDivideResponse>
    response_map_x = new Map<String, calculatorServices.doDivideResponse>();
75              response_map_x.put('response_x', response_x);
76              WebServiceCallout.invoke(
77                this,
78                request_x,
79                response_map_x,
80                new String[]{endpoint_x,
81                '',
82                'http://calculator.services/',
83                'doDivide',
84                'http://calculator.services/',
85                'doDivideResponse',
86                'calculatorServices.doDivideResponse'}
87              );
```

```
88              response_x = response_map_x.get('response_x');
89              return response_x.return_x;
90          }
91          public Double doSubtract(Double arg0,Double arg1) {
92              calculatorServices.doSubtract request_x = new
    calculatorServices.doSubtract();
93              request_x.arg0 = arg0;
94              request_x.arg1 = arg1;
95              calculatorServices.doSubtractResponse response_x;
96              Map<String, calculatorServices.doSubtractResponse>
    response_map_x = new Map<String,
    calculatorServices.doSubtractResponse>();
97              response_map_x.put('response_x', response_x);
98              WebServiceCallout.invoke(
99                this,
100               request_x,
101               response_map_x,
102               new String[]{endpoint_x,
103               '',
104               'http://calculator.services/',
105               'doSubtract',
106               'http://calculator.services/',
107               'doSubtractResponse',
108               'calculatorServices.doSubtractResponse'}
109             );
110             response_x = response_map_x.get('response_x');
111             return response_x.return_x;
112         }
113         public Double doMultiply(Double arg0,Double arg1) {
114             calculatorServices.doMultiply request_x = new
    calculatorServices.doMultiply();
115             request_x.arg0 = arg0;
116             request_x.arg1 = arg1;
117             calculatorServices.doMultiplyResponse response_x;
118             Map<String, calculatorServices.doMultiplyResponse>
    response_map_x = new Map<String,
    calculatorServices.doMultiplyResponse>();
119             response_map_x.put('response_x', response_x);
120             WebServiceCallout.invoke(
121               this,
122               request_x,
123               response_map_x,
124               new String[]{endpoint_x,
125               '',
```

```
            'http://calculator.services/',
            'doMultiply',
            'http://calculator.services/',
            'doMultiplyResponse',
            'calculatorServices.doMultiplyResponse'}
          );
          response_x = response_map_x.get('response_x');
          return response_x.return_x;
      }
      public Double doAdd(Double arg0,Double arg1) {
          calculatorServices.doAdd request_x = new
  calculatorServices.doAdd();
          request_x.arg0 = arg0;
          request_x.arg1 = arg1;
          calculatorServices.doAddResponse response_x;
          Map<String, calculatorServices.doAddResponse> response_map_x
  = new Map<String, calculatorServices.doAddResponse>();
          response_map_x.put('response_x', response_x);
          WebServiceCallout.invoke(
            this,
            request_x,
            response_map_x,
            new String[]{endpoint_x,
            '',
            'http://calculator.services/',
            'doAdd',
            'http://calculator.services/',
            'doAddResponse',
            'calculatorServices.doAddResponse'}
          );
          response_x = response_map_x.get('response_x');
          return response_x.return_x;
      }
  }
}
```

ParkLocator.apxc :

```
public class ParkLocator {
    public static String[] country(String country){
        ParkService.ParksImplPort parks = new
  ParkService.ParksImplPort();
```

```
        String[] parksname = parks.byCountry(country);
        return parksname;
    }
}
```

ParkLocatorTest.apxc :

```
@isTest
private class ParkLocatorTest{
    @isTest
    static void testParkLocator() {
        Test.setMock(WebServiceMock.class, new
ParkServiceMock());
        String[] arrayOfParks = ParkLocator.country('India');

        System.assertEquals('Park1', arrayOfParks[0]);
    }
}
```

ParkServiceMock.apxc :

```
@isTest
global class ParkServiceMock implements WebServiceMock {
    global void doInvoke(
            Object stub,
            Object request,
            Map<String, Object> response,
            String endpoint,
            String soapAction,
            String requestName,
            String responseNS,
            String responseName,
            String responseType) {
        ParkService.byCountryResponse response_x = new
ParkService.byCountryResponse();
        List<String> lstOfDummyParks = new List<String>
{'Park1','Park2','Park3'};
        response_x.return_x = lstOfDummyParks;

        response.put('response_x', response_x);
```

```
    }
}
```

Apex Specialist Superbadge

CreateDefaultData.apxc :

```
public with sharing class CreateDefaultData{
    Static Final String TYPE_ROUTINE_MAINTENANCE = 'Routine

    //gets value from custom metadata How_We_Roll_Settings__mdt to know if Default data was created
    @AuraEnabled
    public static Boolean isDataCreated() {
        How_We_Roll_Settings__c customSetting = How_We_Roll_Settings__c.getOrgDefaults();
        return customSetting.Is_Data_Created__c;
    }

    //creates Default Data for How We Roll application
    @AuraEnabled
    public static void createDefaultData(){
        List<Vehicle__c> vehicles = createVehicles();
        List<Product2> equipment = createEquipment();
        List<Case> maintenanceRequest = createMaintenanceRequest(vehicles);
        List<Equipment_Maintenance_Item__c> joinRecords = createJoinRecords(equipment, maintenanceRequest);

        updateCustomSetting(true);
    }


    public static void updateCustomSetting(Boolean isDataCreated){
        How_We_Roll_Settings__c customSetting = How_We_Roll_Settings__c.getOrgDefaults();
        customSetting.Is_Data_Created__c = isDataCreated;
        upsert customSetting;
    }

    public static List<Vehicle__c> createVehicles(){
```

```
        List<Vehicle__c> vehicles = new List<Vehicle__c>();
        vehicles.add(new Vehicle__c(Name = 'Toy Hauler RV',
Air_Conditioner__c = true, Bathrooms__c = 1, Bedrooms__c = 1, Model__c =
'Toy Hauler RV'));
        vehicles.add(new Vehicle__c(Name = 'Travel Trailer RV',
Air_Conditioner__c = true, Bathrooms__c = 2, Bedrooms__c = 2, Model__c =
'Travel Trailer RV'));
        vehicles.add(new Vehicle__c(Name = 'Teardrop Camper',
Air_Conditioner__c = true, Bathrooms__c = 1, Bedrooms__c = 1, Model__c =
'Teardrop Camper'));
        vehicles.add(new Vehicle__c(Name = 'Pop-Up Camper',
Air_Conditioner__c = true, Bathrooms__c = 1, Bedrooms__c = 1, Model__c =
'Pop-Up Camper'));
        insert vehicles;
        return vehicles;
    }

    public static List<Product2> createEquipment(){
        List<Product2> equipments = new List<Product2>();
        equipments.add(new Product2(Warehouse_SKU__c =
'55d66226726b611100aaf741',name = 'Generator 1000 kW',
Replacement_Part__c = true,Cost__c = 100 ,Maintenance_Cycle__c = 100));
        equipments.add(new Product2(name = 'Fuse
aintenance_Cycle__c =
30  ));
        equipments.add(new Product2(name = 'Breaker
ntenance_Cycle__c =
15));
        equipments.add(new Product2(name = 'UPS 20
ntenance_Cycle__c =
60));
        insert equipments;
        return equipments;

    }

    public static List<Case> createMaintenanceRequest(List<Vehicle__c>
vehicles){
        List<Case> maintenanceRequests = new List<Case>();
        maintenanceRequests.add(new Case(Vehicle__c =
vehicles.get(1).Id, Type = TYPE_ROUTINE_MAINTENANCE, Date_Reported__c =
Date.today()));
        maintenanceRequests.add(new Case(Vehicle__c =
vehicles.get(2).Id, Type = TYPE_ROUTINE_MAINTENANCE, Date_Reported__c =
```

```
    Date.today()));
53          insert maintenanceRequests;
54          return maintenanceRequests;
55      }
56
57      public static List<Equipment_Maintenance_Item__c>
    createJoinRecords(List<Product2> equipment, List<Case>
    maintenanceRequest){
58          List<Equipment_Maintenance_Item__c> joinRecords = new
    List<Equipment_Maintenance_Item__c>();
59          joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
    equipment.get(0).Id, Maintenance_Request__c =
    maintenanceRequest.get(0).Id));
60          joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
    equipment.get(1).Id, Maintenance_Request__c =
    maintenanceRequest.get(0).Id));
61          joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
    equipment.get(2).Id, Maintenance_Request__c =
    maintenanceRequest.get(0).Id));
62          joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
    equipment.get(0).Id, Maintenance_Request__c =
    maintenanceRequest.get(1).Id));
63          joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
    equipment.get(1).Id, Maintenance_Request__c =
    maintenanceRequest.get(1).Id));
64          joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
    equipment.get(2).Id, Maintenance_Request__c =
    maintenanceRequest.get(1).Id));
65          insert joinRecords;
66          return joinRecords;
67
68      }
69  }
```

CreateDefaultDataTest.apxc :

```
1   @isTest
2   private class CreateDefaultDataTest {
3       @isTest
4       static void createData_test(){
5           Test.startTest();
6           CreateDefaultData.createDefaultData();
7           List<Vehicle__c> vehicles = [SELECT Id FROM Vehicle__c];
8           List<Product2> equipment = [SELECT Id FROM Product2];
```

```
        List<Case> maintenanceRequest = [SELECT Id FROM Case];
        List<Equipment_Maintenance_Item__c> joinRecords = [SELECT Id FROM Equipment_Maintenance_Item__c];

        System.assertEquals(4, vehicles.size(), 'There should have been

        System.assertEquals(4, equipment.size(), 'There should have been

        System.assertEquals(2, maintenanceRequest.size(), 'There should

        System.assertEquals(6, joinRecords.size(), 'There should have


    }

    @isTest
    static void updateCustomSetting_test(){
        How_We_Roll_Settings__c customSetting = How_We_Roll_Settings__c.getOrgDefaults();
        customSetting.Is_Data_Created__c = false;
        upsert customSetting;

        System.assertEquals(false, CreateDefaultData.isDataCreated(), 'The custom setting How_We_Roll_Settings__c.Is_Data_Created__c should be


        customSetting.Is_Data_Created__c = true;
        upsert customSetting;

        System.assertEquals(true, CreateDefaultData.isDataCreated(), 'The custom setting How_We_Roll_Settings__c.Is_Data_Created__c should be


    }
}
```

MaintenanceRequest.apxt :

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

MaintenanceRequestHelper.apxc :

```
public with sharing class MaintenanceRequestHelper {
    public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
        Set<Id> validIds = new Set<Id>();


        For (Case c : updWorkOrders){
            if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){
                if (c.Type == 'Repair' || c.Type == 'Routine

                    validIds.add(c.Id);



                }
            }
        }

        if (!validIds.isEmpty()){
            List<Case> newCases = new List<Case>();
            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)
                                                          FROM Case WHERE Id IN :validIds]);
            Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
            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];

        for (AggregateResult ar : results){
            maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
        }

            for(Case cc : closedCasesM.values()){
                Case nc = new Case (
                    ParentId = cc.Id,
                Status = 'New',
                    Subject = 'Routine Maintenance',
                    Type = 'Routine Maintenance',
                    Vehicle__c = cc.Vehicle__c,
```

```
                    Equipment__c =cc.Equipment__c,
                    Origin = 'Web',
                    Date_Reported__c = Date.Today()

                );

                If (maintenanceCycles.containskey(cc.Id)){
                    nc.Date_Due__c = Date.today().addDays((Integer)
maintenanceCycles.get(cc.Id));
                }

                newCases.add(nc);
             }

            insert newCases;

            List<Equipment_Maintenance_Item__c> clonedWPs = new
List<Equipment_Maintenance_Item__c>();
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c wp :
closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c wpClone = wp.clone();
                    wpClone.Maintenance_Request__c = nc.Id;
                    ClonedWPs.add(wpClone);

                }
             }
             insert ClonedWPs;
         }
     }
}
```

MaintenanceRequestHelperTest.apxc :

```
@istest
public with sharing class MaintenanceRequestHelperTest {

    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';
    private static final string REQUEST_ORIGIN = 'Web';
    private static final string REQUEST_TYPE = 'Routine Maintenance';
    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(vehicleList.get(i).id,
    equipmentList.get(i).id));
```

```
        }
        insert requestList;

        for(integer i = 0; i < 300; i++){
            workPartList.add(createWorkPart(equipmentList.get(i).id,
   requestList.get(i).id));
        }
        insert workPartList;

        test.startTest();
        for(case req : requestList){
            req.Status = CLOSED;
            oldRequestIds.add(req.Id);
        }
        update requestList;
        test.stopTest();

        list<case> allRequests = [select id
                                  from case
                                  where status =: STATUS_NEW];

        list<Equipment_Maintenance_Item__c> workParts = [select id
                                                         from
   Equipment_Maintenance_Item__c
                                                         where
   Maintenance_Request__c in: oldRequestIds];

        system.assert(allRequests.size() == 300);
    }
}
```

WarehouseCalloutService.apxc :

```
public with sharing class WarehouseCalloutService {

    private static final String WAREHOUSE_URL = 'https://th-superbadge
-apex.herokuapp.com/equipment';

    //@future(callout=true)
    public static void runWarehouseEquipmentSync(){

        Http http = new Http();
        HttpRequest request = new HttpRequest();

```

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


        List<Product2> warehouseEq = new List<Product2>();

        if (response.getStatusCode() == 200){
            List<Object> jsonResponse =
(List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());

            for (Object eq : jsonResponse){
                Map<String,Object> mapJson = (Map<String,Object>)eq;
                Product2 myEq = new Product2();
                myEq.Replacement_Part__c = (Boolean)
mapJson.get('replacement');
                myEq.Name = (String) mapJson.get('name');
                myEq.Maintenance_Cycle__c = (Integer)
mapJson.get('maintenanceperiod');
                myEq.Lifespan_Months__c = (Integer)
mapJson.get('lifespan');
                myEq.Cost__c = (Decimal) mapJson.get('lifespan');
                myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
                myEq.Current_Inventory__c = (Double)
mapJson.get('quantity');
                warehouseEq.add(myEq);
            }

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

        }
    }
}
```

WarehouseCalloutServiceMock.apxc :

```
@isTest
global class WarehouseCalloutServiceMock implements HttpCalloutMock {
    // implement http mock callout
    global static HttpResponse respond(HttpRequest request){

        System.assertEquals('https://th-superbadge-
));
        System.assertEquals('GET', request.getMethod());

        // Create a fake response
        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":
false,"quantity":5,"name":"Generator 1000
,"cost":5000,"sku":"100003"}]'
);
        response.setStatusCode(200);
        return response;
    }
}
```

WarehouseCalloutServiceTest.apxc :

```
@isTest

private class WarehouseCalloutServiceTest {
    @isTest
    static void testWareHouseCallout(){
        Test.startTest();
        // implement mock callout test here
        Test.setMock(HTTPCalloutMock.class, new
WarehouseCalloutServiceMock());
        WarehouseCalloutService.runWarehouseEquipmentSync();
        Test.stopTest();
        System.assertEquals(1, [SELECT count() FROM Product2]);
    }
}
```

WarehouseSyncSchedule.apxc :

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

        WarehouseCalloutService.runWarehouseEquipmentSync();
    }
}
```

WarehouseSyncScheduleTest.apxc :

```
@isTest
public class WarehouseSyncScheduleTest {

    @isTest static void WarehousescheduleTest(){
        String scheduleTime = '00 00 01 * * ?';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class, new
WarehouseCalloutServiceMock());
        String jobID=System.schedule('Warehouse Time To Schedule

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


    }
}
```