

Module: Get Started With Apex triggers

AccountAddressTrigger.apxt

```
trigger AccountAddressTrigger on Account (before insert, before update) {
```

```
    for(Account account:Trigger.New){
```

```
        if(account.Match_Billing_Address__c == True){
```

```
            account.ShippingPostalCode = account.BillingPostalCode;
```

```
        }
```

```
    }
```

```
}
```

The screenshot shows the Salesforce Developer Console interface. The top part displays the code for the `AccountAddressTrigger.apxt` trigger. The code is as follows:

```
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 }
```

The bottom part of the screenshot shows the **Logs** tab, which contains a table of log entries. The table has columns for User, Application, Operation, Time, Status, Read, and Size. The log entries are as follows:

User	Application	Operation	Time	Status	Read	Size
Sudipta Datta	Unknown	/services/data/v48.0/t...	5/23/2022, 3:41:51 AM	Success	Unread	330.8 KB
Sudipta Datta	Unknown	/services/data/v48.0/t...	5/23/2022, 3:41:49 AM	Success	Unread	8.35 KB
Sudipta Datta	Browser	/aura	5/23/2022, 3:39:54 AM	Success	Unread	4.5 KB
Sudipta Datta	Unknown	/services/data/v48.0/t...	5/23/2022, 3:31:22 AM	Success	Unread	8.27 KB
Sudipta Datta	Unknown	/services/data/v48.0/t...	5/23/2022, 3:31:20 AM	Success	Unread	8.14 KB
Sudipta Datta	Unknown	/services/data/v48.0/t...	5/23/2022, 3:29:25 AM	Assertion Failed: Expec...	Unread	7.72 KB

Module: Apex Testing: Bulk Apex Triggers

ClosedOpportunityTrigger.apxt

```

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

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

    for(Opportunity opp: Trigger.New){

        if(opp.StageName == 'Closed Won'){

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

        }

    }

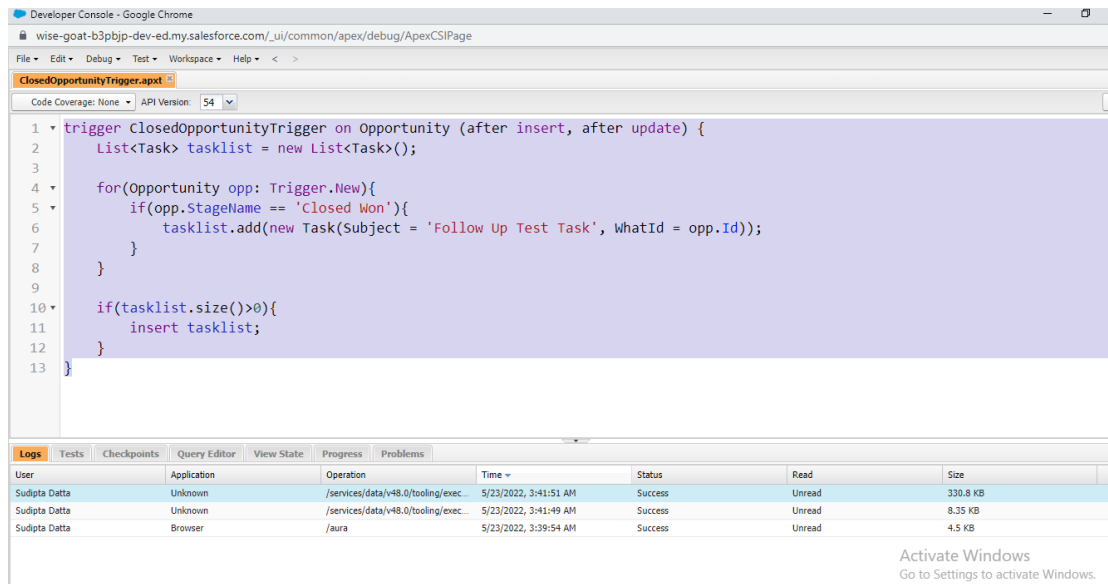
    if(tasklist.size()>0){

        insert tasklist;

    }

}

```



Module:Get Started With Apex Unit Tests

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
@TestVisible 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
@TestVisible 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;
    }
}

```

TestVerifyDate.apxc

@isTest

private class TestVerifyDate {

@isTest static void Test_CheckDates_case1(){

Date D = VerifyDate.CheckDates(date.parse('01/01/2020'), date.parse('01/05/2020'));

System.assertEquals(date.parse('01/05/2020'), D);

}

@isTest static void Test_CheckDates_case2(){

Date D = VerifyDate.CheckDates(date.parse('01/01/2020'), date.parse('05/05/2020'));

System.assertEquals(date.parse('01/31/2020'), D);

}

@isTest static void Test_DateWithin30Days_case1(){

Boolean flag = VerifyDate.DateWithin30Days(date.parse('01/01/2020'), date.parse('12/30/2019'));

System.assertEquals(false, flag);

}

@isTest static void Test_DateWithin30Days_case2(){

Boolean flag = VerifyDate.DateWithin30Days(date.parse('01/01/2020'), date.parse('02/02/2020'));

```

        System.assertEquals(false, flag);
    }

    @isTest static void Test_DateWithin30Days_case3(){

        Boolean flag = VerifyDate.DateWithin30Days(date.parse('01/01/2020'), date.parse('01/15/2020'));

        System.assertEquals(false, flag);
    }

    @isTest static void Test_SetEndOfMonthDate(){

        Date returndate = VerifyDate.SetEndOfMonthDate(date.parse('01/01/2020'));

    }

}

```

The screenshot shows the Salesforce Developer Console with the following components:

- Code Editor:** Displays the `VerifyDate` class with the following code:


```

1 public class VerifyDate {
2
3     //method to handle potential checks against two dates
4     public static Date CheckDates(Date date1, Date date2) {
5         //if date2 is within the next 30 days of date1, use date2. Otherwise use the end of the month
6         if(DateWithin30Days(date1,date2)) {
7             return date2;
8         } else {
9             return SetEndOfMonthDate(date1);
10        }
11    }
12
13    //method to check if date2 is within the next 30 days of date1
14    @TestVisible private static Boolean DateWithin30Days(Date date1, Date date2) {
15        //check for date2 being in the past
16        if( date2 < date1) { return false; }
17    }
18 }

```
- Tests Tab:** Shows a test run summary table.

Status	Test Run	Enqueued Time	Duration	Failures	Total
✖	TestRun @ 4:07:35 pm			1	6
- Overall Code Coverage:** A table showing coverage for various classes.

Class	Percent	Lines
Overall	59%	
AccountAddressTrigger	0%	0/3
ClosedOppSystemTrigger	0%	0/6
VerifyDate	100%	13/13

Developer Console - Google Chrome

wise-goat-b3pbjp-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help < >

VerifyDate.apxc TestVerifyDate.apxc

Code Coverage: None API Version: 53 Run Test Go To

```

1  @isTest
2  private class TestVerifyDate {
3
4  @isTest static void Test_CheckDates_case1(){
5      Date D = VerifyDate.CheckDates(date.parse('01/01/2020'), date.parse('01/05/2020'));
6      System.assertEquals(date.parse('01/05/2020'), D);
7  }
8
9
10 @isTest static void Test_CheckDates_case2(){
11     Date D = VerifyDate.CheckDates(date.parse('01/01/2020'), date.parse('05/05/2020'));
12     System.assertEquals(date.parse('01/31/2020'), D);
13 }
14 @isTest static void Test_DateWithin30Days_case1(){
15     Boolean flag = VerifyDate.DateWithin30Days(date.parse('01/01/2020'), date.parse('12/30/2019'));
16     System.assertEquals(false, flag);

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total
✖	TestRun @ 4:07:35 pm			1	6

Overall Code Coverage

Class	Percent	Lines
Overall	59%	
AccountAddressTrigger	0%	0/3
ClosedOpportunityTrigger	0%	0/6
VerifyDate	100%	13/13

Go to Settings to activate Windows.

Module: Test Apex Triggers

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 for DML');

 }

 }

```
}
```

TestRestrictContactByName.apxc 

```
@isTest
```

```
public class TestRestrictContactByName {
```

```
    @isTest static void Test_insertupdateContact(){
```

```
        Contact cnt = new Contact();
```

```
        cnt.LastName = 'INVALIDNAME';
```

```
        Test.startTest();
```

```
        Database.SaveResult result = Database.insert(cnt, false);
```

```
        Test.stopTest();
```

```
        System.assert(!result.isSuccess());
```

```
        System.assert(result.getErrors().size() > 0);
```

```
        System.assertEquals('The Last Name "INVALIDNAME" is not allowed for  
DML',result.getErrors()[0].getMessage());
```

```
    }
```

```
}
```

Developer Console - Google Chrome

wise-goat-b3pbjp-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help

RestrictContactByName.apxt TestRestrictContactByName.apxc

Code Coverage: None API Version: 53 Go To

```

1 trigger RestrictContactByName on Contact (before insert, before update) {
2
3     //check contacts prior to insert or update for invalid data
4     For (Contact c : Trigger.New) {
5         if(c.LastName == 'INVALIDNAME') { //invalidname is invalid
6             c.AddError('The Last Name "'+c.LastName+'" is not allowed for DML');
7         }
8     }
9
10
11
12
13 }
14

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total
✖	TestRun @ 4:24:31 pm			1	1
✖	TestRun @ 4:24:39 pm			1	1

Overall Code Coverage

Class	Percent	Lines
Overall	64%	
AccountAddressTrigger	0%	0/3
ClosedOpportunityTrigger	0%	0/6
RestrictContactByName	100%	3/3
VerifyDate	100%	13/13

Developer Console - Google Chrome

wise-goat-b3pbjp-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help

RestrictContactByName.apxt TestRestrictContactByName.apxc

Code Coverage: None API Version: 54 Run Test Go To

```

2 public class TestRestrictContactByName {
3
4     @isTest static void Test_insertupdateContact(){
5         Contact cnt = new Contact();
6         cnt.LastName = 'INVALIDNAME';
7
8         Test.startTest();
9         Database.SaveResult result = Database.insert(cnt, false);
10        Test.stopTest();
11
12        System.assert(!result.isSuccess());
13        System.assert(result.getErrors().size() > 0);
14        System.assertEquals('The Last Name "INVALIDNAME" is not allowed for DML',result.getErrors()[0].getMessage());
15
16    }
17 }

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total
✖	TestRun @ 4:24:31 pm			1	1
✖	TestRun @ 4:24:39 pm			1	1

Overall Code Coverage

Class	Percent	Lines
Overall	64%	
AccountAddressTrigger	0%	0/3
ClosedOpportunityTrigger	0%	0/6
RestrictContactByName	100%	3/3
VerifyDate	100%	13/13

Module: Create Test Data for Apex Tests

```
public class RandomContactFactory {
```

```
    public static List<Contact> generateRandomContacts(Integer numcnt, string lastname){
```

```
        List<Contact> contacts = new List<Contact> ();
```



```

for(Integer i=0;i<numcnt;i++){

    Contact cnt = new Contact(FirstName = 'Test '+i, LastName = lastname);

    contacts.add(cnt);

}

return contacts;

}

}

```

The screenshot shows the Salesforce Developer Console with the file `RandomContactFactory.apxc` open. The code defines a class `RandomContactFactory` with a static method `generateRandomContacts` that creates a list of contacts. Below the code editor, the `Tests` tab is active, showing a table of test results.

Status	Test Run	Enqueued Time	Duration	Failures	Total

Overall Code Coverage		
Class	Percent	Lines
Overall	51%	
AccountAddressTrigger	0%	0/3
ClosedOpportunityTrigger	0%	0/6
RandomContactFactory	0%	0/6
RestrictContactByName	100%	3/3
VerifyDate	100%	13/13

Asynchronous Apex > Use Future Methods

AccountProcessor.apxc

```

public class AccountProcessor {

    @future

    public static void countContacts(List<Id> accountIds){

```

```
List<Account> accountsToUpdate = new List<Account>();
```

```
List<Account> accounts = [Select Id, Name, (Select Id from Contacts) from Account Where Id in :accountIds];
```

```
for(Account acc:accounts){
```

```
    List<Contact> contactList = acc.Contacts;
```

```
    acc.Number_Of_Contacts__c = contactList.size();
```

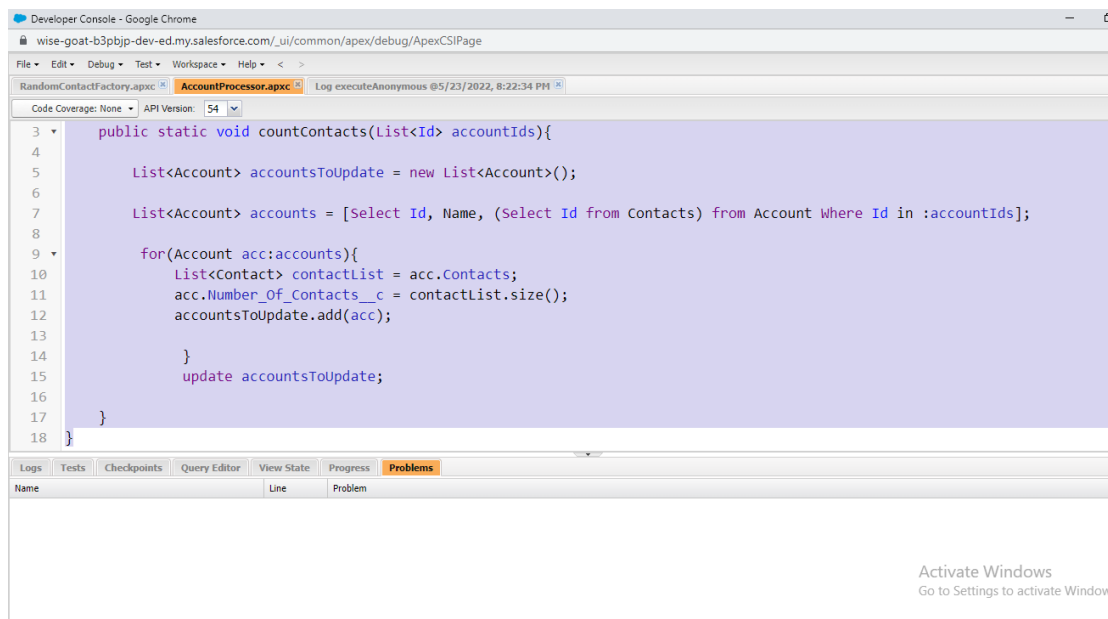
```
    accountsToUpdate.add(acc);
```

```
}
```

```
    update accountsToUpdate;
```

```
}
```

```
}
```



Developer Console - Google Chrome

wise-goat-b3pbjp-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help < >

RandomContactFactory.apxc AccountProcessor.apxc Log executeAnonymous @5/23/2022, 8:22:34 PM

Execution Log

Timestamp	Event	Details
20:22:34:002	USER_INFO	[EXTERNAL]0051w00000W1a1goldfishpubg2@wise-goat-b3pbjp.com((GMT-07:00) Pacific Daylight Time (America/Los_Angeles))GMT-07:00
20:22:34:002	EXECUTION_ST...	
20:22:34:002	CODE_UNIT_ST...	[EXTERNAL]execute_anonymous_apex
20:22:34:002	VARIABLE_SCO...	[1]accountIds[List<Id> true false
20:22:34:002	HEAP_ALLOCATE	[79]Bytes:3
20:22:34:002	HEAP_ALLOCATE	[84]Bytes:152
20:22:34:002	HEAP_ALLOCATE	[399]Bytes:408
20:22:34:002	HEAP_ALLOCATE	[412]Bytes:408
20:22:34:002	HEAP_ALLOCATE	[520]Bytes:48
20:22:34:002	HEAP_ALLOCATE	[139]Bytes:6
20:22:34:002	HEAP_ALLOCATE	[EXTERNAL]Bytes:5
20:22:34:003	STATEMENT_EX...	[1]
20:22:34:003	STATEMENT_EX...	[1]
20:22:34:003	HEAP_ALLOCATE	[1]Bytes:4
20:22:34:003	HEAP_ALLOCATE	[EXTERNAL]Bytes:4
20:22:34:003	VARIABLE_ASSI...	[1]accountIds[]0xf1d6c21

☐ This Frame ☐ Executable ☐ Debug Only ☐ Filter [Click here to filter the log](#)

Logs Tests Checkpoints Query Editor View State Progress **Problems**

Name	Line	Problem
------	------	---------

Activate Windows

AccountProcessorTest.apxc

@IsTest

```
private static void testCountContacts(){

    Account newAccount = new Account(Name='Test Account');

    insert newAccount;

    Contact newContact1 = new Contact(FirstName='John',LastName='Doe',AccountId = newAccount.Id);

    insert newContact1;

    Contact newContact2 = new Contact(FirstName='Jane',LastName='Doe',AccountId = newAccount.Id);

    insert newContact2;

    List<Id> AccountIds = new List<Id>();

    accountIds.add(newAccount.Id);

    Test.startTest();

    AccountProcessor.countContacts(accountIds);
```

```

Test.stopTest();

}

}

```

The screenshot shows the Salesforce Developer Console with the `AccountProcessorTest.apxc` file open. The code defines a test class that creates two contacts, inserts them, and then calls `AccountProcessor.countContacts` with a list of account IDs. The test results table at the bottom shows four test runs, with the last one being successful. The overall code coverage is 64%.

Status	Test Run	Enqueued Time	Duration	Failures	Total
✗	TestRun @ 8:34:55 pm			1	1
✗	TestRun @ 8:40:09 pm			1	1
✗	TestRun @ 8:43:32 pm			1	1
✓	TestRun @ 8:51:05 pm			0	1

Overall Code Coverage		
Class	Percent	Lines
Overall	64%	
AccountAddressTrigger	66%	2/3
AccountProcessor	100%	8/8
ClosedOpportunityTrigger	0%	0/6
RandomContactFactory	0%	0/6
RestrictContactByName	66%	2/3

Module:Use Batch Apex

LeadProcessor.apxc

```

global class LeadProcessor implements Database.Batchable<sObject> {

    global integer count = 0;

    global Database.QueryLocator start(Database.BatchableContext bc){

        return Database.getQueryLocator('SELECT ID, LeadSource FROM Lead');

    }

    global void execute(Database.BatchableContext bc, List<Lead> L_list){

        List<Lead> L_list_new = new List<lead>();
    }
}

```

```

for(lead L:L_list){

    L.leadsource = 'Dreamforce';

    L_list_new.add(L);

    count += 1;

}

update L_list_new;

}

```

```


global void finish(Database.BatchableContext bc){

    system.debug('count = ' + count);

}

}

```

LeadProcessorTest.apxc 

@isTest

```
public class LeadProcessorTest {
```

@isTest

```
public static void testit(){
```

```
    List<lead> L_list = new List<Lead>();
```

```
    for(Integer i=0; i<200; i++){
```

```
        Lead L = new lead();
```

```
        L.LastName = 'name' + i;
```

```
        L.Company = 'Company';
```

```

        L.Status = 'Random Status';

        L_list.add(L);
    }

    insert L_list;

    Test.startTest();

    LeadProcessor lp = new LeadProcessor();

    Id batchId = Database.executeBatch(lp);

    Test.stopTest();

}
}

```

The screenshot shows the Salesforce Developer Console with the Apex code for the `LeadProcessor` class and the Test Results table.

Apex Code:

```

1 global class LeadProcessor implements Database.Batchable<Object> {
2     global integer count = 0;
3
4     global Database.QueryLocator start(Database.BatchableContext bc){
5         return Database.getQueryLocator('SELECT ID, LeadSource FROM Lead');
6     }
7
8     global void execute(Database.BatchableContext bc, List<Lead> l_list){
9         List<Lead> l_list_new = new List<Lead>();
10
11         for(Lead L:l_list){
12             L.leadsource = 'Dreamforce';
13             l_list_new.add(L);
14             count += 1;
15         }
16     }
17 }

```

Test Results Table:

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	TestRun @ 10:13:17 pm			0	1

Overall Code Coverage Table:

Class	Percent	Lines
Overall	50%	
AccountAddressTrigger	66%	2/3
AccountProcessor	100%	8/8
ClosedOpportunityTrigger	0%	0/6
LeadProcessor	0%	0/11
RandomContactFactory	0%	0/6

Developer Console - Google Chrome

wise-goat-b3pbjp-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help

AccountProcessorTest.apxc LeadProcessor.apxc **LeadProcessorTest.apxc** ClosedOpportunityTrigger.apxt VerifyDate.apxc RestrictContactByName.apxt AccountAddressTrigger.apxt TestVerifyDate.apxc

Code Coverage: None API Version: 54 Run Test Go To

```

1  @isTest
2  public class LeadProcessorTest {
3
4      @isTest
5      public static void testit(){
6          List<lead> L_list = new List<Lead>();
7
8          for(Integer i=0; i<200; i++){
9              Lead L = new lead();
10             L.LastName = 'name' + i;
11             L.Company = 'Company';
12             L.Status = 'Random Status';
13             L_list.add(L);
14         }
15         insert L_list;

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total	Overall Code Coverage
✓	7071w0000002tyl	Mon May 23 2022 22:34:50 GM...		0	1	
✓	AccountProcessorTest			0	1	AccountProcessor 100% 8/8
✓	Dashboard_Pal.DashboardPalTest			0	1	ClosedOpportunityTrigger 0% 0/6
✓	LeadProcessorTest			0	1	LeadProcessor 100% 11/11
✗	TestRestrictContactByName			1	1	RandomContactFactory 0% 0/6
✗	TestVerifyDate			1	6	RestrictContactByName 100% 8/8
✓	trihdtps.tt_UtilControllerTest			0	2	VerifyDate 100% 13/13

Module: Control Processes with Queueable Apex

AddPrimaryContact.apxc

public class AddPrimaryContact implements Queueable{

private Contact con;

private String state;

public AddPrimaryContact(Contact con, String state){

 this.con = con;

 this.state = state;

}

public void execute(QueueableContext context){

 List<Account> accounts = [Select Id, Name, (Select FirstName, Id from contacts)

```

        from Account where BillingState = :state Limit 200];

List<Contact> primaryContacts = new List<Contact>();

for(Account acc:accounts){

    contact c = con.clone();

    c.AccountId = acc.Id;

    primaryContacts.add(c);

}

if(primaryContacts.size() > 0){

    insert primaryContacts;

}

}

}

AddPrimaryContactTest.apxc ✖

@isTest

public class AddPrimaryContactTest {

    static testmethod void testQueueable(){

        List<Account> testAccounts = new List<Account>();

        for(Integer i=0;i<50;i++){

            testAccounts.add(new Account(Name='Account '+i,BillingState='CA'));

        }
    }
}

```



```
for(Integer j=0;j<50;j++){

    testAccounts.add(new Account(Name='Account '+j,BillingState='NY'));

}

insert testAccounts;


Contact testContact = new Contact(FirstName = 'John', LastName = 'Doe');

insert testContact;


AddPrimaryContact addit = new addPrimaryContact(testContact, 'CA');


Test.startTest();

system.enqueueJob(addit);

Test.stopTest();


System.assertEquals(50,[Select count() from Contact where accountId in (Select Id from Account where
BillingState='CA')]);

}

}
```

The screenshot shows the Salesforce Developer Console with the `AddPrimaryContact.apex` class open. The class implements the `Queueable` interface. Below the code, the 'Tests' tab shows a table of test results and an 'Overall Code Coverage' summary.

```

1 public class AddPrimaryContact implements Queueable{
2
3     private Contact con;
4     private String state;
5
6     public AddPrimaryContact(Contact con, String state){
7         this.con = con;
8         this.state = state;
9     }
10
11    public void execute(QueueableContext context){
12        List<Account> accounts = [Select Id, Name, (Select FirstName, Id from contacts)
13                                from Account where BillingState = :state Limit 200];
14        List<Contact> primaryContacts = new List<Contact>();
15    }

```

Status	Test Run	Enqueued Time	Duration	Failures	Total
✖	7075w0000003he0	Wed May 25 2022 20:20:47 GH...		0	1
✖	7075w0000003he0	Wed May 25 2022 20:24:36 GH...		0	1
✔	TestRun @ 8:29:22 pm			0	1

Overall Code Coverage		
Class	Percent	Lines
Overall	79%	
AccountAddressTrigger	66%	2/3
AccountProcessor	100%	8/8
AddPrimaryContact	100%	13/13
ClosedOpportunityTrigger	100%	11/11
LeadProcessor	100%	11/11

The screenshot shows the Salesforce Developer Console with the `AddPrimaryContactTest` class open. The class contains a static test method `testQueueable()`. Below the code, the 'Tests' tab shows a table of test results and an 'Overall Code Coverage' summary.

```

1 @isTest
2 public class AddPrimaryContactTest {
3
4     static testmethod void testQueueable(){
5         List<Account> testAccounts = new List<Account>();
6         for(Integer i=0;i<50;i++){
7             testAccounts.add(new Account(Name='Account '+i,BillingState='CA'));
8         }
9         for(Integer j=0;j<50;j++){
10            testAccounts.add(new Account(Name='Account '+j,BillingState='NY'));
11        }
12        insert testAccounts;
13
14        Contact testContact = new Contact(FirstName = 'John', LastName = 'Doe');
15    }

```

Status	Test Run	Enqueued Time	Duration	Failures	Total
✖	7075w0000003he0	Wed May 25 2022 20:20:47 GH...		0	1
✖	7075w0000003he0	Wed May 25 2022 20:24:36 GH...		0	1
✔	TestRun @ 8:29:22 pm			0	1

Overall Code Coverage		
Class	Percent	Lines
Overall	79%	
AccountAddressTrigger	66%	2/3
AccountProcessor	100%	8/8
AddPrimaryContact	100%	13/13
ClosedOpportunityTrigger	100%	11/11
LeadProcessor	100%	11/11

Module: Apex Integration Services

Apex Integration Overview

DailyLeadProcessor.apxc

global class DailyLeadProcessor implements Schedulable{

global void execute(SchedulableContext ctx){

List<lead> leadstoupdate = new List<lead>();

List<Lead> leads = [Select id From Lead Where LeadSource = Null Limit 200];

```

for(Lead l:leads){

    l.LeadSource = 'DreamForce';

    leadstoupdate.add(l);

}

update leadstoupdate;

}

}

```

DailyLeadProcessorTest.apxc

@isTest

```
private class DailyLeadProcessorTest {
```

```
    public static String CRON_EXP = '0 0 0 15 3 ? 2023';
```

```
    static testmethod void testScheduledJob(){
```

```
        List<lead> leads = new List<lead>();
```

```
        for (Integer i=0; i<200; i++){
```

```
            Lead l = new Lead(
```

```
                FirstName = 'First ' + i,
```

```
                LastName = 'LastName',
```

```
                Company = 'The Inc'
```

```
            );
```

```
            leads.add(l);
```

```
        }
```

```
        insert leads;
```

```
        Test.startTest();
```

```
String jobId = System.schedule('ScheduledApexTest',CRON_EXP,new DailyLeadProcessor());
```

```
Test.stopTest();
```

```
List<Lead> checkleads = new List<Lead>();
```

```
checkleads = [Select Id From Lead Where LeadSource = 'Dreamforce' and Company = 'The Inc'];
```

```
System.assertEquals(200, checkleads.size(), 'Leads were not created');
```

```
}
```

```
}
```

The screenshot displays the Salesforce Developer Console. The top pane shows the Apex code for the `DailyLeadProcessor` class, which implements the `Schedulable` interface. The code defines an `execute` method that queries for leads with a `LeadSource` of `Null` and updates them with a `LeadSource` of `'Dreamforce'`. The bottom pane shows the `Tests` tab with a table of test results. The table includes columns for `Status`, `Test Run`, `Enqueued Time`, `Duration`, `Failures`, and `Total`. The results show that the `DailyLeadProcessor` test passed with 100% coverage. A right-hand pane displays the `Overall Code Coverage` for the project, showing 81% overall coverage and 100% coverage for the `DailyLeadProcessor` class.

Status	Test Run	Enqueued Time	Duration	Failures	Total
✗	7071w0000003ifm	Wed May 25 2022 21:43:10 GM...		0	1
✗	TestRun @ 9:43:39 pm			1	1
✗	TestRun @ 9:43:56 pm			1	1
✓	TestRun @ 9:44:10 pm			0	1
✗	7071w0000003iIR	Wed May 25 2022 21:45:35 GM...		0	1
✗	7071w0000003iJP	Wed May 25 2022 21:47:14 GM...		0	1
✓	TestRun @ 9:49:21 pm			0	1

Class	Percent	Lines
Overall	81%	
AccountAddressTrigger	66%	2/3
AccountProcessor	100%	8/8
AddPrimaryContact	100%	13/13
ClosedOpportunityTrigger	100%	6/6
DailyLeadProcessor	100%	7/7

Developer Console - Google Chrome

wise-goat-b3pbjp-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help

h.apxc DailyLeadProcessor.apxc DailyLeadProcessorTest.apxc ClosedOpportunityTrigger.apxc VerifyDate.apxc TestVerifyDate.apxc TestRestrictContactByName.apxc AccountAddressTrigger.apxc

Code Coverage: None API Version: 54 Run Test Go To

```

1 @isTest
2 private class DailyLeadProcessorTest {
3
4     public static String CRON_EXP = '0 0 0 15 3 ? 2023';
5     static testmethod void testscheduledJob(){
6         List<lead> leads = new List<lead>();
7         for (Integer i=0; i<200; i++){
8             Lead l = new Lead(
9                 FirstName = 'First ' + i,
10                LastName = 'LastName',
11                Company = 'The Inc'
12            );
13            leads.add(l);
14        }
15        insert leads;

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total
✗	7071w00000031fm	Wed May 25 2022 21:43:10 GM...		0	1
✗	TestRun @ 9:43:39 pm			1	1
✗	TestRun @ 9:43:56 pm			1	1
✓	TestRun @ 9:44:10 pm			0	1
✗	7071w00000031R	Wed May 25 2022 21:45:35 GM...		0	1
✗	7071w00000031P	Wed May 25 2022 21:47:14 GM...		0	1
✓	TestRun @ 9:49:21 pm			0	1

Overall Code Coverage		
Class	Percent	Lines
Overall	81%	
AccountAddressTrigger	66%	2/3
AccountProcessor	100%	8/8
AddPrimaryContact	100%	13/13
ClosedOpportunityTrigger	100%	6/6
DailyLeadProcessor	100%	7/7

Apex REST Callouts

AnimalLocator.apxc

```

public class AnimalLocator{
    public static String getAnimalNameById(Integer x){
        Http http = new Http();
        HttpRequest req = new HttpRequest();
        req.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/' + x);
        req.setMethod('GET');
        Map<String, Object> animal= new Map<String, Object>();
        HttpResponse res = http.send(req);
        if (res.getStatusCode() == 200) {
            Map<String, Object> results = (Map<String, Object>)JSON.deserializeUntyped(res.getBody());
            animal = (Map<String, Object>) results.get('animal');
        }
        return (String)animal.get('name');
    }
}

```

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 );

}

}

```

AnimalLocatorMock.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 );

    }

}

```

The screenshot shows the Salesforce Developer Console in Google Chrome. The browser address bar displays the URL: `wise-goat-b3pbjp-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. The console's tab bar shows several open files, with `AnimalLocator.apxc` selected and highlighted in orange. Below the tabs, the code editor displays the following Apex code for the `AnimalLocator` class:

```

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-callout.herokuapp.com/animals/' + x);
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 }
16

```

At the bottom of the console, there is a navigation bar with tabs for `Logs`, `Tests`, `Checkpoints` (which is active), `Query Editor`, `View State`, `Progress`, and `Problems`. Below this bar, the `Checkpoints` and `Checkpoint Locations` sections are visible, each with a table structure for tracking execution details.

Checkpoints				Checkpoint Locations		
Namespace	Class	Line	Date	File	Line	Iteration

```

1  @isTest
2  private class AnimalLocatorTest{
3      @isTest static void AnimalLocatorMock1() {
4          Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
5          String result = AnimalLocator.getAnimalNameById(3);
6          String expectedResult = 'chicken';
7          System.assertEquals(result,expectedResult );
8      }
9  }

```

```

1  @isTest
2  global class AnimalLocatorMock implements HttpCalloutMock {
3      // Implement this interface method
4      global HTTPResponse respond(HTTPRequest request) {
5          // Create a fake response
6          HttpResponse response = new HttpResponse();
7          response.setHeader('Content-Type', 'application/json');
8          response.setBody('{"animals": ["majestic badger", "fluffy bunny", "scary bear", "chicken", "mighty moose"]}');
9          response.setStatusCode(200);
10         return response;
11     }
12 }

```

Checkpoints				Checkpoint Locations		
Namespace	Class	Line	Date	File	Line	Iteration

Apex Integration Services

Apex SOAP Callouts

ParkLocator.apxc

```

public class ParkLocator {

    public static string[] country(string theCountry) {

        ParkService.ParksImplPort parkSvc = new ParkService.ParksImplPort(); // remove space

        return parkSvc.byCountry(theCountry);

    }

}

```

```

1 public class ParkLocator {
2     public static String[] country(String theCountry) {
3         ParkService.ParksImplPort parkSvc = new ParkService.ParksImplPort(); // remove space
4         return parkSvc.byCountry(theCountry);
5     }
6 }

```

ParkLocatorTest.apxc

@isTest

```
private class ParkLocatorTest {
```

```
    @isTest static void testCallout() {
```

```
        Test.setMock(WebServiceMock.class, new ParkServiceMock ());
```

```
        String country = 'United States';
```

```
        List<String> result = ParkLocator.country(country);
```

```
        List<String> parks = new List<String>{'Yellowstone', 'Mackinac National Park', 'Yosemite'};
```

```
        System.assertEquals(parks, result);
```

```
    }
```

```
}
```

```

1 @isTest
2 private class ParkLocatorTest {
3     @isTest static void testCallout() {
4         Test.setMock(WebServiceMock.class, new ParkServiceMock ());
5         String country = 'United States';
6         List<String> result = ParkLocator.country(country);
7         List<String> parks = new List<String>{'Yellowstone', 'Mackinac National Park', 'Yosemite'};
8         System.assertEquals(parks, result);
9     }
10 }
11
12

```

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	LeadProcessorTest			0	1
✓	ParkLocatorTest			0	1
✓	ParkServiceMock			0	0
✗	TestRestrictContactByName			1	1
✗	TestVerifyDate			1	6

Overall Code Coverage		
Class	Percent	Lines
Overall	74%	
AccountAddressTrigger	66%	2/3
AccountProcessor	100%	8/8

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) {

// start - specify the response you want to send

ParkService.byCountryResponse response_x = new ParkService.byCountryResponse();

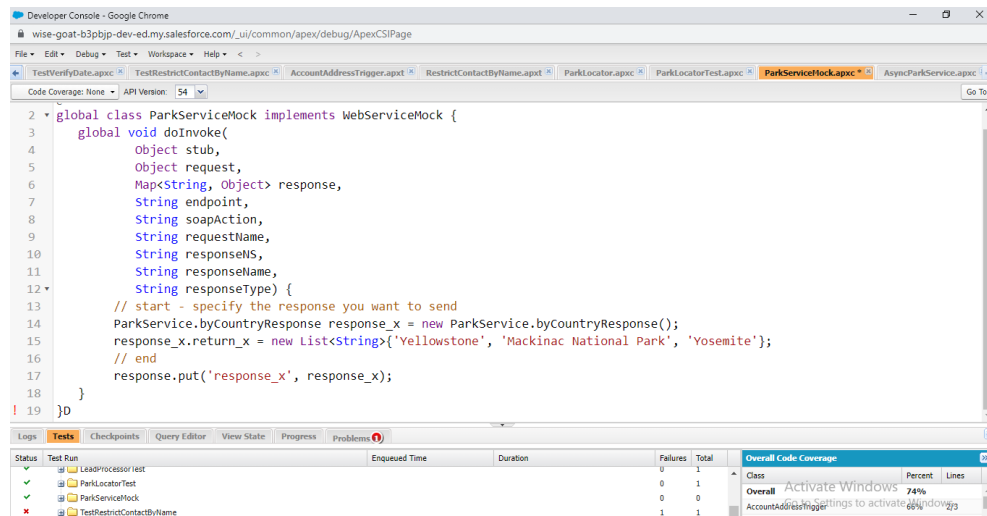
response_x.return_x = new List<String>{'Yellowstone', 'Mackinac National Park', 'Yosemite'};

// end

response.put('response_x', response_x);

}

}



AsyncParkService.apxc

//Generated by wsdl2apex

```
public class AsyncParkService {

    public class byCountryResponseFuture extends System.WebServiceCalloutFuture {

        public String[] getValue() {

            ParkService.byCountryResponse response =
(ParkService.byCountryResponse)System.WebServiceCallout.endInvoke(this);

            return response.return_x;

        }

    }

    public class AsyncParksImplPort {

        public String endpoint_x = 'https://th-apex-soap-service.herokuapp.com/service/parks';

        public Map<String,String> inputHttpHeaders_x;

        public String clientCertName_x;

        public Integer timeout_x;

        private String[] ns_map_type_info = new String[]{ 'http://parks.services/', 'ParkService';
```

```
public AsyncParkService.byCountryResponseFuture beginByCountry(System.Continuation continuation,String
arg0) {

    ParkService.byCountry request_x = new ParkService.byCountry();

    request_x.arg0 = arg0;

    return (AsyncParkService.byCountryResponseFuture) System.WebServiceCallout.beginInvoke(

        this,

        request_x,

        AsyncParkService.byCountryResponseFuture.class,

        continuation,

        new String[]{endpoint_x,

            "",

            'http://parks.services/',

            'byCountry',

            'http://parks.services/',

            'byCountryResponse',

            'ParkService.byCountryResponse'}

        );

    }

}
```

```

1 //Generated by wsdl2apex
2
3 public class AsyncParkService {
4     public class byCountryResponseFuture extends System.WebServiceCalloutFuture {
5         public String[] getValue() {
6             ParkService.byCountryResponse response = (ParkService.byCountryResponse)System.WebServiceCallout.endInvoke(this);
7             return response.return_x;
8         }
9     }
10    public class AsyncParksImplPort {
11        public String endpoint_x = 'https://th-apex-soap-service.herokuapp.com/service/parks';
12        public Map<String,String> inputHttpHeaders_x;
13        public String clientCertName_x;
14        public Integer timeout_x;
15        private String[] ns_map_type_info = new String[]{'http://parks.services/', 'ParkService'};
16        public AsyncParkService.byCountryResponseFuture beginByCountry(System.Continuation continuation,String arg0) {
17            ParkService.byCountry request_x = new ParkService.byCountry();
18            request_x.arg0 = arg0;
19        }
20    }
21 }

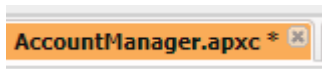
```

Status	Test Run	Enqueued Time	Duration	Failures	Total
✗	7071w0000004Gje	Tue May 31 2022 03:03:41 GM...		0	1
✗	7071w0000004GK3	Tue May 31 2022 03:04:03 GM...		0	1
✓	Dashboard_Pal.DashboardPalTest			0	1
✓	trilhdtps_tt_UtilControllerTest			0	2
✓	AccountProcessorTest			0	1

Class	Percent	Lines
Overall	74%	
AccountAddressTriggerSettings to activate	66%	2/3
AccountProcessor	100%	8/8

Apex Integration Services

Apex Web Services



@RestResource(urlMapping='/Accounts/*/contacts')

global class AccountManager {

@HttpGet

global static Account getAccount() {

RestRequest req = RestContext.request;

String acctId = req.requestURI.substringBetween('Accounts/', '/contacts');

Account acc = [SELECT Id, Name, (SELECT Id, Name FROM Contacts)

FROM Account WHERE Id = :acctId];

return acc;

}

}

Developer Console - Google Chrome

wise-goat-b3pbjp-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help < >

rtunityTrigger.apxt VerifyDate.apxt TestVerifyDate.apxt TestRestrictContactByName.apxt AccountAddressTrigger.apxt RestrictContactByName.apxt **AccountManager.apxt** AccountManagerTest.apxt

Code Coverage: None API Version: 54 Go To

```

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

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	trifidtops.tt.UtilControllerTest			0	2
✓	AccountManagerTest			0	1
✓	AccountProcessorTest			0	1
✓	AddPrimaryContactTest			0	1
✓	AnimalLocatorHook			0	0
✗	AnimalLocatorTest			1	1
✓	DailyLeadProcessorTest			0	1

Overall Code Coverage

Class	Percent	Lines
Overall	75%	
AccountAddressTrigger	66%	2/3
AccountManager	100%	5/5
AccountProcessor	100%	8/8
AddPrimaryContact	100%	13/13
AnimalLocator	100%	11/11

AccountManagerTest.apxt

@isTest

```
private class AccountManagerTest {
```

```
    private static testMethod void getAccountTest1() {
```

```
        Id recordId = createTestRecord();
```

```
        // Set up a test request
```

```
        RestRequest request = new RestRequest();
```

```
        request.requestUri = 'https://na1.salesforce.com/services/apexrest/Accounts/' + recordId + '/contacts';
```

```
        request.httpMethod = 'GET';
```

```
        RestContext.request = request;
```

```
        // Call the method to test
```

```
        Account thisAccount = AccountManager.getAccount();
```

```
        // Verify results
```

```
        System.assert(thisAccount != null);
```

```
System.assertEquals('Test record', thisAccount.Name);
```

```
}
```

```
// Helper method
```

```
static Id createTestRecord() {
```

```
// Create test record
```

```
Account TestAcc = new Account(
```

```
    Name='Test record');
```

```
insert TestAcc;
```

```
Contact TestCon= new Contact(
```

```
    LastName='Test',
```

```
    AccountId = TestAcc.id;
```

```
return TestAcc.id;
```

```
}
```

```
}
```

The screenshot displays the Salesforce Developer Console interface. The top pane shows the source code for a test class named `AccountManagerTest`. The code includes a `@isTest` annotation, a `private class` definition, and a `private static testMethod` named `getAccountTest1`. This method performs several steps: it calls `createTestRecord()` to get an `Id`, sets up a `RestRequest` with a specific URI and method, calls the `AccountManager.getAccount()` method, and then uses `System.assertEquals` to verify the results.

The bottom pane shows the 'Tests' tab with a table of test results. The table has columns for Status, Test Run, Enqueued Time, Duration, Failures, and Total. All tests listed are successful (Status: green checkmark). To the right of the table, the 'Overall Code Coverage' is shown as 75%.


Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	trhdps_@_UIControllerTest			0	2
✓	AccountManagerTest			0	1
✓	AccountProcessorTest			0	1
✓	AddPrimaryContactTest			0	1
✓	AnimalLocatorMock			0	0
✓	AnimalLocatorTest			1	1
✓	DailyLeadProcessorTest			0	1

Overall Code Coverage: 75%

Class	Percent	Lines
Overall	75%	
AccountAddressTrigger	66%	2/3
AccountManager	100%	5/5
AccountProcessor	100%	8/8
AddPrimaryContact	100%	13/13
AnimalLocator	100%	11/11

SUPERBADGE:APEX SPECIALIST

Step2: Automate record creation

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 Maintenance'){

                    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));
```

```
    } else {
```

```
        nc.Date_Due__c = Date.today().addDays((Integer) cc.Equipment__r.maintenance_Cycle__c);
```

```
    }
```



```

        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;

}

}

}

```

MaintenanceRequest.apxt 

```

trigger MaintenanceRequest on Case (before update, after update) {

```

```

    if(Trigger.isUpdate && Trigger.isAfter){

```

```
MaintenanceRequestHelper.updateWorkOrders(Trieger.New, Trieger.OldMap);
```

```
}
```

```
}
```

Step3: Synchronize Salesforce data with an external system

WarehouseCalloutService.apxc

```
public with sharing class WarehouseCalloutService implements Queueable {
```

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

```
    //class that makes a REST callout to an external warehouse system to get a list of  
    equipment that needs to be updated.
```

```
    //The callout's JSON response returns the equipment records that you upsert in  
    Salesforce.
```

```
@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());
```

```
    //class maps the following fields: replacement part (always true), cost, current  
inventory, lifespan, maintenance cycle, and warehouse SKU
```

```
    //warehouse SKU will be external ID for identifying which equipment records to  
update within Salesforce
```

```
    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 = (Integer) mapJson.get('cost');
```

```
        myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
```

```
        myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
```

```
        myEq.ProductCode = (String) mapJson.get('_id');
```

```
        warehouseEq.add(myEq);
```

```

    }

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

        System.debug('Your equipment was synced with the warehouse one');
    }
}

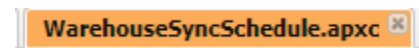
}

public static void execute (QueueableContext context){
    runWarehouseEquipmentSync();
}

}

```

Step4: Schedule synchronization

 WarehouseSyncSchedule.apxc

```

global with sharing class WarehouseSyncSchedule implements Schedulable{

    global void execute(SchedulableContext ctx){

        System.enqueueJob(new WarehouseCalloutService());

    }
}

```

```
}
```

Step 5: Test automation logic

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 Maintenance'){

                    validIds.add(c.Id);

                }

            }

        }

    }

    //When an existing maintenance request of type Repair or Routine Maintenance is
closed,

    //create a new maintenance request for a future routine checkup.

    if (!validIds.isEmpty()){

        Map<Id,Case> closedCases = 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>();
```

```
//calculate the maintenance request due dates by using the maintenance cycle  
defined on the related equipment records.
```

```
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'));  
  
}
```

```
List<Case> newCases = new List<Case>();
```

```
for(Case cc : closedCases.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 multiple pieces of equipment are used in the maintenance request,

    //define the due date by applying the shortest maintenance cycle to today's date.

    //If (maintenanceCycles.containsKey(cc.Id)){

        nc.Date_Due__c = Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));

    //} else {

        //  nc.Date_Due__c = Date.today().addDays((Integer)
cc.Equipment__r.maintenance_Cycle__c);

    //}

    newCases.add(nc);

}

insert newCases;

List<Equipment_Maintenance_Item__c> clonedList = new
List<Equipment_Maintenance_Item__c>();

for (Case nc : newCases){

    for (Equipment_Maintenance_Item__c clonedListItem :
closedCases.get(nc.ParentId).Equipment_Maintenance_Items__r){

```

```

        Equipment_Maintenance_Item__c item = clonedListItem.clone();

        item.Maintenance_Request__c = nc.Id;

        clonedList.add(item);

    }

}

insert clonedList;

}

}

}

```

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';

PRIVATE STATIC Vehicle__c createVehicle(){


```
Vehicle__c Vehicle = new Vehicle__C(name = 'SuperTruck');  
  
return Vehicle;  
  
}
```

```
PRIVATE STATIC Product2 createEq(){  
  
    product2 equipment = new product2(name = 'SuperEquipment',  
  
        lifespan_months__C = 10,  
  
        maintenance_cycle__C = 10,  
  
        replacement_part__c = true);  
  
    return equipment;  
  
}
```

```
PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id equipmentId){  
  
    case cs = new case(Type=REPAIR,  
  
        Status=STATUS_NEW,  
  
        Origin=REQUEST_ORIGIN,  
  
        Subject=REQUEST_SUBJECT,  
  
        Equipment__c=equipmentId,  
  
        Vehicle__c=vehicleId);  
  
    return cs;  
  
}
```

```
PRIVATE STATIC Equipment_Maintenance_Item__c createWorkPart(id equipmentId,id  
requestId){
```

```
    Equipment_Maintenance_Item__c wp = new  
    Equipment_Maintenance_Item__c(Equipment__c = equipmentId,
```

```
                                Maintenance_Request__c = requestId);
```

```
    return wp;
```

```
}
```

```
@istest
```

```
private static void testMaintenanceRequestPositive(){
```

```
    Vehicle__c vehicle = createVehicle();
```

```
    insert vehicle;
```

```
    id vehicleId = vehicle.Id;
```

```
    Product2 equipment = createEq();
```

```
    insert equipment;
```

```
    id equipmentId = equipment.Id;
```

```
    case somethingToUpdate = createMaintenanceRequest(vehicleId,equipmentId);
```

```
    insert somethingToUpdate;
```

```
    Equipment_Maintenance_Item__c workP =  
    createWorkPart(equipmentId,somethingToUpdate.id);
```

```
insert workP;
```

```
test.startTest();
```

```
somethingToUpdate.status = CLOSED;
```

```
update somethingToUpdate;
```

```
test.stopTest();
```

```
Case newReq = [Select id, subject, type, Equipment__c, Date_Reported__c, Vehicle__c,  
Date_Due__c
```

```
from case
```

```
where status =:STATUS_NEW];
```

```
Equipment_Maintenance_Item__c workPart = [select id
```

```
from Equipment_Maintenance_Item__c
```

```
where Maintenance_Request__c =:newReq.Id];
```

```
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);
```

```
SYSTEM.assertEquals(newReq.Date_Reported__c, system.today());
```

```
}
```

@istest

private static void testMaintenanceRequestNegative(){

Vehicle__C vehicle = createVehicle();

insert vehicle;

id vehicleId = vehicle.Id;

product2 equipment = createEq();

insert equipment;

id equipmentId = equipment.Id;

case emptyReq = createMaintenanceRequest(vehicleId,equipmentId);

insert emptyReq;

Equipment_Maintenance_Item__c workP = createWorkPart(equipmentId, emptyReq.Id);

insert workP;

test.startTest();

emptyReq.Status = WORKING;

update emptyReq;

test.stopTest();

```
list<case> allRequest = [select id  
                        from case];
```

```
Equipment_Maintenance_Item__c workPart = [select id  
                                           from Equipment_Maintenance_Item__c  
                                           where Maintenance_Request__c = :emptyReq.Id];
```

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

@istest

```
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  
list<Equipment_Maintenance_Item__c>();  
  
    list<case> requestList = new list<case>();  
  
    list<id> oldRequestIds = new list<id>();  
  
    for(integer i = 0; i < 300; i++){  
  
        vehicleList.add(createVehicle());  
  
        equipmentList.add(createEq());  
  

```

```
}

insert vehicleList;

insert equipmentList;


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

    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);
```

```
}
```

```
}
```

MaintenanceRequest.apxt

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

Step 6: Test callout logic

WarehouseCalloutServiceMock.apxc

```
@isTest
```

```
global class WarehouseCalloutServiceMock implements HttpCalloutMock {
```

```
    // implement http mock callout
```

```
    global static HttpResponse respond(HttpRequest request) {
```

```

    HttpResponse response = new HttpResponse();

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

    response.setBody(['{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5,"name":"Generator 1000 kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"},{"_id":"55d66226726b611100aaf742","replacement":true,"quantity":183,"name":"Cooling Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004"},{"_id":"55d66226726b611100aaf743","replacement":true,"quantity":143,"name":"Fuse 20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"}]);

    response.setStatusCode(200);

    return response;
}
}

```

WarehouseCalloutServiceTest.apxc

```

@Test
private class WarehouseCalloutServiceTest {

    // implement your mock callout test here

    @isTest
    static void testWarehouseCallout() {

        test.startTest();

        test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());

        WarehouseCalloutService.execute(null);
    }
}

```



```
test.stopTest();
```

```
List<Product2> product2List = new List<Product2>();
```

```
product2List = [SELECT ProductCode FROM Product2];
```

```
System.assertEquals(3, product2List.size());
```

```
System.assertEquals('55d66226726b611100aaf741', product2List.get(0).ProductCode);
```

```
System.assertEquals('55d66226726b611100aaf742', product2List.get(1).ProductCode);
```

```
System.assertEquals('55d66226726b611100aaf743', product2List.get(2).ProductCode);
```

```
}
```

```
}
```

Step 7: Test scheduling logic

WarehouseSyncSchedule.apxc

```
global with sharing class WarehouseSyncSchedule implements Schedulable{
```

```
    global void execute(SchedulableContext ctx){
```

```
        System.enqueueJob(new WarehouseCalloutService());
```

```
    }
```

```
}
```

WarehouseCalloutServiceMock.apxc

```
@isTest
```

```
global class WarehouseCalloutServiceMock implements HttpCalloutMock {
```

```
// implement http mock callout

global static HttpResponse respond(HttpRequest request) {

    HttpResponse response = new HttpResponse();

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

    response.setBody('[{ "_id": "55d66226726b611100aaf741", "replacement": false, "quantity": 5, "name": "Generator 1000 kW", "maintenanceperiod": 365, "lifespan": 120, "cost": 5000, "sku": "100003"}, {" _id": "55d66226726b611100aaf742", "replacement": true, "quantity": 183, "name": "Cooling Fan", "maintenanceperiod": 0, "lifespan": 0, "cost": 300, "sku": "100004"}, {" _id": "55d66226726b611100aaf743", "replacement": true, "quantity": 143, "name": "Fuse 20A", "maintenanceperiod": 0, "lifespan": 0, "cost": 22, "sku": "100005"}]');

    response.setStatusCode(200);

    return response;

}

}
```

WarehouseSyncScheduleTest.apxc

```
@isTest

public with sharing class WarehouseSyncScheduleTest {

    // implement scheduled code here

    //

    @isTest static void test() {

        String scheduleTime = '00 00 00 * * ? *';
```

```
Test.startTest();
```

```
Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
```

```
String jobId = System.schedule('Warehouse Time to Schedule to test', scheduleTime,  
new WarehouseSyncSchedule());
```

```
CronTrigger c = [SELECT State FROM CronTrigger WHERE Id =: jobId];
```

```
System.assertEquals('WAITING', String.valueOf(c.State), 'JobId does not match');
```

```
Test.stopTest();
```

```
}
```

```
}
```

