```
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
1
APEX SPECIALIST SUPER BADGE CODES
APEX TRIGGERS
AccountAddressTrigger.axpt:
trigger AccountAddressTrigger on Account (before insert,before
update) {
for(Account account:Trigger.New){
if(account.Match_Billing_Address__c == True){
account.ShippingPostalCode = account.BillingPostalCode;
}
}
}
ClosedOpportunityTrigger.axpt:
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;
}
}
APEX TESTING
VerifyData.apxc:
public class VerifyDate {
public static Date CheckDates(Date date1, Date date2) {
if(DateWithin30Days(date1,date2)) {
return date2;
} else {
return SetEndOfMonthDate(date1);
}
}
@TestVisible private static Boolean DateWithin30Days(Date date1,
```

```
   Date
40 date2) {
41 //check for date2 being in the past
42 if( date2 < date1) { return false; }
43 SPSGP-90546-Salesforce Developer Catalyst
44 Self-Learning & Super Badges
45 2
46 APEX SPECIALIST SUPER BADGE CODES
47 //check that date2 is within (>=) 30 days of date1
48 Date date30Days = date1.addDays(30); //create a date 30 days away
49 from date1
50 if( date2 >= date30Days ) { return false; }
51 else { return true; }
52 }
53 //method to return the end of the month of a given date
54 @TestVisible private static Date SetEndOfMonthDate(Date date1) {
55 Integer totalDays = Date.daysInMonth(date1.year(),
   date1.month());
56 Date lastDay = Date.newInstance(date1.year(), date1.month(),
57 totalDays);
58 return lastDay;
59 }
60 }
61 TestVerifyData.apxc:
62 @isTest
63 private class TestVerifyDate {
64 @isTest static void Test_CheckDates_case1(){
65 Date D =
66 VerifyDate.CheckDates(date.parse('01/01/2022'),date.parse('01/05/
   2022
67 System.assertEquals(date.parse('01/05/2022'), D);
68 }
69 @isTest static void Test_CheckDates_case2(){
70 Date D = VerifyDate.CheckDates(date.parse('01/01/2022'),
71 date.parse('05/05/2022'));
72 System.assertEquals(date.parse('01/31/2022'), D);
73 }
74 @isTest static void Test_Within30Days_case1(){
75 Boolean flag =
   VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
```

```
76 date.parse('12/30/2021'));
77 System.assertEquals(false, flag);
78 }
79 @isTest static void Test_Within30Days_case2(){
80 Boolean flag =
   VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
81 date.parse('02/02/2021'));
82 System.assertEquals(false, flag);
83 }
84 @isTest static void Test_Within30Days_case3(){
85 SPSGP-90546-Salesforce Developer Catalyst
86 Self-Learning & Super Badges
87 3
88 APEX SPECIALIST SUPER BADGE CODES
89 Boolean flag =
   VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
90 date.parse('01/15/2022'));
91 System.assertEquals(true, flag);
92 }
93 @isTest static void Test_SetEndOfMonthDate(){
94 Date returndate =
95 VerifyDate.SetEndOfMonthDate(date.parse('01/01/2022'));
96 }
97 }
98 RestrictContactByName.apxt:
99 trigger RestrictContactByName on Contact (before insert, before
100 update) {
101 //check contacts prior to insert or update for invalid data
102 For (Contact c : Trigger.New) {
103 if(c.LastName == 'INVALIDNAME') { //invalidname is invalid
104 c.AddError('The Last Name "'+c.LastName+'" is not allowed for

105 }
106 }
107 }
108 TestRestrictContactByName.apxc:
109 @isTest
110 private class TestRestrictContactByName {
111 @isTest static void Test_insertupdateContact(){
112 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
result.getErrors()[0].getMessage());
}
}
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
4
APEX SPECIALIST SUPER BADGE CODES
RandomContactFactory.apxc:
public class RandomContactFactory {
public static List<Contact> generateRandomContacts(Integer num_cnts,
string lastname) {
List<Contact> contacts = new List<Contact>();
for(Integer i = 0; i < num_cnts; i++) {
Contact cnt = new Contact(FirstName = 'Test' +i,LastName = lastname);
contacts.add(cnt);
}
return contacts;
}
}
ASYNCHRONOUS APEX
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;
}
}
AccountProcessorTest.apxc:
@isTest
public class AccountProcessorTest {
@isTest
private static void testCountContacts() {
Account newAccount = new Account(Name = 'Test Account');
insert newAccount;
Contact newContact1 = new Contact(FirstName = 'John',LastName =
'Doe',AccountId =
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
5
APEX SPECIALIST SUPER BADGE CODES
newAccount.Id);
insert newContact1;
Contact newContact2 = new Contact(FirstName = 'John',LastName =
'Doe',AccountId =
newAccount.Id);
insert newContact2;
List<Id> accountIds = new List<Id>();
accountIds.add(newAccount.Id);
Test.startTest();
AccountProcessor.countContacts(accountIds);
Test.stopTest();
}
}
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
```

```
187 }
188 global void execute(Database.BatchableContext bc, List<Lead>
   L_list){
189 List<lead> L_list_new = new List<lead>();
190 for(lead L: L_list){
191 L.leadSource = 'Dreamforce';
192 L_list_new.add(L);
193 count += 1;
194 }
195 update L_list_new;
196 }
197 global void finish(Database.BatchableContext bc){
198 system.debug('count = ' + count);
199 }
200 }
201 LeadProcessorTest.apxc:
202 @isTest
203 public class LeadProcessorTest {
204 @isTest
205 public static void testit() {
206 SPSGP-90546-Salesforce Developer Catalyst
207 Self-Learning & Super Badges
208 6
209 APEX SPECIALIST SUPER BADGE CODES
210 List<lead> L_list = new List<lead>();
211 for(Integer i = 0; i < 200; i++) {
212 Lead L = new Lead();
213 L.LastName = 'name' + i;
214 L.Company = 'Company';
215 L.Status = 'Random Status';
216 L_list.add(L);
217 }
218 insert L_list;
219 Test.startTest();
220 LeadProcessor lp = new LeadProcessor();
221 Id batchId = Database.executeBatch(lp);
222 Test.stopTest();
223 }
224 }
225 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,LastName,
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;
}
}
}
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
7
APEX SPECIALIST SUPER BADGE CODES
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')]);
}
}
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;
}
}
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
8
APEX SPECIALIST SUPER BADGE CODES
DailyLeadProcessorTest.apxc:
@isTest
private class DailyLeadProcessorTest {
public static String CRON_EXP = '0 0 0 15 3 ? 2024';
static testmethod void testScheduledJob() {
List<Lead> leads = new List<Lead>();
```

```
301 for(Integer i = 0; i < 200; i++) {
302 Lead l = new Lead(
303 FirstName = 'First' + i,
304 LastName = 'LastName',
305 Company = 'The Inc'
306 );
307 leads.add(l);
308 }
309 insert leads;
310 Test.startTest();
311 String jobId = System.schedule('ScheduledApexTest',CRON_EXP,new
312 DailyLeadProcessor());
313 Test.stopTest();
314 List<Lead> checkleads = new List<Lead>();
315 checkleads = [Select Id From Lead Where LeadSource =
   'Dreamforce' and
316 Company = 'The Inc'];
317 System.assertEquals(200,checkleads.size(),'Leads were not

318 }
319 }
320 APEX INTEGRATION SERVICES
321 AnimalLocator.apxc:
322 public class AnimalLocator{
323 public static String getAnimalNameById(Integer x){
324 Http http = new Http();
325 HttpRequest req = new HttpRequest();
326 req.setEndpoint('https://th-apex-http-
   callout.herokuapp.com/animals/'
327 + x);
328 req.setMethod('GET');
329 Map<String, Object> animal= new Map<String, Object>();
330 HttpResponse res = http.send(req);
331 if (res.getStatusCode() == 200) {
332 SPSGP-90546-Salesforce Developer Catalyst
333 Self-Learning & Super Badges
334 9
335 APEX SPECIALIST SUPER BADGE CODES
336 Map<String, Object> results = (Map<String,
337 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
global class AnimalLocatorMock implements HttpCalloutMock {
// Implement this interface method
global HTTPResponse respond(HTTPRequest request) {
// Create a fake response
HttpResponse response = new HttpResponse();
response.setHeader('Content-Type', 'application/json');
response.setBody('{"animals": ["majestic badger", "fluffy
  bunny",
"scary bear", "chicken", "mighty
moose"]}');
response.setStatusCode(200);
return response;
}
}
ParkLocator.apxc:
public class ParkLocator {
public static string[] country(string theCountry) {
ParkService.ParksImplPort parkSvc = new
  ParkService.ParksImplPort();
// remove space
return parkSvc.byCountry(theCountry);
}
}
```

```
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
10
APEX SPECIALIST SUPER BADGE CODES
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
System.assertEquals(parks, result);
}
}
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
// end
response.put('response_x', response_x);
}
}
AccountManager.apxc:
@RestResource(urlMapping='/Accounts/*/contacts')
global class AccountManager {
@HttpGet
```

```
global static Account getAccount() {
RestRequest req = RestContext.request;
String accId = req.requestURI.substringBetween('Accounts/',
'/contacts');
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
11
APEX SPECIALIST SUPER BADGE CODES
Account acc = [SELECT Id, Name, (SELECT Id, Name FROM Contacts)
FROM Account WHERE Id = :accId];
return acc;
}
}
AccountManagerTest.apxc:
@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;
}
}
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
12
APEX SPECIALIST SUPER BADGE CODES
APEX SPECIALIST SUPER BADGE
Challenge-1
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',
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
13
APEX SPECIALIST SUPER BADGE CODES
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;
}
}
}
SPSGP-90546-Salesforce Developer Catalyst
```

```
Self-Learning & Super Badges
14
APEX SPECIALIST SUPER BADGE CODES
MaintenanceRequest.apxt:
trigger MaintenanceRequest on Case (before update, after update)
  {
if(Trigger.isUpdate && Trigger.isAfter){
MaintenanceRequestHelper.updateWorkOrders(Trigger.New,
Trigger.OldMap);
}
}
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

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,
```

```
570 SPSGP-90546-Salesforce Developer Catalyst
571 Self-Learning & Super Badges
572 15
573 APEX SPECIALIST SUPER BADGE CODES
574 Vehicle__c=vehicleId);
575 return cs;
576 }
577 PRIVATE STATIC Equipment_Maintenance_Item__c createWorkPart(id
578 equipmentId,id requestId){
579 Equipment_Maintenance_Item__c wp = new
580 Equipment_Maintenance_Item__c(Equipment__c =
581 equipmentId,
582 Maintenance_Request__c = requestId);
583 return wp;
584 }
585 @istest
586 private static void testMaintenanceRequestPositive(){
587 Vehicle__c vehicle = createVehicle();
588 insert vehicle;
589 id vehicleId = vehicle.Id;
590 Product2 equipment = createEq();
591 insert equipment;
592 id equipmentId = equipment.Id;
593 case somethingToUpdate =
594 createMaintenanceRequest(vehicleId,equipmentId);
595 insert somethingToUpdate;
596 Equipment_Maintenance_Item__c workP =
597 createWorkPart(equipmentId,somethingToUpdate.id);
598 insert workP;
599 test.startTest();
600 somethingToUpdate.status = CLOSED;
601 update somethingToUpdate;
602 test.stopTest();
603 Case newReq = [Select id, subject, type, Equipment__c,
604 Date_Reported__c, Vehicle__c,
605 Date_Due__c
606 from case
607 where status =:STATUS_NEW];
608 SPSGP-90546-Salesforce Developer Catalyst
609 Self-Learning & Super Badges
```

```
610 16
611 APEX SPECIALIST SUPER BADGE CODES
612 Equipment_Maintenance_Item__c workPart = [select id
613 from Equipment_Maintenance_Item__c
614 where Maintenance_Request__c =:newReq.Id];
615 system.assert(workPart != null);
616 system.assert(newReq.Subject != null);
617 system.assertEquals(newReq.Type, REQUEST_TYPE);
618 SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
619 SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
620 SYSTEM.assertEquals(newReq.Date_Reported__c, system.today());
621 }
622 @istest
623 private static void testMaintenanceRequestNegative(){
624 Vehicle__C vehicle = createVehicle();
625 insert vehicle;
626 id vehicleId = vehicle.Id;
627 product2 equipment = createEq();
628 insert equipment;
629 id equipmentId = equipment.Id;
630 case emptyReq = createMaintenanceRequest(vehicleId,equipmentId);
631 insert emptyReq;
632 Equipment_Maintenance_Item__c workP =
   createWorkPart(equipmentId,
633 emptyReq.Id);
634 insert workP;
635 test.startTest();
636 emptyReq.Status = WORKING;
637 update emptyReq;
638 test.stopTest();
639 list<case> allRequest = [select id
640 from case];
641 Equipment_Maintenance_Item__c workPart = [select id
642 from Equipment_Maintenance_Item__c
643 SPSGP-90546-Salesforce Developer Catalyst
644 Self-Learning & Super Badges
645 17
646 APEX SPECIALIST SUPER BADGE CODES
647 where Maintenance_Request__c = :emptyReq.Id];
648 system.assert(workPart != null);
```

```
649 system.assert(allRequest.size() == 1);
650 }
651 @istest
652 private static void testMaintenanceRequestBulk(){
653 list<Vehicle__C> vehicleList = new list<Vehicle__C>();
654 list<Product2> equipmentList = new list<Product2>();
655 list<Equipment_Maintenance_Item__c> workPartList = new
656 list<Equipment_Maintenance_Item__c>();
657 list<case> requestList = new list<case>();
658 list<id> oldRequestIds = new list<id>();
659 for(integer i = 0; i < 300; i++){
660 vehicleList.add(createVehicle());
661 equipmentList.add(createEq());
662 }
663 insert vehicleList;
664 insert equipmentList;
665 for(integer i = 0; i < 300; i++){
666 requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
667 equipmentList.get(i).id));
668 }
669 insert requestList;
670 for(integer i = 0; i < 300; i++){
671 workPartList.add(createWorkPart(equipmentList.get(i).id,
672 requestList.get(i).id));
673 }
674 insert workPartList;
675 test.startTest();
676 for(case req : requestList){
677 req.Status = CLOSED;
678 oldRequestIds.add(req.Id);
679 }
680 update requestList;
681 SPSGP-90546-Salesforce Developer Catalyst
682 Self-Learning & Super Badges
683 18
684 APEX SPECIALIST SUPER BADGE CODES
685 test.stopTest();
686 list<case> allRequests = [select id
687 from case
688 where status =: STATUS_NEW];
```

```
689 list<Equipment_Maintenance_Item__c> workParts = [select id
690 from Equipment_Maintenance_Item__c
691 where Maintenance_Request__c in: oldRequestIds];
692 system.assert(allRequests.size() == 300);
693 }
694 }
695 Challenge-2
696 WarehouseCalloutService.apxc:
697 public with sharing class WarehouseCalloutService implements
698 Queueable {
699 private static final String WAREHOUSE_URL = 'https://th-
700 //class that makes a REST callout to an external warehouse
   system to
701 get a list of equipment that
702 needs to be updated.
703 //The callout's JSON response returns the equipment records that
   you
704 upsert in Salesforce.
705 @future(callout=true)
706 public static void runWarehouseEquipmentSync(){
707 Http http = new Http();
708 HttpRequest request = new HttpRequest();
709 request.setEndpoint(WAREHOUSE_URL);
710 request.setMethod('GET');
711 HttpResponse response = http.send(request);
712 List<Product2> warehouseEq = new List<Product2>();
713 if (response.getStatusCode() == 200){
714 List<Object> jsonResponse =
715 (List<Object>)JSON.deserializeUntyped(response.getBody());
716 SPSGP-90546-Salesforce Developer Catalyst
717 Self-Learning & Super Badges
718 19
719 APEX SPECIALIST SUPER BADGE CODES
720 System.debug(response.getBody());
721 //class maps the following fields: replacement part (always
   true),
722 cost, current inventory,
723 lifespan, maintenance cycle, and warehouse SKU
724 //warehouse SKU will be external ID for identifying which
   equipment
```

```
725 records to update within
726 Salesforce
727 for (Object eq : jsonResponse){
728 Map<String,Object> mapJson = (Map<String,Object>)eq;
729 Product2 myEq = new Product2();
730 myEq.Replacement_Part__c = (Boolean) mapJson.get('replacement');
731 myEq.Name = (String) mapJson.get('name');
732 myEq.Maintenance_Cycle__c = (Integer)
733 mapJson.get('maintenanceperiod');
734 myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
735 myEq.Cost__c = (Integer) mapJson.get('cost');
736 myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
737 myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
738 myEq.ProductCode = (String) mapJson.get('_id');
739 warehouseEq.add(myEq);
740 }
741 if (warehouseEq.size() > 0){
742 upsert warehouseEq;
743 System.debug('Your equipment was synced with the warehouse

744 }
745 }
746 }
747 public static void execute (QueueableContext context){
748 runWarehouseEquipmentSync();
749 }
750 }
751 WarehouseCalloutServiceMock.apxc:
752 @isTest
753 global class WarehouseCalloutServiceMock implements
   HttpCalloutMock {
754 // implement http mock callout
755 global static HttpResponse respond(HttpRequest request) {
756 SPSGP-90546-Salesforce Developer Catalyst
757 Self-Learning & Super Badges
758 20
759 APEX SPECIALIST SUPER BADGE CODES
760 HttpResponse response = new HttpResponse();
761 response.setHeader('Content-Type', 'application/json');
762 response.setBody('[{"_id":"55d66226726b611100aaf741","replacemen
```

```
   t":fa
763 lse,"quantity":5,"name":"Gene
764 rator 1000
765 kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10

766 af742","replacement":true,"quantity":183,"name":"Cooling
767 Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004
   "},{"
768 _id":"55d66226726b611100aaf743
769 ","replacement":true,"quantity":143,"name":"Fuse
770 20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"
   }]');
771 response.setStatusCode(200);
772 return response;
773 }
774 }
775 WarehouseCalloutServiceTest.apxc:
776 @IsTest
777 private class WarehouseCalloutServiceTest {
778 // implement your mock callout test here
779 @isTest
780 static void testWarehouseCallout() {
781 test.startTest();
782 test.setMock(HttpCalloutMock.class, new
783 WarehouseCalloutServiceMock());
784 WarehouseCalloutService.execute(null);
785 test.stopTest();
786 List<Product2> product2List = new List<Product2>();
787 product2List = [SELECT ProductCode FROM Product2];
788 System.assertEquals(3, product2List.size());
789 System.assertEquals('55d66226726b611100aaf741',
790 product2List.get(0).ProductCode);
791 System.assertEquals('55d66226726b611100aaf742',
792 product2List.get(1).ProductCode);
793 System.assertEquals('55d66226726b611100aaf743',
794 product2List.get(2).ProductCode);
795 }
796 }
797 Challenge-3
798 WarehouseSyncSchedule.apxc:
```

```
global with sharing class WarehouseSyncSchedule implements
Schedulable{
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
21
APEX SPECIALIST SUPER BADGE CODES
global void execute(SchedulableContext ctx){
System.enqueueJob(new WarehouseCalloutService());
}
}
WarehouseSyncScheduuleTest.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 to

scheduleTime, new
WarehouseSyncSchedule());
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 ');
}
}
Challenge-4
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';
```

```
837 private static final string REPAIR = 'Repair';
838 private static final string REQUEST_ORIGIN = 'Web';
839 private static final string REQUEST_TYPE = 'Routine

840 private static final string REQUEST_SUBJECT = 'Testing subject';
841 PRIVATE STATIC Vehicle__c createVehicle(){
842 SPSGP-90546-Salesforce Developer Catalyst
843 Self-Learning & Super Badges
844 22
845 APEX SPECIALIST SUPER BADGE CODES
846 Vehicle__c Vehicle = new Vehicle__C(name = 'SuperTruck');
847 return Vehicle;
848 }
849 PRIVATE STATIC Product2 createEq(){
850 product2 equipment = new product2(name = 'SuperEquipment',
851 lifespan_months__C = 10,
852 maintenance_cycle__C = 10,
853 replacement_part__c = true);
854 return equipment;
855 }
856 PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id
857 equipmentId){
858 case cs = new case(Type=REPAIR,
859 Status=STATUS_NEW,
860 Origin=REQUEST_ORIGIN,
861 Subject=REQUEST_SUBJECT,
862 Equipment__c=equipmentId,
863 Vehicle__c=vehicleId);
864 return cs;
865 }
866 PRIVATE STATIC Equipment_Maintenance_Item__c createWorkPart(id
867 equipmentId,id requestId){
868 Equipment_Maintenance_Item__c wp = new
869 Equipment_Maintenance_Item__c(Equipment__c =
870 equipmentId, Maintenance_Request__c = requestId);
871 return wp;
872 }
873 @istest
874 private static void testMaintenanceRequestPositive(){
875 Vehicle__c vehicle = createVehicle();
```

```
insert vehicle;
id vehicleId = vehicle.Id;
Product2 equipment = createEq();
insert equipment;
id equipmentId = equipment.Id;
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
23
APEX SPECIALIST SUPER BADGE CODES
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;
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
24
APEX SPECIALIST SUPER BADGE CODES
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;
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
```

```
25
APEX SPECIALIST SUPER BADGE CODES
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);
}
}
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'){
SPSGP-90546-Salesforce Developer Catalyst
Self-Learning & Super Badges
26
APEX SPECIALIST SUPER BADGE CODES
```

```
995 if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
996 validIds.add(c.Id);
997 }
998 }
999 }
1000if (!validIds.isEmpty()){
1001List<Case> newCases = new List<Case>();
1002Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id,
   Vehicle__c,
1003Equipment__c,
1004Equipment__r.Maintenance_Cycle__c,(SELECT
   Id,Equipment__c,Quantity__c
1005FROM
1006Equipment_Maintenance_Items__r)
1007FROM Case WHERE Id IN :validIds]);
1008Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
1009AggregateResult[] results = [SELECT Maintenance_Request__c,
1010MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM
1011Equipment_Maintenance_Item__c WHERE
1012Maintenance_Request__c IN :ValidIds GROUP BY
   Maintenance_Request__c];
1013for (AggregateResult ar : results){
1014maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'),
1015(Decimal) ar.get('cycle'));
1016}
1017for(Case cc : closedCasesM.values()){
1018Case nc = new Case (
1019ParentId = cc.Id,
1020Status = 'New',
1021Subject = 'Routine Maintenance',
1022Type = 'Routine Maintenance',
1023Vehicle__c = cc.Vehicle__c,
1024Equipment__c =cc.Equipment__c,
1025Origin = 'Web',
1026Date_Reported__c = Date.Today()
1027);
1028If (maintenanceCycles.containskey(cc.Id)){
1029nc.Date_Due__c = Date.today().addDays((Integer)
1030maintenanceCycles.get(cc.Id));
1031SPSGP-90546-Salesforce Developer Catalyst
```

```
1032Self-Learning & Super Badges
103327
1034APEX SPECIALIST SUPER BADGE CODES
1035}
1036newCases.add(nc);
1037}
1038insert newCases;
1039List<Equipment_Maintenance_Item__c> clonedWPs = new
1040List<Equipment_Maintenance_Item__c>();
1041for (Case nc : newCases){
1042for (Equipment_Maintenance_Item__c wp :
1043closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
1044Equipment_Maintenance_Item__c wpClone = wp.clone();
1045wpClone.Maintenance_Request__c = nc.Id;
1046ClonedWPs.add(wpClone);
1047}
1048}
1049insert ClonedWPs;
1050}
1051}
1052}
1053Challenge-5
1054WarehouseCalloutService.apxc:
1055public with sharing class WarehouseCalloutService implements
1056Queueable {
1057private static final String WAREHOUSE_URL = 'https://th-
1058//class that makes a REST callout to an external warehouse
   system to
1059get a list of equipment that
1060needs to be updated.
1061//The callout's JSON response returns the equipment records that
   you
1062upsert in Salesforce.
1063@future(callout=true)
1064public static void runWarehouseEquipmentSync(){
1065Http http = new Http();
1066HttpRequest request = new HttpRequest();
1067request.setEndpoint(WAREHOUSE_URL);
1068SPSGP-90546-Salesforce Developer Catalyst
1069Self-Learning & Super Badges
```

```
107028
1071APEX SPECIALIST SUPER BADGE CODES
1072request.setMethod('GET');
1073HttpResponse response = http.send(request);
1074List<Product2> warehouseEq = new List<Product2>();
1075if (response.getStatusCode() == 200){
1076List<Object> jsonResponse =
1077(List<Object>)JSON.deserializeUntyped(response.getBody());
1078System.debug(response.getBody());
1079//class maps the following fields: replacement part (always
   true),
1080cost, current inventory,
1081lifespan, maintenance cycle, and warehouse SKU
1082//warehouse SKU will be external ID for identifying which
   equipment
1083records to update within
1084Salesforce
1085for (Object eq : jsonResponse){
1086Map<String,Object> mapJson = (Map<String,Object>)eq;
1087Product2 myEq = new Product2();
1088myEq.Replacement_Part__c = (Boolean) mapJson.get('replacement');
1089myEq.Name = (String) mapJson.get('name');
1090myEq.Maintenance_Cycle__c = (Integer)
1091mapJson.get('maintenanceperiod');
1092myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
1093myEq.Cost__c = (Integer) mapJson.get('cost');
1094myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
1095myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
1096myEq.ProductCode = (String) mapJson.get('_id');
1097warehouseEq.add(myEq);
1098}
1099if (warehouseEq.size() > 0){
1100upsert warehouseEq;
1101System.debug('Your equipment was synced with the warehouse

1102}
1103}
1104}
1105public static void execute (QueueableContext context){
1106runWarehouseEquipmentSync();
```

```
1107}
1108SPSGP-90546-Salesforce Developer Catalyst
1109Self-Learning & Super Badges
111029
1111APEX SPECIALIST SUPER BADGE CODES
1112}
1113WarehouseCalloutServiceMock.apxc:
1114@isTest
1115global class WarehouseCalloutServiceMock implements
   HttpCalloutMock {
1116// implement http mock callout
1117global static HttpResponse respond(HttpRequest request) {
1118HttpResponse response = new HttpResponse();
1119response.setHeader('Content-Type', 'application/json');
1120response.setBody('[{"_id":"55d66226726b611100aaf741","replacemen
   t":fa
1121lse,"quantity":5,"name":"Gene
1122rator 1000
1123kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10

1124af742","replacement":true,"quantity":183,"name":"Cooling
1125Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004
   "},{"
1126_id":"55d66226726b611100aaf743
1127","replacement":true,"quantity":143,"name":"Fuse
112820A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"
   }]');
1129response.setStatusCode(200);
1130return response;
1131}
1132}
1133WarehouseCalloutServiceTest.apxc:
1134@isTest
1135global class WarehouseCalloutServiceMock implements
   HttpCalloutMock {
1136// implement http mock callout
1137global static HttpResponse respond(HttpRequest request) {
1138HttpResponse response = new HttpResponse();
1139response.setHeader('Content-Type', 'application/json');
1140response.setBody('[{"_id":"55d66226726b611100aaf741","replacemen
```

```
   t":fa
1141lse,"quantity":5,"name":"Gene
1142rator 1000
1143kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10

1144af742","replacement":true,"quantity":183,"name":"Cooling
1145Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004
   "},{"
1146_id":"55d66226726b611100aaf743
1147","replacement":true,"quantity":143,"name":"Fuse
114820A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"
   }]');
1149SPSGP-90546-Salesforce Developer Catalyst
1150Self-Learning & Super Badges
115130
1152APEX SPECIALIST SUPER BADGE CODES
1153response.setStatusCode(200);
1154return response;
1155}
1156}
1157Challenge-6
1158WarehouseSyncSchedule.apxc:
1159global with sharing class WarehouseSyncSchedule implements
1160Schedulable{
1161global void execute(SchedulableContext ctx){
1162System.enqueueJob(new WarehouseCalloutService());
1163}
1164}
1165WarehouseSyncScheduleTest.apxc:
1166@isTest
1167public class WarehouseSyncScheduleTest {
1168@isTest static void WarehousescheduleTest(){
1169String scheduleTime = '00 00 01 * * ?';
1170Test.startTest();
1171Test.setMock(HttpCalloutMock.class, new
1172WarehouseCalloutServiceMock());
1173String jobID=System.schedule('Warehouse Time To Schedule to

1174scheduleTime, new
1175WarehouseSyncSchedule());
```

```
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 ');
}
}
```