Apex Triggers : https://trailhead.salesforce.com/content/learn/modules/apex\_triggers?trailmix\_creator\_id=trailblazerconnect&trailmix\_slug=salesforce-developer-catalyst

1. Get Started With Apex Triggers

AccountAddressTrigger Code :

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

for (Account a : Trigger.new) {

if (a.Match\_Billing\_Address\_\_c == TRUE){

a.ShippingPostalCode = a.BillingPostalCode;

}

}

}

2) Bulk Apex Triggers Unit

ClosedOpportunityTrigger Code :

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

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

for (Opportunity o :[SELECT Id,Name FROM Opportunity

WHERE Id IN :Trigger.New]){

taskList.add(new Task(Subject='Follow Up Test Task',

WhatId=o.Id,

Status='Not Started',

Priority='Normal'));

}

if (taskList.size() > 0){

insert taskList;

}

}

Apex Testing : https://trailhead.salesforce.com/content/learn/modules/apex\_testing?trailmix\_creator\_id=trailblazerconnect&trailmix\_slug=salesforce-developer-catalyst

1)Get Started with Apex Unit Testing

VerifyDate Code :

public class VerifyDate {

//method to handle potential checks against two dates

public static Date CheckDates(Date date1, Date date2) {

//if date2 is within the next 30 days of date1, use date2. Otherwise use the end of the month

if(DateWithin30Days(date1,date2)) {

return date2;

} else {

return SetEndOfMonthDate(date1);

}

}

//method to check if date2 is within the next 30 days of date1

private static Boolean DateWithin30Days(Date date1, Date date2) {

//check for date2 being in the past

if( date2 < date1) { return false; }

//check that date2 is within (>=) 30 days of date1

Date date30Days = date1.addDays(30); //create a date 30 days away from date1

if( date2 >= date30Days ) { return false; }

else { return true; }

}

//method to return the end of the month of a given date

private static Date SetEndOfMonthDate(Date date1) {

Integer totalDays = Date.daysInMonth(date1.year(), date1.month());

Date lastDay = Date.newInstance(date1.year(), date1.month(), totalDays);

return lastDay;

}

}

TestVerifyDate Code :

@isTest

private class TestVerifyDate {

@isTest static void testCheckDates() {

Date now = Date.today();

Date lastOfTheMonth = Date.newInstance(now.year(), now.month(), Date.daysInMonth(now.year(), now.month()));

Date plus60 = Date.today().addDays(60);

Date d1 = VerifyDate.CheckDates(now, now);

System.assertEquals(now, d1);

Date d2 = VerifyDate.CheckDates(now, plus60);

System.assertEquals(lastOfTheMonth, d2);

}

}

2) Test Apex Triggers Unit

RestrictContactByName Code :

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 Code :

@isTest

private class TestRestrictContactByName {

@isTest

static void invalidName() {

try {

Contact c = new Contact(LastName='INVALIDNAME');

insert c;

}

catch (Exception e) {

System.assert(true);

}

}

}

3) Create Test Data for Apex Tests :

RandomContactFactory Code :

public class RandomContactFactory {

public static List<Contact> generateRandomContacts(Integer num, String lastName) {

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

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

Contact c = new Contact(FirstName=i.format(), LastName=lastName);

contacts.add(c);

}

return contacts;

}

}

Asynchronous Apex : https://trailhead.salesforce.com/content/learn/modules/asynchronous\_apex?trailmix\_creator\_id=trailblazerconnect&trailmix\_slug=salesforce-developer-catalyst

1)Quiz

2)Use Future Methods

AccountProcessor Code :

public class AccountProcessor {

@future

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

List<Account> accounts = [SELECT Id,

Name,

Number\_of\_Contacts\_\_c,

(

SELECT Contact.Id

FROM Contacts

)

FROM Account

WHERE Id in :accountIds];

for (Account a : accounts) {

a.Number\_of\_Contacts\_\_c = a.Contacts.size();

}

update accounts;

}

}

AccountProcessorTest Code :

@isTest

private class AccountProcessorTest {

static TestMethod void myTest() {

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

for (Integer i=0; i<100; i++) {

Account account = new Account();

account.Name = 'AccountProcessorTest Account ' + i;

accounts.add(account);

}

insert accounts;

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

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

for (Account a : accounts) {

accountIds.add(a.Id);

for (Integer i=0; i<5; i++) {

Contact contact = new Contact();

contact.FirstName = 'AccountProcessor Test Contact';

contact.LastName = String.valueOf(i);

contact.AccountId = a.Id;

contacts.add(contact);

}

}

insert contacts;

Test.startTest();

AccountProcessor.countContacts(accountIds);

Test.stopTest();

List<Account> results = [SELECT Id, Number\_of\_Contacts\_\_c

FROM Account

WHERE Id in :accountIds];

for (Account a : results) {

System.AssertEquals(5, a.Number\_of\_Contacts\_\_c);

}

}

}

3)Use Batch Apex

LeadProcessor Code :

global class LeadProcessor implements Database.Batchable<sObject>, Database.Stateful {

global Integer recs\_processed = 0;

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

String sQuery = '';

sQuery += 'SELECT Id, Name, Status,';

sQuery += 'LeadSource ';

sQuery += 'FROM Lead ';

sQuery += 'LIMIT 100000';

return Database.getQueryLocator(sQuery);

}

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

for (Lead l : scope) {

l.LeadSource = 'Dreamforce';

recs\_processed += 1;

}

update scope;

}

global void finish(Database.BatchableContext bc) {

AsyncApexJob job = [SELECT Id,

Status,

NumberOfErrors,

TotalJobItems,

JobItemsProcessed,

CreatedBy.Email

FROM AsyncApexJob

WHERE Id = :bc.getJobId()];

String s = '';

s += job.JobItemsProcessed + ' job items processed ';

s += 'out of ' + job.TotalJobItems + ' total job items. ';

s += job.NumberOfErrors + ' error(s) encountered. ';

System.debug(s);

s = recs\_processed + ' record(s) processed.';

System.debug(s);

}

}

LeadProcessorTest Code :

@isTest

private class LeadProcessorTest {

@testSetup

static void createLeads() {

List<Lead> leads = new List<Lead>();

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

Lead l = new Lead();

l.FirstName = 'Test';

l.LastName = 'Lead';

l.Company = 'Test Lead ' + i;

leads.add(l);

}

insert leads;

}

static TestMethod void myTest() {

Test.startTest();

LeadProcessor lp = new LeadProcessor();

Id batchId = Database.executeBatch(lp);

Test.stopTest();

System.assertEquals(200, [SELECT Count()

FROM Lead

WHERE Name = 'Test Lead'

AND LeadSource = 'Dreamforce']);

}

}

4)Controp Processes with Queueable Apex

AddPrimaryContact Code :

public class AddPrimaryContact implements Queueable {

private Contact contactObj;

private String state\_code;

public AddPrimaryContact(Contact c, String s) {

this.contactObj = c;

this.state\_code = s;

}

public void execute(QueueableContext context) {

List<Account> accounts = [SELECT Id

FROM Account

WHERE BillingState = :this.state\_code

LIMIT 200];

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

for (Account a : accounts) {

Contact c = this.contactObj.clone(false, false, false, false);

c.AccountId = a.Id;

contacts.add(c);

}

if (contacts.size() > 0) {

insert contacts;

}

}

}

AddPrimaryContactTest Code :

@isTest

private class AddPrimaryContactTest {

@testSetup

static void setup() {

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

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

Account ny = new Account();

ny.Name = 'Test Account (NY)';

ny.BillingState = 'NY';

accounts.add(ny);

Account ca = new Account();

ca.Name = 'Test Account (CA)';

ca.BillingState = 'CA';

accounts.add(ca);

}

insert accounts;

}

static TestMethod void myTest() {

Contact contactObj = new Contact(

FirstName = 'California',

LastName = 'Bob'

);

String state\_abbrev = 'CA';

Test.startTest();

AddPrimaryContact apc = new AddPrimaryContact(contactObj, state\_abbrev);

Id jobId = System.enqueueJob(apc);

Test.stopTest();

List<Account> accounts = [SELECT Id, (SELECT Contact.Name FROM Account.Contacts) FROM Account WHERE BillingState = 'CA'];

System.assertEquals(50, accounts.size());

for (Account a : accounts) {

System.assertEquals(a.Contacts.size(), 1);

}

}

}

5)Schedule Jobs Using the Apex Scheduler

DailyLeadProcessor Code :

global class DailyLeadProcessor implements Schedulable {

global void execute(SchedulableContext ctx) {

List<Lead> leads = [SELECT Id,

LeadSource

FROM Lead

WHERE LeadSource = '' OR LeadSource = null

LIMIT 200];

for (Lead l : leads) {

l.LeadSource = 'Dreamforce';

}

if (leads.size() > 0) {

update leads;

}

}

}

DailyLeadProcessorTest Code :

@isTest

private class DailyLeadProcessorTest {

@testSetup

static void setup() {

List<Lead> leads = new List<Lead>();

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

Lead l = new Lead();

l.FirstName = 'Test';

l.LastName = 'Lead ' + i;

l.Company = 'Test Company ' + i;

leads.add(l);

}

insert leads;

}

static TestMethod void myTest() {

String jobName = 'Daily Lead Processor - Test';

String CRON\_EXP = '0 0 0 15 3 ? 2017'; // dummy cron entry

test.startTest();

DailyLeadProcessor dp = new DailyLeadProcessor();

String JobId = System.schedule(jobName, CRON\_EXP, dp);

test.stopTest();

List<Lead> results = [SELECT Id FROM Lead WHERE LeadSource = 'Dreamforce'];

System.assertEquals(200, results.size());

}

}

Apex Integration Services :https://trailhead.salesforce.com/content/learn/modules/apex\_integration\_services?trailmix\_creator\_id=trailblazerconnect&trailmix\_slug=salesforce-developer-catalyst

1)Quiz

2)Apex REST Callouts

AnimalLocator Code :

public class AnimalLocator {

public static HttpResponse makeGetCallout {

Http http = new Http();

HttpRequest request = new HttpRequest();

request.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/:id');

request.setMethod('GET');

HttpResponse response = http.send(request);

// If the request is successful, parse the JSON response.

if (response.getStatusCode() == 200) {

// Deserialize the JSON string into collections of primitive data types.

Map<Integer, Object> Results

}

}

}

AnimalLocatorTest Code :

@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 Code :

@isTest  
global class AnimalLocatorMock implements HttpCalloutMock {  
 global HTTPResponse respond(HTTPRequest request) {  
 HttpResponse response = new HttpResponse();  
 response.setHeader('Content-Type', 'application/json');  
 response.setBody('{"animal":{"id":1,"name":"chicken","eats":"chicken food","says":"cluck cluck"}}');  
 response.setStatusCode(200);  
 return response;  
 }  
}

2)Apex SOAP Callouts

ParkService Code :

//Generated by wsdl2apex

public class ParkService {

public class byCountryResponse {

public String[] return\_x;

private String[] return\_x\_type\_info = new String[]{'return','http://parks.services/',null,'0','-1','false'};

private String[] apex\_schema\_type\_info = new String[]{'http://parks.services/','false','false'};

private String[] field\_order\_type\_info = new String[]{'return\_x'};

}

public class byCountry {

public String arg0;

private String[] arg0\_type\_info = new String[]{'arg0','http://parks.services/',null,'0','1','false'};

private String[] apex\_schema\_type\_info = new String[]{'http://parks.services/','false','false'};

private String[] field\_order\_type\_info = new String[]{'arg0'};

}

public class ParksImplPort {

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

public Map<String,String> inputHttpHeaders\_x;

public Map<String,String> outputHttpHeaders\_x;

public String clientCertName\_x;

public String clientCert\_x;

public String clientCertPasswd\_x;

public Integer timeout\_x;

private String[] ns\_map\_type\_info = new String[]{'http://parks.services/', 'ParkService'};

public String[] byCountry(String arg0) {

ParkService.byCountry request\_x = new ParkService.byCountry();

request\_x.arg0 = arg0;

ParkService.byCountryResponse response\_x;

Map<String, ParkService.byCountryResponse> response\_map\_x = new Map<String, ParkService.byCountryResponse>();

response\_map\_x.put('response\_x', response\_x);

WebServiceCallout.invoke(

this,

request\_x,

response\_map\_x,

new String[]{endpoint\_x,

'',

'http://parks.services/',

'byCountry',

'http://parks.services/',

'byCountryResponse',

'ParkService.byCountryResponse'}

);

response\_x = response\_map\_x.get('response\_x');

return response\_x.return\_x;

}

}

}

ParkLocator Code :

public class ParkLocator {

public static String[] country(String country){

ParkService.ParksImplPort parks = new ParkService.ParksImplPort();

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

return parksname;

}

}

ParkLocatorTest Code :

@isTest

private class ParkLocatorTest{

@isTest

static void testParkLocator() {

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

String[] arrayOfParks = ParkLocator.country('India');

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

}

}

ParkServiceMock Code :

@isTest

global class ParkServiceMock implements WebServiceMock {

global void doInvoke(

Object stub,

Object request,

Map<String, Object> response,

String endpoint,

String soapAction,

String requestName,

String responseNS,

String responseName,

String responseType) {

ParkService.byCountryResponse response\_x = new ParkService.byCountryResponse();

List<String> lstOfDummyParks = new List<String> {'Park1','Park2','Park3'};

response\_x.return\_x = lstOfDummyParks;

response.put('response\_x', response\_x);

}

}

4) Apex Web Services

AccountManager Code :

@RestResource(urlMapping='/Accounts/\*/contacts')  
global with sharing class AccountManager {  
   
   
 @HttpGet  
 global static account getAccount() {  
   
 RestRequest request = RestContext.request;  
   
 String accountId = request.requestURI.substring(request.requestURI.lastIndexOf('/')-18,  
 request.requestURI.lastIndexOf('/'));  
 List<Account> a = [select id, name, (select id, name from contacts) from account where id = :accountId];  
 List<contact> co = [select id, name from contact where account.id = :accountId];  
 system.debug('\*\* a[0]= '+ a[0]);  
 return a[0];  
  
 }  
  
}

AccountManagerTest Code :

@Istest(SeeAllData=true)  
public class AccountManagerTest {  
   
  
   
 @IsTest  
 public static void testaccountmanager() {  
 RestRequest request = new RestRequest();  
 request.requestUri = 'https://mannharleen-dev-ed.my.salesforce.com/services/apexrest/Accounts/00190000016cw4tAAA/contacts';  
 request.httpMethod = 'GET';  
 RestContext.request = request;  
   
 system.debug('test account result = '+ AccountManager.getAccount());  
  
  
   
 }  
   
  
}

APEX SPECIALIST SUPERBADGE :

https://trailhead.salesforce.com/content/learn/modules/apex\_integration\_services?trailmix\_creator\_id=trailblazerconnect&trailmix\_slug=salesforce-developer-catalyst

1)Quiz

2)Automate Record Creation

MaintenanceRequestHelper Code :

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));

}

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;

}

}

### }

### MaitenanceRequest Code :

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

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

MaintenanceRequestHelper.updateWorkOrders(Trigger.New, Trigger.OldMap);

}

}

3)Synchronize Salesforce Data

WarehouseCalloutService Code :

public with sharing class WarehouseCalloutService {

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

//@future(callout=true)

public static void runWarehouseEquipmentSync(){

Http http = new Http();

HttpRequest request = new HttpRequest();

request.setEndpoint(WAREHOUSE\_URL);

request.setMethod('GET');

HttpResponse response = http.send(request);

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

if (response.getStatusCode() == 200){

List<Object> jsonResponse = (List<Object>)JSON.deserializeUntyped(response.getBody());

System.debug(response.getBody());

for (Object eq : jsonResponse){

Map<String,Object> mapJson = (Map<String,Object>)eq;

Product2 myEq = new Product2();

myEq.Replacement\_Part\_\_c = (Boolean) mapJson.get('replacement');

myEq.Name = (String) mapJson.get('name');

myEq.Maintenance\_Cycle\_\_c = (Integer) mapJson.get('maintenanceperiod');

myEq.Lifespan\_Months\_\_c = (Integer) mapJson.get('lifespan');

myEq.Cost\_\_c = (Decimal) mapJson.get('lifespan');

myEq.Warehouse\_SKU\_\_c = (String) mapJson.get('sku');

myEq.Current\_Inventory\_\_c = (Double) mapJson.get('quantity');

warehouseEq.add(myEq);

}

if (warehouseEq.size() > 0){

upsert warehouseEq;

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

System.debug(warehouseEq);

}

}

}

}

4)Schedule Synchronization

WarehouseSyncSchedule Code :

global class WarehouseSyncSchedule implements Schedulable {

global void execute(SchedulableContext ctx) {

WarehouseCalloutService.runWarehouseEquipmentSync();

}

}

5)Test Automatic Logic

MaintenanceRequestHelperTest Code :

@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;

}

MaintenanceRequestHelper Code :

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));

}

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 Code :

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

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

MaintenanceRequestHelper.updateWorkOrders(Trigger.New, Trigger.OldMap);

}

}

6) Test Callout Logic

WarehouseCalloutService Code :

public with sharing class WarehouseCalloutService {

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

//@future(callout=true)

public static void runWarehouseEquipmentSync(){

Http http = new Http();

HttpRequest request = new HttpRequest();

request.setEndpoint(WAREHOUSE\_URL);

request.setMethod('GET');

HttpResponse response = http.send(request);

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

if (response.getStatusCode() == 200){

List<Object> jsonResponse = (List<Object>)JSON.deserializeUntyped(response.getBody());

System.debug(response.getBody());

for (Object eq : jsonResponse){

Map<String,Object> mapJson = (Map<String,Object>)eq;

Product2 myEq = new Product2();

myEq.Replacement\_Part\_\_c = (Boolean) mapJson.get('replacement');

myEq.Name = (String) mapJson.get('name');

myEq.Maintenance\_Cycle\_\_c = (Integer) mapJson.get('maintenanceperiod');

myEq.Lifespan\_Months\_\_c = (Integer) mapJson.get('lifespan');

myEq.Cost\_\_c = (Decimal) mapJson.get('lifespan');

myEq.Warehouse\_SKU\_\_c = (String) mapJson.get('sku');

myEq.Current\_Inventory\_\_c = (Double) mapJson.get('quantity');

warehouseEq.add(myEq);

}

if (warehouseEq.size() > 0){

upsert warehouseEq;

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

System.debug(warehouseEq);

}

}

}

}

WarehouseCalloutServiceTest Code :

@isTest

private class WarehouseCalloutServiceTest {

@isTest

static void testWareHouseCallout(){

Test.startTest();

// implement mock callout test here

Test.setMock(HTTPCalloutMock.class, new WarehouseCalloutServiceMock());

WarehouseCalloutService.runWarehouseEquipmentSync();

Test.stopTest();

System.assertEquals(1, [SELECT count() FROM Product2]);

}

}

WarehouseCalloutServiceMock Code :

@isTest

global class WarehouseCalloutServiceMock implements HttpCalloutMock {

// implement http mock callout

global static HttpResponse respond(HttpRequest request){

System.assertEquals('https://th-superbadge-apex.herokuapp.com/equipment', request.getEndpoint());

System.assertEquals('GET', request.getMethod());

// Create a fake response

HttpResponse response = new HttpResponse();

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

response.setBody('[{"\_id":"55d66226726b611100aaf741","replacement":false,"quantity":5,"name":"Generator 1000 kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"}]');

response.setStatusCode(200);

return response;

}

}

7) Test Scheduling Logic

WarehouseSyncSchedule Code :

global class WarehouseSyncSchedule implements Schedulable {

global void execute(SchedulableContext ctx) {

WarehouseCalloutService.runWarehouseEquipmentSync();

}

}

WarehouseSyncScheduleTest Code :

@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 Test', 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 ');

}

}

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |