APEX SPECIALIST SUPER BADGE CODES

APEX TRIGGERS

AccountAddressTrigger.axpt:

```
trigger
AccountAddress
Triggeron
Account (before
insert,before
update) {
for(Account
account:Trigger.
New){
  if(account.Ma
     tch_Billing_
     Address c
     == True){
     account.Shi
     ppingPostal
     Code =
     account.Bill
     ingPostalCo
     de;
    }
  }
}
```

ClosedOpportunityTrigger.axpt:

```
 trigger
ClosedOpportunityTrigg
er on Opportunity
(afterinsert,afterupdate)
{ List<Task> tasklist= new
List<Task>();
for
   (
   O
   p
   p
   o
   r
   t
   u
   n
   i
   t
   y

   o
   p
   p
   :

   T
```

```
r
i
g
g
e
r
.
N
e
w
)
{

i
f
(
o
p
p
.
S
t
a
g
e
N
a
m
e

=
=

'
C
l
o
```

```
s
e
d
W
o
n
'
)
{

    tasklist.add(newTask(Subject =
    'Follow Up Test Task',WhatId =opp.Id));
}
}
if(t
a
s
k
l
i
s
t
.
s
i
z

e
(
)

>

0
)
{

ins
```

```
        ert
        tas
        kli
        st;

      }

    }
```

```
    public class VerifyDate {
```

APEX TESTING

VerifyData.apxc:

```
            public static Date
                CheckDates(Date date1,
                Date date2) {
                if(DateWithin30Days(date1,
                date2)) {
                    return date2;


                } else {


}
                   }
```

```
return SetEndOfMonthDate(date1);

      @TestVisible privatestatic
      Boolean
      DateWithin30Days(Datedate1,
      Date date2){
          /c
      h
      e
      c
      k

      f
      o
      r

      d
      a
      t
      e
      2

      b
      e
      i
      n
      g

      i
      n
      t
      h
      e

      p
      a
      s
      t
```

i
f
(

d
a
t
e
2
<

d
a
t
e
1
)

{

r
e
t
u
r
n
f
a
l
s
e
;

}

APEXSPECIALIST SUPER BADGE CODES

/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 returnthe end of the monthof a given date

```
@TestVis
ible
private
staticDa
te
```

```
        SetEndOf
        MonthDa
        te(Date
        date1){
                IntegertotalDays
                =Date.daysInMonth(date1.y
                ear(), date1.month());
                Date
                lastDay =
                Date.newIn
                stance(dat
                e1.year(),
                date1.mont
                h(),
                totalDays);
                return
                lastDay;
        }

}
```

TestVerifyData.apxc:

```
@isTest
private class TestVerifyDate {
  @isTest static void
  Test_CheckDates_case1(){
    Date D =
    VerifyDate.CheckDates(date.parse('01/01
    /2022'),date.parse('01/05/2022'));
    System.assertEquals(date.parse('01/05/2
    022'), D);
}
  @isTest static void
  Test_CheckDates_case2(){
    Date D =
    VerifyDate.CheckDates(date.parse('01/01
    /2022'), date.parse('05/05/2022'));
    System.assertEquals(date.parse('01/31/2
    022'), D);
}
@i
  s
  T
  e
  s
  t

  s
  t
  a
  t
  i
  c

  v
  o
  i
  d

  T
  e
  s
  t
  _
  W
```

```
i
t
h
i
n
3
0
D
a
y
s
_
c
a
s
e
1
(
)
{
B
o
o
l
e
a
n
```

```
        fl
        a
        g

        =

VerifyDate.DateWithi
n30Days(date.parse(
'01/01/2022'),
date.parse('12/30/20
21'));
        System.assertEquals(false, flag);
    }
@is
        T
        e
        s
        t

        s
        t
        a
        t
        i
        c

        v
        o
        i
        d
```

```
T
e
s
t
_
W
i
t
h
i
n
3
0
D
a
y
s
_
c
a
s
e
2
(
)
{
B
```

```
o
o
l
e
a
n

fl
a
g

=

VerifyDate.DateWithi
n30Days(date.parse(
'01/01/2022'),
date.parse('02/02/20
21'));
    System.assertEquals(false, flag);
 }
@isTest static void
Test_Within30Days_case3(){
```

(beforeinsert, before update){

/check contacts prior to insert or updat

Boolean flag =

APEXSPECIALIST SUPER BADGE CODES

```
VerifyDate.DateWithi
n30Days(date.parse(
'01/01/2022'),
date.parse('01/15/20
22'));
    System.assertEquals(true, flag);
  }
  @isTest static void
  Test_SetEndOfMonthDate(){
    Datereturndate
    =VerifyDate.SetEndOfMonthDate(date.par
    se('01/01/2022'));
}
}
```

RestrictContactByName.apxt:

```
trigger RestrictContactByName on Contact
```

e

f
o
r
i
n
v
a
l
i
d

d
a
t
a

F
o
r

(
C
o
n
t
a
c
t
c

:

T
r
i
g
g
e
r

```
.
N
e
w
)

{

        if(c.LastName ==
            'INVALIDNAME') {
            /invalidname is
            invalid
            c.AddError('The
            Last Name
            "'+c.LastName+'"
            is not allowedfor
            DML');
        }


    }
}
```

TestRestrictContactByName.apxc:

```
@isTest
private class TestRestrictContactByName
  {
  @i
  sT
  es
  tst
  at
  ic
  vo
```

```
id
Te
st_
ins
ert
up
da
te
Co
nt
act
(){
   Co
   nt
   act
   c
   nt
   = n
   ew
   Co
   nt
   ac
   t();
   cn
   t.L
   ast
   Na
   m
   e =
   'IN
   VA
   LI
   DN
   AM
   E';
```

```
Te
st.
st
art
Te
st(
);
Da
ta
ba
se.
Sa
ve
Re
su
lt
re
sul
t=
Da
ta
ba
se.
ins
ert
(c
nt,
fal
se)
;Te
st.
st
op
Te
st(
);
Sy
```

```
ste
m.
as
se
rt(!
re
sul
t.is
Su
cc
es
s()
);
Sy
ste
m.
as
se
rt(
re
sul
t.g
et
Err
or
s().
siz
e()
>
0);
System.assertEquals('The Last
Name"INVALIDNAME" is notallowed for  DML',
result.getErrors()[0].getMessage());
   }

}
```

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(First
            Name = '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<Accoun
```

```
        t>();

        List<Account> accounts = [Select Id,
        Name, (Select Id from Contacts)from
        Account Where Id in
    :accountIds];
        For(Account acc: accounts) {
                                   Lis
          t<Contact>
          contactList =
          acc.contacts;
          acc.Number_
          Of_Contacts
          c =
          contactList.s
          ize();
          accountsToU
          pdate.add(a
          cc);
        }
        update accountsToUpdate;
      }
    }
```

**AccountP
rocessorT
est.apxc:**

```
@isTest
public class AccountProcessorTest {
            @isTest
    private static void testCountContacts() {
        Ac
        co
        u
nt
ne
wA
cc
ou
nt
=
n
ew
Ac
co
un
t(N
a
m
e =
'Te
st
Ac
co
un
t');
ins
ert
ne
wA
cc
ou
nt;
ContactnewContact1 = new
Contact(FirstName ='John',LastName =
'Doe',AccountId =
```

newAccount.Id);

APEXSPECIALIST SUPER BADGE CODES

```
        insert newContact1;

        Contact newContact2 =new
Contact(FirstName ='John',LastName =
'Doe',AccountId = newAccount.Id);
        insert newContact2;
        Lis
        t<I
        d>
        ac
        co
        un
        tI
        ds
        = n
        ew
        Lis
        t<I
        d>
        ();
        ac
        co
        un
        tId
        s.a
        dd
        (n
        ew
```

```
Ac
co
un
t.I
d);
Te
st.
st
art
Te
st(
);
Ac
co
un
tPr
oc
es
so
r.c
ou
nt
Co
nt
act
s(a
c
co
un
tId
s);
Te
st.
st
op
Te
st(
);
    }
}
```

Lead Processor.apxc:

```
global class
          Lead
          Proc
          ess
          or
          impl
          eme
          nts
          Data
          base
          .Batc
          habl
          e<sO
          bject
          >{
          glob
          alInt
          eger
          cou
          nt =
          0;

  global
  Database.QueryL
  ocator
  start(Database.Ba
  tchableContext
  bc) { return
  Database.getQuer
  yLocator('SELECT
  ID,LeadSource
  FROM Lead');
  }
  global void
execute(Datab
ase.BatchableC
ontext bc,
List<Lead>
L_list){
List<lead>
L_list_new =
new
List<lead>();
for
  (
  l
  e
  a
  d

  L
  :

  L
  _
  l
  i
  s
  t
  )
  {

  L
  .
  l
  e
  a
  d
  S
  o
  u
  r
  c
  e
```

```
= 'Dreamforce';
L_list_new.add(L);count +
```

```
        = 1;
    }
    update L_list_new;
}
global void finish(Database.Batch
ableContext bc) {

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

LeadProcessorTest.apxc:

```
@isTest
pu
    bl
```

```
ic class LeadProcessorTest {@isTe
```

s

t

pu

b
l
i
c
s
t
a
t
i
c

v
o
i
d
t
e
s
t
i
t
(
)
{

APEX SPECIALIST SUPER BADGE

CODES

```
Lis
t<l
ea
d>
L_l
ist
=
n
ew
Lis
t<l
ea
d>
();f
or(
Int
eg
er
i =
0; i
<
20
0;
i+
+) {
    Le
    ad
    L =
    n
    ew
    Le
    ad
    ();
    L.L
    ast
    Na
    m
    e =
    'na
    m
    e'
    + i;
    L.C
    o
    m
    pa
    ny
    =
    'Co
    m
    pa
    ny
    ';
    L.S
    tat
    us
    =
    'Ra
    nd
    o
    m
    St
    at
    us
    ';
    L_l
    ist.
    ad
    d(
    L);
}
ins
ert
L_l
ist;
Te
st.
sta
```

```
rt
Te
st(
);
Le
ad
Pr
oc
es
s
or
lp
= n
ew
Le
ad
Pr
oc
es
so
r();
Id
ba
tc
h
Id
=
Da
ta
ba
se.
ex
ec
ut
eB
at
ch
(lp
);
Te
st.
```

```
        st
        op
        Te
        st(
        );
    }
}
```

AddPrimaryContact.apxc:

```
pu
b
l
i
c

c
l
a
s
s

A
d
d
P
r
i
m
a
r
y
C
o
n
t
a
c
t

i
m
p
l
e
m
e
n
t
s
Q
u
e
u
e
a
b
l
e
{

p
r
i
v
a
t
e

C
```

```
ontact con;
private String state;
public AddPrimaryContact(Con
tact con, String state) {
this.con = con;
```

```
    this.state = state;
}
public void execute(QueueableContext
context){
    List<Account> accounts = [Select
                         Id,Name,(Select
                         FirstName,LastNam
                         e, Id from contacts)
                         from
                         Accountwhere
                         BillingState =
                         :state Limit 200];
    List<
    Cont
    act>
    prim
    aryC
    onta
    cts =
    new
    List<
    Cont
    act>(
    );
    for(A
    ccou
    nt
    acc :
    acco
    unt
    s) {
        Co
        nt
        a
        ct
        c =
        co
        n.c
        lo
        ne
        ();
        c.A
        cc
        ou
        nt
        Id
        =
        ac
        c.I
        d;
        pri
        m
        ar
        yC
        on
        tac
        ts.
        a
        dd
        (c
        );
    }
    if(
        p
        r
        i
        m
        a
        r
        y
        C
        o
        n
        t
        a
        c
        t
        s
        .
s
i
z
e
(
)

>

0
)

{

i
n
s
e
r
t
p
r
i
m
a
r
y
C
o
n
t
a
c
t
s
;

}

}

}
```

@isTest public class

APEX SPECIALIST SUPER BADGE CODES

AddPrimaryContactTest.apxc:

```
AddPrimaryContactTest { static

tes
tm
et
h
od
vo
```

```
id
tes
tQ
ue
ue
ab
le
() {
   Lis
   t<
   Ac
   co
   un
   t>
   tes
   tA
   cc
   ou
   n
   ts
   =
   ne
   wL
   ist
   <A
   cc
   ou
   nt
   >(
   );
   for
   (In
   te
   g
   er
   i =
   0; i
   <
   50;
   i+
   +) {
```

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

   }

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

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

   }

   insert testAccounts;

   Contact
   testContact
   =new
   Contact(FirstNa
   me ='John',
   LastName =
   'Doe'); insert
   testContact;
   AddPrimaryCon
   tact addit =
   new
   AddPrimaryCon
   tact(testContac
   t,'CA');
   Test.startTest();
   system.enqueu
   eJob(ad
   dit); Test.stopTest();
   System.assertEquals(50, [Select
count()from Contact where accountId
in (Select Idfrom Account where
BillingState = 'CA')]);
   }

}
```

DailyLeadProcessor.apxc:

```
gl
   o
   b
```

```
al class DailyLeadProcessor implements Sch
edulable{ global void execute(Schedulable
C
o
n
t
e
x
t

c
t
x
)

{

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

   List<Lead> leads = [Select
   id From LeadWhere
   LeadSource = NULL
   Limit200]; for(Lead l:
   leads) {
      l.L
      ea
      dS
      ou
      rce
      =
      'Dr
      ea
      mf
      orc
      e';
      le
      ad
      st
      ou
      pd
      at
      e.a
         dd
         (l);
      }

      update leadstoupdate;

   }

}
```

APEX SPECIALIST SUPER BADGE CODES

DailyLeadProcessorTest.apxc:

```
@i
s T
e s
t

private class DailyLeadProcessorTest {

          public static String CRON_EXP= '0
          0 0 15 3 ?
   20
      2
      4
```

```
'
;

s
t
a
t
i
c

t
e
s
t
m
e
t
h
o
d

v
o
i
d

t
e
s
t
S
c
h
e
d
u
l
e
d
J
o
```

```
b
(
)

{

L
i
s
t
<
L
e
a
d
>

l
e
a
d
s
=

n
e
w

Lis
t<L
ea
d>
();
for
(In
te
g
er
i =
0; i
<
```

```
20
0;
i+
+){
  Le
    a
    d

    l

    =

    n
    e
    w

    L
    e
    a
    d
    (

    F
    i
    r
    s
    t
    N
    a
    m
    e

    =

    '
    F
    i
    r
    s
    t
    '
```

```
        + i,
        La
        st
        Na
        m
        e =
        'La
        st
        Na
        m
        e',
        Co
        m
        pa
        ny
        =
        'T
        he
        In
        c'
    );
    leads.add(l);
}
ins
ert
le
ad
s;
Te
st.
sta
rt
Te
st(
);
String jobId
      =System.schedule(
      'ScheduledApexTe
```

```
            st',CRON_EXP,new
            DailyLeadProcesso
            r());
            Test.stopTest();
        List<Lead> checkleads = new List<Lead>();

        checkleads = [SelectIdFrom Lead Where
        LeadSource = 'Dreamforce' and Company
        = 'TheInc'];
        System.assertEquals(200,checkleads.size
        (),'Leads were not created');

    }
}
```

G R A TI O N S E R VI C E S

AnimalLocator.apxc:

```
public class AnimalLocator{
```

A P E X IN TE

pu b l i c s t a t i

```
c

S
t
r
i
n
g

g
e
t
A
n
i
m
a
l
N
a
m
e
B
y
I
d
(
I
n
t
e
g
e
r

x
)
{

H
t
t
p
h
t
t
p

=

n
e
w

H
t
t
p
(
)
;

HttpRequest req =new HttpRequest();
req.setEndpoint('https: /th-apex-http-
callout.herokuapp.com/animals/'
+x); req.setMethod('GET');
Ma
p<
Str
in
g,
Ob
jec
t>
an
im
al=
n
ew
Ma
p<
```

```
String, Object>();
HttpResponse res = http.send(req);
  if (res.getStatusCode() == 200) {
```

APEX SPECIALIST SUPER BADGE CODES

```
  Map<String, Object> results =
(Map<String,
Object>)JSON.deserializeUntyped
(res.getBody()); animal=
(Map<String, Object>)
results.get('animal');
```

```
        }
return (String)animal.get('name');

    }

}

@isTest
private class AnimalLocatorTest{
```

AnimalLocatorTest.apxc:

```
    @isTest static vo
```

```
id AnimalLocatorMock1() { Test.setMock(Ht
tpCalloutMock.class, new AnimalLocatorMoc
k());
string result = AnimalLocator.getAnimalNameById(3);
String expectedResult =
'chicken';
System.assertEquals(result,exp
ectedResult);
    }
}
```

AnimalLocatorMock.apxc:

```
@isTest
global class AnimalLocatorMock implements
HttpCalloutMock {
    / Implementthis interface method
   global HTTPResponse
   respond(HTTPRequest request) {
       / Create a fake response
      HttpResponsere
```

```
        sponse = 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 staticstring[] country(string
    theCountry) {
        ParkService.ParksImplPort
        parkSvc = new
        ParkService.ParksImplPort();/
        removespace return
        parkSvc.byCountry(theCountry);
    }
}
```

@isTest private class

APEX SPECIALIST SUPER BADGE CODES

ParkLocatorTest.apxc:

```
    Pa
    rk
Lo
cat
or
Te
st
{
@i
sT
e
st
st
ati
cv
o
id
tes
tC
all
ou
t()
{
    Test.setMo
    ck(WebServ
    iceMock.cla
    ss, new
    ParkService
    Mock());
    String
    country=
    'United
    States';
    List<String> result =
    ParkLocator.country(country);
    List<String> parks =new
     List<String>{'Yellowstone',
     'MackinacNationalPark', 'Yosemite'};
     System.assertEquals(parks, result);
}
}
```

ParkServiceMock.apxc:

```
@isTest
global class ParkServiceMock implem
ents WebServiceMock {
    global void doInvok
```

e
(

Ob
je
ct
st
u
b,
Ob
je
ct
re
qu
es
t,
Map<String, Object>

response, String endpoint,

Str
i
ng
so
ap
Ac
tio
n,
Str
i
ng
re
qu

es
tN
a
m
e,
Str
i
ng
re
sp
on
se
N
S,
Str
i
ng
re
sp
on
se
Na
m
e,
Str
in
gr
es
po
ns
eT
yp
e){
/start -specifythe response you want to send

ParkService.byCountryResponse
response_x = new
ParkService.byCountryResponse();
response_x.return_x = new
List<String>{'Yellowstone', 'Mackinac

NationalPark', 'Yosemite'};
/ e
n
d

r
e
s
p
o
n
s
e
.
p
u
t
(
'
r
e
s
p
o
n
s
e
_
x
'
,
r
e

s
p
o
n
s
e
_
x
)
;

}
}

AccountManager.apxc:

@
Re
st
Re
so
urc
e(
url
Ma
pp
ing
='/
Ac
co
un
ts/
*/
co
nt

```
ac
ts')
gl
ob
al
cla
ss
Ac
co
un
tM
an
ag
er
{
  @HttpGet
  gl
    o
    b
    a
    l

    s
    t
    a
    t
    i
    c

    A
    c
    c
    o
    u
    n
    t

    g
    e
    t
```

```
A
c
c
o
u
n
t
(
)

{
R
e
s
t
R
e
q
u
e
s
t

r
e
q

=

R
e
s
t
C
o
n
t
e
x
t
.
```

```
request;

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

APEX SPECIALIST SUPER BADGE CODES

```
Account acc =
        [SELECTId, Name, (SELECTId, Name FROM Contacts) FROM AccountWHERE Id = :accId];
```

```
        return acc;
    }
}
```

Account Manager Test.apxc:

```
@isTest
private class AccountManagerTest {

    private static testM
```

```
ethod void getAccountTest1() {
Id recordId

= createTestRecord();
/ Set up a test request
RestRequest request= new RestRequest();
request.requestUri= 'https:
/na1.salesforce.com/services/ap
exrest/Accounts/'+ recordId
+'/contacts' ;
request.httpMethod = 'GET
```

```
';
Re
stC
on
tex
t.r
eq
ue
st=
re
qu
es
t;
 / Call the method to test

Account thisAccount =
AccountManager.getAccount();

 / V
e
r
i
f
y

r
e
s
u
l
t
s

S
y
s
t
e
m
.
a
s
s
e
r
t
(
t
h
i
s
A
c
c
o
u
n
t

!
=
n
u
l
l
)
;

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


}


 / Helper method
 static Id createTestRecord() {
  / Create test record
 Ac
  c
  o
  u
```

nt TestAcc = new Account(Name='Test recor	d'); insert TestAcc; ContactTestCon=new Contact(LastName='Test', AccountId

```
    =
    Te
    st
    Ac
    c.i
    d);
    ret
    u
    rn
    Te
    st
    Ac
    c.I
    d;
  }
}
```

APEX SPECIALIST SUPER BADGE CODES

APEX SPECIALIST SUPER BADGE

Ch all en g e-1

MaintenanceRequ estHelper.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 (nonUpdCa
        seMap.get(
        c.Id).Status
        != 'Closed'
        &&
        c.Status ==
        'Closed'){ if
        (c.Type ==
        'Repair' ||
        c.Type ==
        'Routine
        Maintenan
        ce'){
          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)
                                        FR
        OM Case
        WHERE Id IN
        :validIds]);
        Map<Id,Deci
        mal>
        maintenance
        Cycles = new
        Map<ID,Deci
        mal>();Aggreg
        ateResult[]
        results =
        [SELECT
        Maintenance
        _Request  c,
MIN(Equipmentr.Maintenance_Cyclec
)cycle FROM
Equipment_Maintenance_Item c
WHEREMaintenance_Request c IN
:ValidIds GROUP BY
Maintenance_Request c];


      for (AggregateResult ar : results){

        maintenanceCycles.put((Id)ar.get('Main
        tenance_Request__c'),(Decimal)
        ar.get('cycle'));

      }


        for
          (
          C
          a
          s
          e

          c
          c

          :

          c
          l
          o
          s
          e
          d
          C
          a
          s
          e
          s
          M
          .
          v
          a
          l
          u
          e
          s
          (
          )
          )
          {
```

C
a
s
e

n
c

=

n
e
w

C
a
s
e

(

Pa
r
e
n
t
I
d

=
c
c
.
I
d
,

S
t
a
t

u
s

='New',

APEX SPECIALIST SUPER BADGE CODES

Su
bj
e
ct
=
'R
ou
tin
eM
ai
nt
en
an
ce
',
Ty
pe
=
'R
ou
ti
ne
Ma
int

```
en
an
ce
',
Ve
hic
le
c =
cc.
Ve
hic
le
c,
Eq
ui
p
m
e
nt
c
=c
c.E
qu
ip
m
e
nt
c,
Ori
g
in
=
'W
eb
',
Date_Reportedc = Date.Today()
```

```
            );

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

            newCases.add(nc);
         }

        insert newCases;

        List<Equipment_Maintenance_Item c>
clonedWPs = new
List<Equipment_Maintenance_Item  c>();
        for (Casenc : newCases){
            for (Equipment_Maintenance_Item c
wp :
closedCasesM.get(nc.ParentId).Equipment_M
aintenance_Items  r){
               Eq
               ui
               p
               m
               en
               t_
```

```
Ma
int
en
an
ce
_It
e
m
c
wp
Cl
o
ne
=
wp
.cl
on
e(
);
wp
Cl
on
e.
Ma
int
en
an
ce
_R
eq
ue
st
c =
nc.
Id;
Cl
on
ed
W
Ps.
ad
```

```
                d(
                wp
                Cl
                on
                e);

              }
            }
            insert ClonedWPs;
          }
        }
      }
```

APEX SPECIALIST SUPER BADGE CODES

MaintenanceRequest.apxt:

```
trigger
   MaintenanceReq
   uest on Case
   (before update,
   after update) {
   if(Trigger.isUpda
   te &&
   Trigger.isAfter){
        MaintenanceRequestHelper.updateWork
        Orders(Trigger.New, Trigger.OldMap);
    }
}
```

MaintenanceRequestHelperTest.apxc:

```
@i
s t
e s
t
 public with sharing class
 MaintenanceRequestHelperTest {

    pri
    va
    te
    sta
    tic
    fin
    al
    str
    i
    ng
    ST
    AT
    US
    _N
    EW
    ='
    Ne
    w';
    pri
    va
    te
    sta
```

```
tic
fin
al
str
i
ng
W
OR
KI
N
G=
'W
or
ki
ng
';
pri
va
te
sta
tic
fin
al
str
i
ng
CL
OS
ED
=
'Cl
os
ed
';
pri
va
te
sta
tic
fin
al
```

str
i
ng
RE
PA
IR
=
'Re
pa
ir';
private staticfinal string REQUEST_ORIGIN =
'Web';

p
ri
v
a
te
st
at
ic
fi
n
al
st
ri
n
g
R
E
Q
U
E
S
T
_
T
Y
P
E
=

'R
o
u
ti
n
e
M
ai
n
te
n
a
n
c
e
';
p
ri
v
a
te
st
at
ic
fi
n
al
st
ri
n
g
R
E
Q
U
E
S
T
_
S
U

```
BJ
E
C
T
=
'T
e
st
in
g
s
u
bj
e
ct
';

PRIVATE STATICVehicle_c createVehicle(){
    Ve
    hic
    le
    c
    Ve
    hic
    le=
    n
    ew
    Ve
    hic
    le
    C(
    na
    m
    e =
    'Su
    pe
    rTr
    uc
    k');
    ret
    u
    rn
    Ve
    hic
    le;
}

PRIVATE STATIC Product2 createEq(){
    product2equipment = new product2(name
    ='SuperEquipment',
                lif
                es
                pa
                n_
                m
                on
                t
                hs
                C =
                10,
                m
                ai
                nt
                en
                an
                ce
                _c
                yc
                le
                C
                = 10,
                re
                pl
                ac
                e
                m
                en
                t_
                pa
                rt
                c =
                tru
                e);
    return equipment;
}


PRIVATE STATIC
    Case
    createMaintena
    nceRequest(id
    vehicleId, id
    equipmentId){
    case cs = new
    case(Type=REP
    AIR,
            St
            at
            us
            =S
            TA
            TU
            S_
            NE
            W,
            Ori
            gi
            n=
            RE
            QU
            ES
            T_
            OR
            IGI
            N,
            Su
            bj
            ec
            t=
            RE
            QU
            ES
            T_
            SU
            BJ
            EC
            T,
            Equipment_c=equipmentId,
```

APEX SPECIALIST SUPER BADGE CODES

```
            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,

                                        Ma
                                        int
                                        en
                                        an
                                        ce_
                                        Re
                                        qu
                                        est_
                                        c =
                                        req
                                        ue
                                        stI
                                        d);

    return wp;
  }

  @istest
  private static void

    tes
    tM
    ai
    nt
    en
    an
    ce
    Re
```

```
qu
est
Po
siti
ve(
){
Ve
hic
lec
ve
hic
le=
cr
ea
te
Ve
hic
le(
);
insert vehicle;
id vehicleId = vehicle.Id;

Pr
od
uc
t2
eq
ui
p
m
e
nt
=
cre
at
eE
q(
);
ins
```

ert
eq
ui
p
m
en
t;
id
eq
ui
p
m
en
tId
=e
qu
ip
m
en
t.I
d;

case
somethingToUp
date =
createMaintena
nceRequest(ve
hicleId,equipm
entId);
insertsomethin
gToUpdate;

Equipment_Maintenance_Item c workP
=createWorkPart(equipmentId,somethin
gToUpdate.id); insert workP;

tes
t.s
tar
tT
es
t();
so
m
et
hi
ng
To
Up
da
te.
st
at
us
=
CL
OS
E
D;
up
da
te
so
m
et
hi
ng
To
Up
da
te;
tes
t.s
to
pT
es
t();

Case newReq = [Select id, subject, type,

Equipment__c, Date_Reported__c,
Vehicle__c,

Date_Due_c

from case

where status =:STATUS_NEW];

APEX SPECIALIST SUPER BADGE CODES

Equipment_Maintenance_Item_c
workPart = [select id

from
Equipment_Maint
enance_Item c

where
Maintenance_Req
uest__c
=:newReq.Id];

syst

em.a
ssert
(wor
kPa
rt !=
null
);
syst
em.a
ssert
(new
Req.
Subj
ect
!=
null
);
syste
m.as
sertE
qual
s(ne
wRe
q.Ty
pe,
REQ
UEST
_TYP
E);
SYST
EM.a
ssert
Equa
ls(ne
wRe
q.Eq
uip
me
nt c,
equi
pme

```
    ntId
    );
    SYST
    EM.a
    ssert
    Equa
    ls(ne
    wRe
    q.Ve
    hicle
    c,
    vehi
    cleId
    );
    SYSTEM.assertEquals(newReq.Date_Rep
    orted__c, system.today());
}

@istest
pri
    v
    a
    t
    e

    s
    t
    a
    t
    i
    c

    v
    o
    i
    d

    t
    e
```

```
s
t
M
a
i
n
t
e
n
a
n
c
e
R
e
q
u
e
s
t
N
e
g
a
t
i
v
e
(
)
{

V
e
h
i
c
l
e

C
```

```
v
e
h
i
c
l
e
=
c
r
e
a
t
e
V
e
h
i
c
l
e
(
)
;

insert vehicle;
id vehicleId = vehicle.Id;


pr
od
uc
t2
eq
ui
p
m
e
nt
```

```
=
cre
at
eE
q(
);
ins
ert
eq
ui
p
m
en
t;
id
eq
ui
p
m
en
tId
=e
qu
ip
m
en
t.I
d;


case emptyReq
=
createMaintena
nceRequest(ve
hicleId,equipm
entId);
insertemptyRe
q;


Equipment_Mai
```

ntenance_Item
c workP
=createWorkPar
t(equipmentId,
emptyReq.Id);
insertworkP;

tes
t.s
tar
tT
es
t();
e
m
pt
yR
eq
.St
at
us
=W
OR
KI
N
G;
up
da
te
e
m
pt
yR
e
q;
tes
t.s
to
pT
es

t();

list<case> allRequest = [select id

from case];

Equipment_Maintenance_Item_c
workPart = [select id

fromEquipment_Mai
ntenance_Item c

APEX SPECIALIST SUPER BADGE CODES

where
Maintenance_Request_
c = :emptyReq.Id];

sy
ste
m.
as

```
    se
    rt(
    wo
    rk
    Pa
    rt
    !=
    nu
    ll);
    sy
    ste
    m.
    as
    se
    rt(
    all
    Re
    qu
    es
    t.si
    ze
    ()
    ==
    1);
}


@istest
priv
    a
    t
    e
    st
    a
    t
    ic
    v
    o
    i
    d
```

```
t
e
st
M
ai
n
t
e
n
a
n
c
e
R
e
q
u
e
st
B
u
lk
(
){
li
st
<
V
e
h
ic
le
C
>
v
e
h
ic
le
Li
st
```

```
=
n
e
w
li
st
<
V
e
h
ic
le
C
>(
);
li
st
<
P
r
o
d
u
ct
2
>
e
q
ui
p
m
e
n
tL
i
st
=
n
e
w
li
```

```
st
<
P
r
o
d
u
ct
2
>(
);
li
st
<
E
q
ui
p
m
e
n
t
_
M
ai
n
t
e
n
a
n
c
e
_I
t
e
m
c
>
w
o
```

```
r
k
P
a
rt
Li
st
= n
e
w

l
i
s
t
<
E
q
u
i
p
m
e
n
t
_
M
a
i
n
t
e
n
a
n
c
e
_
I
t
e
```

```
m

c
>
(
)
;
lis
t<c
as
e>
re
qu
est
Li
st
=
ne
wli
st<
ca
se
>(
);
lis
t<i
d>
ol
dR
eq
ue
stI
ds
=n
ew
lis
t<i
d>
();
```

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

```
List.add(createVehicle());equipmentList.a
```

```
  d
  d
  (
  c
  r
  e
  a
  t
  e
  E
  q
  (
  )
  )
  ;
}
insert
```

```
ve
hic
leL
ist;
ins
ert
eq
ui
p
m
en
tLi
st;
```

```
for(integer i = 0; i < 300; i++){
    requestList.add(createMaintenanceReq
    uest(vehicleList.get(i).id,
    equipmentList.get(i).id));
}
insert requestList;


for(integer i =
    0; i < 300;
    i++){
    workPartList.
    add(createWo
    rkPart(equip
    mentList.get(i
    ).id,
    requestList.g
    et(i).id));
}
insert workPartList;
```

```
tes
t.s
tar
tTe
st(
);
for
(ca
se
r
eq
:
    requestList){
    re
    q.
    St
    at
    us
    =
    CL
    OS
    E
    D;
    ol
    dR
    eq
    ue
    stI
    ds.
    ad
    d(r
    eq
    .Id
    );
}
update requestList;
```

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

      list<Equipment_Maintenance_Item_c>
      workParts = [select id
                        fromEquipme
                        nt_Maintenan
                        ce_Item c
                        where
                        Maintenance_
                        Request c in:
                        oldRequestIds
                        ];

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

```
test.stopTest();

list<case> allRequests = [select id
```

WarehouseCallou tService.apxc:

```
public with
   sharingclass
   WarehouseCallou
   tService
   implements
   Queueable {
   private static final
   String
   WAREHOUSE_URL
   = 'https:
   /t
h
-
s
u
p
e
r
b
a
d
g
e
-

a
p
e
x
.
h
e
r
o
k
u
a
p
p
.
c
o
m
/
e
q
u
i
p
m
e
n
t
'
;
```

```
    /class that makesaREST callout to an
externalwarehouse system to get a list of
equipmentthat needs to be updated.
    /The callout’s JSON response
    returns the equipmentrecords
    that you upsert inSalesforce.

   @f
   ut
   ur
   e(c
   all
   ou
   t=t
   ru
```

e)

pu

bl

ic

st

ati

cv

o

id

ru
n
Wa
re
ho
us
eE
qu
ip
m
en
tS
yn
c(
){
Ht
tp
htt
p=
n
ew
Ht
tp(
);
HttpRequest request=new
HttpRequest();

re
qu
est
.se
tE
nd
po
int
(W
AR
EH
OU
SE
_U
RL
);
re
qu
est
.se
tM
et
ho
d('
GE
T');
Ht
tp
Re
sp
on
se
re
sp
on
se
=

htt

p.s

en

d(r

eq

ue

st

);

Lis

t<

Pr

od

uc

t2>

wa

re

ho

us

e

Eq

= n

ew

Lis

t<

Pr

od

uc

t2>

();

if

(re

sp

on

se.

ge

tSt

at

us

Co

de

()

==

20

0){

List<Object>
jsonRespon
se
=(List<Object
>)JSON.deseri
alizeUntyped
(response.get
Body());

APEX SPECIALIST SUPER BADGE CODES

System.debug(response.getBody());

/class maps the following fields:replacement part (alwaystrue), cost, currentinventory, lifespan, maintenance cycle, and warehouse SKU
/warehouse SKU will be external ID for identifying which equipment records toupdate withinSalesforce
for (Object eq : jsonResponse){

M
a
p
<
S

tring,Object> mapJson =(Map<String,Object	>)eq;Product2 myEq = newProduct2();myEq.Replacement_Part c =(Boolean)mapJson.get('replace

```
ment');
myEq.Na
me =
(String)
mapJson.g
et('name');
myEq.Maintenance_Cycle c = (Integer)
mapJson.get('maintenanceperiod');
myEq.Lifespan_Months c = (Integer)
mapJson.get('lifespan');
myEq.C
ost c =
(Intege
r)
mapJs
on.get(
'cost');
myEq.
Wareh
ouse_S
KU c =
(Strin
g)
mapJs
on.get(
'sku');
myEq.C
urrent
_Inven
tory c =
(Doubl
e)
mapJs
on.get('
quantit
y');
myEq.
Produc
tCode
= (Strin
g)
```

```
    mapJs
    on.get('
    _id');
    wareh
    ouseEq
    .add(m
    yEq);
}

if
    (w
    ar
    eh
    ou
    se
    Eq.
    si
    ze
    ()>
    0){
    up
    se
    rt
    wa
    re
    ho
    us
    eE
    q;
    System.debug('Your equipmentwas
    synced with the warehouse one');
}
```

```
        }
    }

    public static void
        execute
        (QueueableCo
        ntext context){
        runWarehouse
        EquipmentSyn
        c();
    }

}

@isTest
```

WarehouseCalloutServiceMock.apxc:

```
global classWarehouseCalloutServiceMock
implements HttpCalloutMock {
     / implement http mock callout
    global staticHttpResponse
    respond(HttpRequest request){
```

SPECIALIST SUPER BADGE CODES

```
Ht
tp
Re
sp
on
se
re
sp
on
se
= n
ew
Ht
tp
Re
sp
on
se(
);
res
po
ns
e.s
et
He
ad
er(
'Co
nt
en
t-
Ty
pe
',
'ap
```

APEX

```
        pli
        cat
        io
        n/
        jso
        n');



response.setBody('[{"_id":"55d66226726b61110
0aaf741","replacement":false,"quantity":5,"nam
e"
:"Gene    rator            1000
kW","maintenanceperiod":365,"lifespan":120,"co
st":5000,"sku":"100003"},{"_id":"55d66226726b61
1100a
af742","replacement":true,"quantity":183,"nam
e":"Cooling
Fan","maintenanceperiod":0,"lifespan":0,"cost":
300,"sku":"100004"},{"_id":"55d66226726b611100
aaf743
","replacement":true,"quantity":143,"name":"F
use
20A","maintenanceperiod":0,"lifespan":0,"cost"
:22,"sku":"100005"}]');
        response.setStatusCode(200);


        return response;
    }
}
```

WarehouseCalloutServiceTest.apxc:

```
@IsTest
private class WarehouseCalloutServiceTest {
    / i
        m
        p
        l
        e
        m
        e
        n
        t

        y
        o
        u
        r

        m
        o
        c
        k

        c
        a
        l
        l
        o
        u
        t

        t
        e
        s
        t
        h
        e
        r
```

e

@
i
s
T
e
s
t

sta
t
i
c

v
o
i
d

t
e
s
t
W
a
r
e
h
o
u
s
e
C
a
l
l
o
u
t
(

)

{
t
e
s
t
.
s
t
a
r
t
T
e
s
t
(
)
;

test.setMock(HttpCalloutMock.class,new
WarehouseCalloutServiceMock());
WarehouseCalloutService.execute(null);
test.stopTest();

List<
Prod
uct
2>
prod
uct2
List
=
new
List<
Prod
uct2
>();p
rodu

```
        ct2Li
        st =
        [SEL
        ECTP
        rodu
        ctCo
        de
        FR
        OM
        Prod
        uct2
        ];

        System.assertE
        quals(3,
        product2List.siz
        e());
        System.assertE
        quals('55d66226
        726b611100aaf7
        41',
        product2List.ge
        t(0).ProductCod
        e);
        System.assertE
        quals('55d66226
        726b611100aaf7
        42',
        product2List.ge
        t(1).ProductCod
        e);
        System.assertE
        quals('55d66226
        726b611100aaf7
        43',
        product2List.ge
        t(2).ProductCod
        e);
    }
}
```

Challenge-3

WarehouseSyncSchedule.apxc:

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

APEX SPECIALIST SUPER BADGE CODES

```
    global void execute(SchedulableContext
        ctx){
        System.enqueueJob(newWarehouseCallo
        utService());
    }
}
```

WarehouseSyncScheduuleTest.apxc:

```
@isTest
public class WarehouseSyncScheduleTest {
```

```
@isTest static void WarehouseScheduleTest() {
    String scheduleTime = '00 00 01 * * ?
'
;

T
e
s
t
.
s
t
a
r
t
T
e
s
t
(
)
;

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

String
jobID=System.schedule('Warehouse Time To
Scheduleto Test', scheduleTime, new
WarehouseSyncSchedule());
Test.stopTest();

 /Contains schedule information for a
scheduledjob. CronTrigger is similarto 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.assertE
quals(jobID,
a.Id,'Schedule
');


    }
}
```

Challenge-4

MaintenanceRequestHelperTest.apxc:

```
@istest
public with sharing class
MaintenanceRequestHelperTest {


  pri
  va
  te
  sta
  tic
  fin
  al
  str
  i
  ng
  ST
```

AT
US
_N
EW
='
Ne
w';
pri
va
te
sta
tic
fin
al
str
i
ng
W
OR
KI
N
G=
'W
or
ki
ng
';
pri
va
te
sta
tic
fin
al
str
i
ng
CL
OS
ED
=

'Cl
os
ed
';
pri
va
te
sta
tic
fin
al
str
i
ng
RE
PA
IR
=
'Re
pa
ir';
private staticfinal string REQUEST_ORIGIN =
'Web';

p
ri
v
a
te
st
at
ic
fi
n
al
st
ri
n
g
R
E

Q
U
E
S
T
_
T
Y
P
E
=
'R
o
u
ti
n
e
M
ai
n
te
n
a
n
c
e
';
p
ri
v
a
te
st
at
ic
fi
n
al
st
ri
n

g
R
E
Q
U
E
S
T
_
S
U
BJ
E
C
T
=
'T
e
st
in
g
s
u
bj
e
ct
';

PRIVATE STATICVehicle__c createVehicle(){

APEX SPECIALIST SUPER BADGE CODES

```
    Ve
    hic
    le
    c
    Ve
    hic
    le=
    n
    ew
    Ve
    hic
    le
    C(
    na
    m
    e =
    'Su
    pe
    rTr
    uc
    k');
    ret
    u
    rn
    Ve
    hic
    le;
}


PRIVATE STATIC Product2 createEq(){
    product2 eq
                u
                i
                p
                m
                e
                n
                t
```

```
=
n
e
w

p
r
o
d
u
c
t
2
(
n
a
m
e

=
'
S
u
p
e
r
E
q
u
i
p
m
e
n
t
'
,

l
i
f
```

```
e
s
p
a
n
_
m
o
n
t
h
s

C

=

1
0
,

m
a
i
n
t
e
n
a
n
c
e
_
c
y
c
l
e

C
```

```
                        = 10,
                        re
                        pl
                        ac
                        e
                        m
                        en
                        t_
                        pa
                        rt
                        c =
                        tru
                        e);
   return equipment;
}
```

```
PRIVATE STATIC
   Case
   createMaintena
   nceRequest(id
   vehicleId, id
   equipmentId){
   case cs = new
   case(Type=REP
   AIR,
                St
                at
                us
                =S
                TA
```

```
TU
S_
NE
W,
Ori
gi
n=
RE
QU
ES
T_
OR
IGI
N,
Su
bj
ec
t=
RE
QU
ES
T_
SU
BJ
EC
T,
Eq
ui
p
m
e
nt
c=
eq
ui
p
m
en
tI
d,
Ve
```

```
                    hic
                    lec
                    =v
                    eh
                    icl
                    eI
                    d);
        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 =
eq
        u
        i
        p
        m
        e
        n
        t
        I
        d
        ,

        M
        a
        i
        n
        t
        e
        n
        a
        n
        c
```

```
e_Request c =request Id);
returnwp;
}

@istest
```

```
private static void testMaintenanceRequest
```

```
Positive(){
Vehiclecvehicle =
createVehicle();
insert vehicle;
id vehicleId = vehicle.Id;

Product2equipment=createEq();
insertequipment;
idequipmentId
```

=e
qu
ip
m
en
t.I
d;

APEX SPECIALIST SUPER BADGE CODES

case
somethingToUp
date =
createMaintena
nceRequest(ve
hicleId,equipm
entId);
insertsomethin
gToUpdate;

Equipment_Maintenance_Item c workP
=createWorkPart(equipmentId,somethin
gToUpdate.id); insert workP;

tes
t.s
tar
tT

es
t();
so
m
et
hi
ng
To
Up
da
te.
st
at
us
=
CL
OS
E
D;
up
da
te
so
m
et
hi
ng
To
Up
da
te;
tes
t.s
to
pT
es
t();

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
                    fromEquipment_
                    Maintenance_It
                    em c
                    where
                    Maintenance_Req
                    uest__c
                    =:newReq.Id];

    syst
    em.a
    ssert
    (wor
    kPa
    rt !=
    null
    );
    syst
    em.a
    ssert
    (new
    Req.
    Subj
    ect
    !=
    null
    );
    syste
    m.as
    sertE
    qual
```

```
    s(ne
    wRe
    q.Ty
    pe,
    REQ
    UEST
    _TYP
    E);
    SYST
    EM.a
    ssert
    Equa
    ls(ne
    wRe
    q.Eq
    uip
    me
    nt c,
    equi
    pme
    ntId
    );
    SYST
    EM.a
    ssert
    Equa
    ls(ne
    wRe
    q.Ve
    hicle
    c,
    vehi
    cleId
    );
    SYSTEM.assertEquals(newReq.Date_Rep
    orted___c, system.today());
}

@istest
```

```
private static void

    tes
    tM
    ai
    nt
    en
    an
    ce
    Re
    qu
    est
    Ne
    ga
    tiv
    e(
    ){
    Ve
    hic
    le
    C
    ve
    hic
    le=
    cr
    ea
    te
    Ve
    hic
    le(
    );
    insert vehicle;
```

```
id vehicleId = vehicle.Id;

pr
od
uc
t2
eq
ui
p
m
e
nt
=
cre
at
eE
q(
);
ins
ert
eq
ui
p
m
en
t;
id
eq
ui
p
m
en
tId
=e
qu
ip
m
en
t.I
```

d;

APEXSPECIALIST SUPER BADGE CODES

case emptyReq
=
createMaintena
nceRequest(ve
hicleId,equipm
entId);
insertemptyRe
q;

Equipment_Mai
ntenance_Item
c workP
=createWorkPar
t(equipmentId,
emptyReq.Id);
insertworkP;

tes
t.s
tar
tT
es
t();
e

m
pt
yR
eq
.St
at
us
=W
OR
KI
N
G;
up
da
te
e
m
pt
yR
e
q;
tes
t.s
to
pT
es
t();

list<case> allRequest = [select id

from case];

Equipment_Maintenance_Item_c
workPart = [select id

fromEquipment_
Maintenance_It
em c

where
Maintenance_Req

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

```
uest_c =
:emptyReq.Id];
```

```
}

@istest
private static void testMaintenanceRequestBulk(){
```

```
li
st
<
V
e
h
ic
le
C
>
v
e
h
ic
le
Li
st
=
n
e
w
li
st
<
V
e
h
ic
le
C
>(
);
li
st
<
P
r
o
d
u
ct
```

```
2
>
e
q
ui
p
m
e
n
tL
i
st
=
n
e
w
li
st
<
P
r
o
d
u
ct
2
>(
);
li
st
<
E
q
ui
p
m
e
n
t
_
M
```

```
ai
n
t
e
n
a
n
c
e
_I
t
e
m
c
>
w
o
r
k
P
a
rt
Li
st
= n
e
w

l
i
s
t
<
E
q
u
i
p
m
e
n
```

```
t
_
M
a
i
n
t
e
n
a
n
c
e
_
I
t
e
m

c
>
(
)
;
lis
t<c
as
e>
re
qu
est
Li
st
=
ne
wli
st<
ca
se
>(
);
lis
t<i
d>
ol
dR
eq
ue
stI
ds
=n
ew
lis
t<i
d>
();

for
  (
  i
  n
  t
  e
  g
  e
  r

  i

  =

  0
  ;

  i

  <

  3
  0
```

```
0
;

i
+
+
)
{

v
e
h
i
c
l
e
L
i
s
t
.
a
d
d
(
c
r
e
a
t
e
V
e
h
i
c
l
e
(
)
)
```

```
;
e
q
u
i
p
m
e
n
t
L
i
s
t
.
a
d
d
(
c
r
e
a
t
e
E
q
(
)
)
;

}

ins
ert
ve
hic
leL
ist;
ins
```

```
ert
eq
ui
p
m
en
tLi
st;
```

APEX SPECIALIST SUPER BADGE CODES

```
for(integer i = 0; i < 300; i++){
    requestList.add(createMaintenanceReq
    uest(vehicleList.get(i).id,
    equipmentList.get(i).id));
}
```

```
insert requestList;
```

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

```
    workPartList.
    add(createWo
    rkPart(equip
    mentList.get(i
    ).id,
    requestList.g
    et(i).id));
}
insert workPartList;


tes
t.s
tar
tTe
st(
);
for
(ca
se
r
eq
:
    re
    qu
    est
    Lis
    t){
    re
    q.
    St
    at
    us
    =
    CL
    OS
    E
    D;
    ol
    dR
    eq
    ue
    stI
    ds.
    ad
    d(r
    eq
    .Id
    );
}
up
da
ter
e
qu
es
tLi
st;
tes
t.st
op
T
es
t();


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


list<Equipment_Maintenance_Item_c>
workParts = [select id
```

```
                                   fromEquipment_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'){
                    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)
```

```
                                   FR
     OM Case
     WHERE Id IN
     :validIds]);
     Map<Id,Deci
     mal>
     maintenance
     Cycles = new
     Map<ID,Deci
     mal>();
     AggregateRes
     ult[] results=
     [SELECT
     Maintenance
     _Request  c,
MIN(Equipmentr.Maintenance_Cyclec
)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('Main
     tenance_Request__c'),(Decimal)
     ar.get('cycle'));
   }


     for
```

```
(
C
a
s
e

c
c

:

c
l
o
s
e
d
C
a
s
e
s
M
.
v
a
l
u
e
s
(
)
)
{
C
a
s
e

n
c
```

```
= new Case (
    ParentId = cc.Id,
Status ='New',
    Su
```

```
bject = 'RoutineMaintenance', Type = 'RoutineMaintenance', Vehicle c = cc.Vehiclec, Equipme
```

```
    nt
    c
    =c
    c.E
    qu
    ip
    m
    e
    nt
    c,
    Ori
    g
    in
    =
    'W
    eb
    ',
    Date_Reportedc = Date.Today()


);

If
(maintenanceCycles.containskey(cc.I
d)){
    nc.Date_Due__c
    =Date.today().addDays((Integer)m
    aintenanceCycles.get(cc.Id));
```

APEX SPECIALIST SUPER BADGE CODES

```
     }

     newCases.add(nc);
 }

insert newCases;

List<Equipment_Maintenance_Item c>
clonedWPs = new

Lis
t
<
E
q
u
i
p
m
e
n
t
_
M
a
i
n
t
e
```

```
n
a
n
c
e
_
I
t
e
m

c
>
(
)
;
f
o
r

(
C
a
s
e
n
c

:

n
e
w
C
a
s
e
s
)
{
```

```
        for (Equipment_Maintenance_Item c
wp :
closedCasesM.get(nc.ParentId).Equipment_M
aintenance_Items r){
          Eq
          ui
          p
          m
          en
          t_
          Ma
          int
          en
          an
          ce
          _It
          e
          m
          c
          wp
          Cl
          o
          ne
          =
          wp
          .cl
          on
          e(
          );
          wp
          Cl
          on
          e.
          Ma
          int
          en
          an
          ce
          _R
          eq
```

```
            ue
            st
            c =
            nc.
            Id;
            Cl
            on
            ed
            W
            Ps.
            ad
            d(
            wp
            Cl
            on
            e);

          }

        }


        insert ClonedWPs;

      }

    }

  }
```

Ch
all
en
g
e-5

W
ar
eh
ou
se
Ca
llo
ut
Se
rvi
ce.
ap
xc:

```
public with sharing
  classWarehouseC
  alloutService
  implements
  Queueable {
  private static final
  String
  WAREHOUSE_URL
  = 'https:
  /t
h
-
s
u
p
e
r
b
a
d
g
e
-

a
```

p
e
x
.
h
e
r
o
k
u
a
p
p
.
c
o
m
/
e
q
u
i
p
m
e
n
t
'
;

/class that makesaREST callout to an externalwarehouse system to get a list of equipmentthat needs to be updated.

/The callout’s JSON response returns the equipmentrecords that you upsert inSalesforce.

@f

ut
ur
e(c
all
ou
t=t
ru
e)
pu
bl
ic
st
ati
cv
o
id

ru
n
Wa
re
ho
us
eE
qu
ip
m
en
tS

yn
c(
){
Ht
tp
htt
p=
n
ew
Ht
tp(
);
HttpRequest

request = new

HttpRequest();r

equest.setEndp

oint(WAREHOUS

E_URL);

APEX SPECIALIST SUPER BADGE CODES

re

qu

est

.se

tM

et

ho

d('

GE

T');

Ht

tp

Re

sp

on

se

re

sp

on

s

e=

htt

p.s

en

d(r

eq

ue

st

```
);
Lis
t<
Pr
od
uc
t2>
wa
re
ho
us
e
Eq
= n
ew
Lis
t<
Pr
```

```
od
uc
t2>
();
if (response.getStatusCode() == 200){
    List<Object>
    jsonRespon
    se
    =(List<Object
    >)JSON.deseri
    alizeUntyped
    (response.get
    Body());
    System.debu
    g(response.g
    etBody());
```

/class maps the following fields:replacement part (alwaystrue), cost, currentinventory, lifespan, maintenance cycle, and warehouse SKU

/warehouse SKU will be external ID for identifying which equipment records toupdate withinSalesforce

```
    for (Object eq : jsonResponse){
        M
        a
        p
        <
        S
        t
        r
        i
```

ng,Object> mapJson =(Map<String,Object>)e	q;Product2 myEq = newProduct2();myEq.Replacement_Part c = (Boolean) mapJson.get('replacement'); myEq.Name =

```
(String)
mapJson.g
et('name');

myEq.Maintenance_Cycle c = (Integer)
mapJson.get('maintenanceperiod');
myEq.Lifespan_Months c = (Integer)
mapJson.get('lifespan');
myEq.C
ost c =
(Intege
r)
mapJs
on.get(
'cost');
myEq.
Wareh
ouse_S
KU c =
(Strin
g)
mapJs
on.get(
'sku');
myEq.C
urrent
_Inven
tory c =
(Doubl
e)
mapJs
on.get('
quantit
```

```
    y');
    myEq.
    Produc
    tCode
    = (Strin
    g)
    mapJs
    on.get('
    _id');
    wareh
    ouseEq
    .add(m
    yEq);
  }


  if
    (w
    ar
    eh
    ou
    se
    Eq.
    si
    ze
    ()>
    0){
    up
    se
    rt
    wa
    re
    ho
    us
    eE
    q;
    System.debug('Your equipmentwas
    synced with the warehouse one');
  }
}
```

```
    }

    public static void
        execute
        (QueueableCo
        ntext context){
        runWarehouse
        EquipmentSyn
        c();
    }

}

@isTest
```

APEXSPECIALIST SUPER BADGE CODES

<u>WarehouseCalloutServiceMock.apxc:</u>

```
global classWarehouseCalloutServiceMock
implements HttpCalloutMock {

    / implement http mock callout

global staticHttpResponse
respond(HttpRequest request){

    Ht
    tp
    Re
    sp
    on
    se
    re
    sp
    on
    se
    =
    n
    ew
    Ht
    tp
    Re
    sp
    on
    se(
    );
    res
    po
    ns
    e.s
    et
    He
    ad
```

```
er(
'Co
nt
en
t-
Ty
pe
',
'ap
pli
cat
io
n/
jso
n');
```

```
response.setBody('[{"_id":"55d66226726b
611100aaf741","replacement":false,"quant
ity":5,"na me":"Gene    rator
1000kW","maintenanceperiod":365,"lifes
pan":120,"cost":5000,"sku":"100003"},{"_id
":"55d6622672
6b611100aaf742","replacement":true,"qu
antity":183,"name":"Cooling
Fan","maintenanceperiod":0,"lifespan":0,"
cost":300,"sku":"100004"},{"_id":"55d66226
726b611 100aaf743
","replacement":true,"quantity":143,"na
me":"Fuse
20A","maintenanceperiod":0,"lifespan":0,
"cost":22,"sku":"100005"}]');
    response.setStatusCode(200);


    return response;

  }

}
```

WarehouseCalloutService

Test.apxc:

```
@isTest

global classWarehouseCalloutServiceMock
implements HttpCalloutMock {

   / implement http mock callout

  global staticHttpResponse
  respond(HttpRequest request){


    Ht
    tp
    Re
    sp
    on
    se
    re
    sp
    on
    se
    =
    n
    ew
    Ht
    tp
    Re
    sp
    on
    se(
    );
    res
    po
    ns
    e.s
    et
    He
    ad
    er(
    'Co
    nt
```

en
t-
Ty
pe
',
'ap
pli
cat
io
n/
jso
n');

response.setBody('[{"_id":"55d66226726b 611100aaf741","replacement":false,"quant ity":5,"na me":"Gene rator 1000kW","maintenanceperiod":365,"lifes pan":120,"cost":5000,"sku":"100003"},{"_id ":"55d6622672 6b611100aaf742","replacement":true,"qu antity":183,"name":"Cooling Fan","maintenanceperiod":0,"lifespan":0," cost":300,"sku":"100004"},{"_id":"55d66226 726b611 100aaf743 ","replacement":true,"quantity":143,"na me":"Fuse

20A","maintenanceperiod":0,"lifespan":0,"cost" :22,"sku":"100005"}]');

```
        response.setStatusCode(200);

        return response;
    }
}
```

Ch all en g e- 6

W ar eh ou se Sy nc Sc he

du le. ap xc:

```
global with sharing
   class
   WarehouseSyncS
   chedule
   implementsSche
   dulable{ global
   void
   execute(Schedul
   ableContext ctx){
      System.enqueueJob(new
      WarehouseCalloutService());
   }
}
```

WarehouseS yncSchedule Test.apxc:

```
@isTest
public class WarehouseSyncScheduleTest {

   @i
      s
      T
      e
      s
      t

      s
      t
      a
      t
      i
```

```
c

v
o
i
d

W
a
r
e
h
o
u
s
e
s
c
h
e
d
u
l
e
T
e
s
t
(
)
{

S
t
r
i
n
g
s
c
h
```

```
e
d
u
l
e
T
i
m
e

=

'
0
0

0
0

0
1

*

*

?
'
;
```

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

        String
jobID=System.schedule('Warehouse Time To
Scheduleto Test', scheduleTime, new
WarehouseSyncSchedule());
        Test.stopTest();

         /Contains schedule information for a
scheduledjob. CronTrigger is similarto 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.assertE
        quals(jobID,
        a.Id,'Schedule
        ');



    }

}
```

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