

Name - Rutuja Jarange

Project Title : Salesforce Developer Catalyst

About : All the details about Projects, Badges Completion and Codes.

Apex Specialist Super badge

TRAILHEAD Today Learn ▾ Credentials ▾ Community ▾ For Companies ▾

Search

Rutuja Jarange 35 badges, 52,900 points

Developer Super Set

+13,000 POINTS

Superbadge

Apex Specialist

Use integration and business logic to push your Apex coding skills to the limit.

☆ +

Completed 6/15/22

Prerequisites

Apex Triggers Apex Testing Asynchronous Apex Apex Integration Services Apex Specialist

APEX TRIGGERS

1. Get Started with Apex Triggers

Creation of an Apex trigger :

Given code is of an apex trigger named as "AccountAddressTrigger"

```
1 trigger AccountAddressTrigger on Account (before insert,  
   before update) {  
2     for(Account a : Trigger.New) {
```

```

3         if(a.Match_Billing_Address__c ==True){
4             a.ShippingPostalCode = a.BillingPostalCode;
5         }
6     }
7 }

```

2. Bulk Apex Triggers

Given is a bulkified Apex trigger that adds a follow-up task to an opportunity if its stage is Closed Won. Fire the Apex trigger after inserting or updating an opportunity.

Apex trigger : ClosedOpportunityTrigger

```

1 trigger ClosedOpportunityTrigger on Opportunity (after
  insert, after update) {
2     list<Task> Tl= new list<Task>();
3     for ( Opportunity o : Trigger.New){
4         if(o.StageName=='Closed Won'){
5             Tl.add(new Task(Subject = 'Follow Up Testd =
              o.ID));
6         }
7     }
8     if(Tl.size()>0){
9         insert Tl;
10    }
11 }

```

Apex Testing

1.Get Started with Apex Unit Tests

Create and install a simple Apex class to test if a date is within a proper range, and if not, returns a date that occurs at the end of the month within the range.

Apex class : VerifyDate

```
1 public class VerifyDate {
2
3     //method to handle potential checks against two dates
4     public static Date CheckDates(Date date1, Date date2) {
5         //if date2 is within the next 30 days of date1, use date2. Otherwise use the
        end of the month
6         if(DateWithin30Days(date1,date2)) {
7             return date2;
8         } else {
9             return SetEndOfMonthDate(date1);
10        }
11    }
12
13    //method to check if date2 is within the next 30 days of date1
14    @TestVisible private static Boolean DateWithin30Days(Date date1, Date date2) {
15        //check for date2 being in the past
16        if( date2 < date1) { return false; }
17
18        //check that date2 is within (>=) 30 days of date1
19        Date date30Days = date1.addDays(30); //create a date 30 days away from date1
20        if( date2 >= date30Days ) { return false; }
21        else { return true; }
22    }
23
24    //method to return the end of the month of a given date
25    @TestVisible private static Date SetEndOfMonthDate(Date date1) {
26        Integer totalDays = Date.daysInMonth(date1.year(), date1.month());
27        Date lastDay = Date.newInstance(date1.year(), date1.month(), totalDays);
28        return lastDay;
29    }
30
31 }
```

Unit Test : TestVerifyDate

```
1  @isTest
2  Private class TestVerifyDate {
3      @isTest static void Test_CheckDates_case1(){
4          Date D =
5              VerifyDate.CheckDates(Date.parse('01/01/2022'),
6              Date.parse('01/05/2022'));
7          System.assertEquals( Date.parse('01/05/2022'),D);
8      }
9      @isTest static void Test_CheckDates_case2(){
10         Date D =
11             VerifyDate.CheckDates(Date.parse('01/01/2022'),
12             Date.parse('05/05/2022'));
13         System.assertEquals( Date.parse('01/31/2022'),D);
14     }
15     @isTest static void Test_DateWithin30Days_case1(){
16         Boolean flag =
17             VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
18             date.parse('12/30/21'));
19         System.assertEquals(false,flag);
20     }
21     @isTest static void Test_DateWithin30Days_case2(){
22         Boolean flag =
23             VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
24             date.parse('02/02/21'));
25         System.assertEquals(false,flag);
26     }
27     @isTest static void Test_DateWithin30Days_case3(){
28         Boolean flag =
29             VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
30             date.parse('1/15/22'));
31         System.assertEquals(true,flag);
32     }
33     @isTest static void Test_SetEndOfMonthDate(){
34         Date returndate =
35             VerifyDate.SetEndOfMonthDate(date.parse('01/01/2022'));
36     }
37 }
```

2. Test Apex Triggers:

Create and install a simple Apex trigger which blocks inserts and updates to any contact with a last name of 'INVALIDNAME'.

Apex trigger : RestrictContactByName

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

Unit Tests : TestRestrictContactByName

```
1 @isTest  
2 public class TestRestrictContactByName {  
3     @isTest static void Test_insertupdateContact(){  
4         Contact cnt = new Contact();  
5         cnt.LastName = 'INVALIDNAME';  
6  
7         Test.startTest();  
8         Database.SaveResult result =  
9         Database.insert(cnt, false);  
10        Test.stopTest();  
11  
12        System.assert(!result.isSuccess());  
13        System.assert(result.getErrors().size() > 0);  
14        System.assertEquals('The Last Name "INVALIDNAME"
```

```
14     }  
15 }
```

3. Create Test Data for Apex Tests:

Create an Apex class that returns a list of contacts based on two incoming parameters

Apex class : RandomContactFactory

```
1 public class RandomContactFactory {  
2     public static List<Contact>  
    generateRandomContacts(Integer numcnt ,String lastname) {  
3         List<Contact> contacts= new List<Contact>();  
4         for(Integer i=0;i<numcnt;i++){  
5             Contact cnt = new Contact(FirstName = 'Test'  
+i , LastName=lastname);  
6             contacts.add(cnt);  
7         }  
8         return contacts;  
9     }  
10 }
```

Asynchronous Apex -

1. Use Future Methods -

Create an Apex class with a future method that accepts a List of Account IDs and updates a custom field on the Account object with the number of contacts associated to the Account.

Apex class : AccountProcessor

```
1 public class AccountProcessor {  
2     @future
```

```

3     public static void countContacts(List<ID>
accountIDs){
4         List<Account> accountsToUpdate = new
List<Account>();
5         List <Account> accounts =[Select Id,name, (Select
ID from contacts)from Account Where ID in:accountIds];
6         for(Account acc: accounts){
7             List<Contact>contactList = acc.Contacts;
8         acc.Number_Of_Contacts__c =contactList.size();
9             accountsToUpdate.add(acc);
10        }
11        update accountsToUpdate;
12
13    }
14 }

```

Apex test class : AccountProcessorTest

```

1  @IsTest
2  public class AccountProcessorTest {
3      @IsTest
4          private static void testCountContacts(){
5              Account newAccount = new Account(Name='Test
6
7              insert newAccount;
8              Contact newContact1= new
Contact(FirstName='John',LastName='Doe',AccountId =
newAccount.Id);
9              insert newContact1;
10             Contact newContact2=new
Contact(FirstName='J',LastName='Doe',AccountId=
newAccount.Id);
11             insert newContact2;
12             List<Id> accountIds = new List<Id>();
13             accountIds.add(newAccount.Id);
14             Test.startTest();
15             AccountProcessor.countContacts(accountIds);
16             Test.stopTest();
17         }
18     }

```

2. Use Batch Apex

Create an Apex class that implements the Database.Batchable interface to update all Lead records in the org with a specific LeadSource.

Apex class : LeadProcessor

```

1  global class LeadProcessor implements
    Database.Batchable<sObject>{
2      global integer count =0;
3
4      global Database.QueryLocator
    start(Database.BatchableContext bc){
5          return Database.getQueryLocator('SELECT ID ,
6      }
7      global void execute(Database.BatchableContext bc,
    List<Lead> L_list){
8          List<Lead> L_list_new = new List<lead>();
9          for (lead L : L_list){
10             L.leadsource= 'Dreamforce';
11             L_list_new.add(L);
12             count +=1;
13         }
14         update L_list_new;
15     }
16     global void finish(Database.BatchableContext bc){
17         system.debug('count = '+ count);
18
19     }
20
21 }
```

Apex test class : LeadProcessorTest

```

1  @isTest
```



```

2 public class LeadProcessorTest {
3
4     @isTest
5     public static void testit(){
6         List<lead> L_list = new List<lead>();
7         for (Integer i=0;i<200;i++){
8             Lead L = new lead();
9             L.LastName = 'name' +i;
10            L.Company = 'Comapany';
11            L.Status = 'Random Status';
12            L_list.add(L);
13        }
14        insert L_list;
15        Test.startTest();
16        LeadProcessor lp = new LeadProcessor();
17        Id batchId = Database.executeBatch(lp);
18        Test.stopTest();
19
20    }
21 }

```

3. Control Processes with Queueable Apex

Apex class : AddPrimaryContact

```

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

```

```

14         Contact c = con.clone();
15     c.AccountId= acc.Id;
16     primaryContacts.add(c);
17     }
18     if(primaryContacts.size()>0){
19         insert primaryContacts;
20     }
21 }
22
23 }

```

Apex test class : AddPrimaryContactTest

```

1 @isTest
2 public class AddPrimaryContactTest {
3     static testmethod void testQueueable(){
4         List<Account> testAccounts = new List<Account>();
5         for(Integer i=0;i<50;i++){
6             testAccounts.add(new Account(Name
7 = 'Account'+i,BillingState='CA'));
8         }
9         for(Integer j=0;j<50;j++){
10            testAccounts.add(new Account(Name
11 = 'Account'+j,BillingState = 'NY'));
12        }
13
14        insert testAccounts;
15
16        Contact testContact = new Contact(FirstName =
17 'John',LastName='Doe');
18        insert testContact;
19
20        AddPrimaryContact addit = new
21        addPrimaryContact(testContact,'CA');
22
23        Test.startTest();
24        System.enqueueJob(addit);
25        Test.stopTest();
26        System.assertEquals(50,[Select count() from
27 Contact where accountId in(Select Id from Account where
28 BillingState = 'CA')]);
29 }

```

```
23
24
25     }
26 }
```

4. Schedule Jobs Using the Apex Scheduler -

Apex class : DailyLeadProcessor

```
1 public without sharing class DailyLeadProcessor implements
  Schedulable{
2     public void execute (SchedulableContext ctx){
3         List <Lead> leads =[SELECT Id, LeadSource FROM Lead
  WHERE LeadSource = null LIMIT 200];
4         for (Lead l:leads){
5             l.LeadSource = 'Dreamforce';
6         }
7         update leads;
8     }
9 }
```

Apex test class : DailyLeadProcessorTest

```
1 @isTest
2 private class DailyLeadProcessorTest{
3     //Seconds Minutes Hours Day_of_month Month
  Day_of_week optional_year
4     public static String CRON_EXP = '0 0 0 2 6 ? 2022';
5
6     static testmethod void testScheduledJob(){
7         List<Lead> leads = new List<Lead>();
8
9         for(Integer i = 0; i < 200; i++){
10             Lead lead = new Lead(LastName = 'Test ' + i,
  LeadSource = '', Company = 'Test Company ' + i, Status =
  'Open - Not Contacted');
11             leads.add(lead);
12         }
13 }
```

```

14         insert leads;
15
16         Test.startTest();
17         // Schedule the test job
18         String jobId = System.schedule('Update LeadSource
19
20         // Stopping the test will run the job
21         Test.stopTest();
22     }
23 }

```

Apex Integration Services -

1. Apex REST Callouts

Create an Apex class that calls a REST endpoint to return the name of an animal.

Apex class: AnimalLocator

```

1 public class AnimalLocator{
2     public static String getAnimalNameById(Integer x){
3         Http http = new Http();
4         HttpRequest req = new HttpRequest();
5
6         req.setEndpoint('https://th-apex-http-
7
8         req.setMethod('GET');
9
10        HttpResponse res = http.send(req);
11        Map<String, Object> results = (Map<String,
12        Object>)JSON.deserializeUntyped(res.getBody());
13        Map<String, Object> animal = (Map<String,
14        Object>) results.get('animal');
15        return (String)animal.get('name');
16    }
17 }

```

```
15 }
```

Test Class : AnimalLocatorTest

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

Mock Test class: AnimalLocatorMock

```
1 @isTest
2 global class AnimalLocatorMock implements HttpCalloutMock
3 {
4     global HTTPResponse respond(HTTPRequest request) {
5         HTTPResponse response = new HTTPResponse();
6         response.setHeader('Content-Type',
            'application/json');
7         response.setBody('{"animal":{"id":1,"name":"chicken","eat
8
9         response.setStatusCode(200);
10        return response;
11    }
12 }
```

2. Apex SOAP Callouts Units :

Generate an Apex class using WSDL2Apex for a SOAP web service, write unit tests

that achieve 100 percent code coverage for the class using a mock response, and run your Apex tests.

Apex class : ParkService

```
1 //Generated by wsdl2apex
2
3 public class ParkService {
4     public class byCountryResponse {
5         public String[] return_x;
6         private String[] return_x_type_info = new
String[]{'return','http://parks.services/',null,'0','-
7         private String[] apex_schema_type_info = new
String[]{'http://parks.services/','false','false'};
8         private String[] field_order_type_info = new
String[]{'return_x'};
9     }
10    public class byCountry {
11        public String arg0;
12        private String[] arg0_type_info = new
String[]{'arg0','http://parks.services/',null,'0','1','fa
13        private String[] apex_schema_type_info = new
String[]{'http://parks.services/','false','false'};
14        private String[] field_order_type_info = new
String[]{'arg0'};
15    }
16    public class ParksImplPort {
17        public String endpoint_x = 'https://th-apex-soap-
18        public Map<String,String> inputHttpHeaders_x;
19        public Map<String,String> outputHttpHeaders_x;
20        public String clientCertName_x;
21        public String clientCert_x;
22        public String clientCertPasswd_x;
23        public Integer timeout_x;
24        private String[] ns_map_type_info = new
String[]{'http://parks.services/', 'ParkService'};
25        public String[] byCountry(String arg0) {
```

```

26         ParkService.byCountry request_x = new
    ParkService.byCountry();
27         request_x.arg0 = arg0;
28         ParkService.byCountryResponse response_x;
29         Map<String, ParkService.byCountryResponse>
    response_map_x = new Map<String,
    ParkService.byCountryResponse>();
30         response_map_x.put('response_x', response_x);
31         WebServiceCallout.invoke(
32             this,
33             request_x,
34             response_map_x,
35             new String[]{endpoint_x,
36                 '',
37                 'http://parks.services/',
38                 'byCountry',
39                 'http://parks.services/',
40                 'byCountryResponse',
41                 'ParkService.byCountryResponse'}
42         );
43         response_x =
    response_map_x.get('response_x');
44         return response_x.return_x;
45     }
46 }
47}

```

Apex class : ParkLocator

```

1 public class ParkLocator {
2     public static String[] country(String country){
3         ParkService.ParksImplPort parks = new
    ParkService.ParksImplPort();
4         String[] parksname = parks.byCountry(country);
5         return parksname;
6     }

```

```
7 }
```

Apex Text Class : ParkLocatorTest

```
1 @isTest
2 private class ParkLocatorTest {
3     @isTest static void testCallout() {
4         // This causes a fake response to be generated
5         Test.setMock(WebServiceMock.class, new
        ParkServiceMock());
6         // Call the method that invokes a callout
7         List<String> result = new List<String>();
8         List<String> expectedvalue = new
        List<String>{'Park1','Park2','Park3'};
9
10        result = ParkLocator.country('India');
11        // Verify that a fake result is returned
12        System.assertEquals(expectedvalue, result);
13    }
14 }
```

Apex Mock Test class : ParkServiceMock

```
1 @isTest
2 global class ParkServiceMock implements WebServiceMock {
3     global void doInvoke(
4         Object stub,
5         Object request,
6         Map<String, Object> response,
7         String endpoint,
8         String soapAction,
9         String requestName,
10        String responseNS,
11        String responseName,
12        String responseType) {
13        // start - specify the response you want to send
14        ParkService.byCountryResponse response_x =
15            new ParkService.byCountryResponse();
```



```

16
17         List<String> myStrings = new List<String>
18         {'Park1','Park2','Park3'};
19
19         response_x.return_x = myStrings;
20         // end
21         response.put('response_x', response_x);
22     }
23 }

```

3.Apex Web Services -

Create an Apex REST class that is accessible at /Accounts/<Account_ID>/contacts. The service will return the account's ID and name plus the ID and name of all contacts associated with the account.

Apex class : AccountManager

```

1  @RestResource(urlMapping = '/Accounts/*/contacts')
2  global with sharing class AccountManager{
3
4      @HttpGet
5      global static Account getAccount(){
6          RestRequest request= restContext.request;
7          string accountId
8          =request.requestURI.substringBetween('/Accounts/', '/contac
9
10         Account result = [SELECT Id ,Name,(Select Id ,
11         Name from Contacts) from Account where Id = :accountId
12         Limit 1];
13         return result;
14     }
15 }

```

Apex Unit Test Class : AccountManagerTest

```

1  @isTest

```

```

2 private class AccountManagerTest{
3     static testMethod void testMethod1(){
4         Account objAccount = new Account(Name = 'test
5
6         insert objAccount;
7         Contact objContact = new Contact(LastName = 'test
8
9         AccountId =
10        objAccount.Id);
11        insert objContact;
12        Id recordId = objAccount.Id;
13        RestRequest request = new RestRequest();
14        request.requestUri =
15        'https://sandeepidentity-dev-
16        ed.my.salesforce.com/services/apexrest/Accounts/'
17        + recordId + '/contacts';
18        request.httpMethod = 'GET';
19        RestContext.request = request;
20        // Call the method to test
21        Account thisAccount =
22        AccountManager.getAccount();
23        // Verify results
24        System.assert(thisAccount != null);
25        System.assertEquals('test Account',
26        thisAccount.Name);
27    }
28 }

```

Apex specialist Challenges

step 1 - Quiz

Step 2 - Automate Record Creation

Apex Class : MaintenanceRequestHelper

```

1 public with sharing class MaintenanceRequestHelper {
2     public static void updateWorkOrders(List<Case>
    updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
3         Set<Id> validIds = new Set<Id>();
4         For (Case c : updWorkOrders){
5             if (nonUpdCaseMap.get(c.Id).Status != 'Closed' &&
    c.Status == 'Closed'){
6                 if (c.Type == 'Repair' || c.Type == 'Routine
7
                        validIds.add(c.Id);
8
                    }
9
        }
10    }
11
12    //When an existing maintenance request of type Repair
    or Routine Maintenance is closed,
13    //create a new maintenance request for a future
    routine checkup.
14    if (!validIds.isEmpty()){
15        Map<Id,Case> closedCases = new
    Map<Id,Case>([SELECT Id, Vehicle__c, Equipment__c,
    Equipment__r.Maintenance_Cycle__c,
16
        (SELECT Id,Equipment__c,Quantity__c FROM
    Equipment_Maintenance_Items__r)
17
                                FROM
    Case WHERE Id IN :validIds]);
18        Map<Id,Decimal> maintenanceCycles = new
    Map<ID,Decimal>();
19
20    //calculate the maintenance request due dates by
    using the maintenancecycle defined on the related equipment
    records.
21        AggregateResult[] results = [SELECT
    Maintenance_Request__c,
22
        MIN(Equipment__r.Maintenance_Cycle__c)cycle
23
                                FROM
    Equipment_Maintenance_Item__c
24
                                WHERE
    Maintenance_Request__c IN :ValidIds GROUP BY
    Maintenance_Request__c];
25

```

```

26         for (AggregateResult ar : results){
27             maintenanceCycles.put((Id)
ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
28         }
29
30         List<Case> newCases = new List<Case>();
31         for(Case cc : closedCases.values()){
32             Case nc = new Case (
33                 ParentId = cc.Id,
34                 Status = 'New',
35                 Subject = 'Routine Maintenance',
36                 Type = 'Routine Maintenance',
37                 Vehicle__c = cc.Vehicle__c,
38                 Equipment__c =cc.Equipment__c,
39                 Origin = 'Web',
40                 Date_Reported__c = Date.Today()
41             );
42
43             //If multiple pieces of equipment are used in
the maintenance request,
44             //define the due date by applying the shortest
maintenance cycle to today's date.
45             //If (maintenanceCycles.containsKey(cc.Id)){
46                 nc.Date_Due__c =
Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
47             //} else {
48                 //      nc.Date_Due__c =
Date.today().addDays((Integer)
cc.Equipment__r.maintenance_Cycle__c);
49             //}
50
51             newCases.add(nc);
52         }
53
54         insert newCases;
55
56         List<Equipment_Maintenance_Item__c> clonedList =
new List<Equipment_Maintenance_Item__c>();
57         for (Case nc : newCases){
58             for (Equipment_Maintenance_Item__c
clonedListItem :
closedCases.get(nc.ParentId).Equipment_Maintenance_Items__r){
59                 Equipment_Maintenance_Item__c item =

```

```

        clonedListItem.clone();
60         item.Maintenance_Request__c = nc.Id;
61         clonedList.add(item);
62     }
63 }
64     insert clonedList;
65 }
66 }
67 }

```

Apex Trigger: MaintenanceRequest

```

1  trigger MaintenanceRequest on Case (before update, after
   update) {
2      if(Trigger.isUpdate && Trigger.isAfter){
3          MaintenanceRequestHelper.updateWorkOrders(Trigger.New,
   Trigger.OldMap);
4      }
5  }

```

Step3 - Synchronize Salesforce data with an external system

Apex Calss : WarehouseCalloutService

```

1  public with sharing class WarehouseCalloutService {
2
3      private static final String WAREHOUSE_URL =
   'https://th-superbadge-apex.herokuapp.com/equipment';
4
5      // complete this method to make the callout (using
   @future) to the
6      // REST endpoint and update equipment on hand.
7      @future(callout=true)
8      public static void runWarehouseEquipmentSync(){
9          Http http = new Http();
10         HttpRequest request = new HttpRequest();
11         request.setEndpoint(WAREHOUSE_URL);
12         request.setMethod('GET');
13         HttpResponse response = http.send(request);
14         // If the request is successful, parse the JSON response.
15         if (response.getStatusCode() == 200) {

```

```

16         // Deserialize the JSON string into collections of primitive data types.
17         List<Object> equipments = (List<Object>)
JSON.deserializeUntyped(response.getBody());
18         List<Product2> products = new
List<Product2>();
19         for(Object o : equipments){
20             Map<String, Object> mapProduct =
(Map<String, Object>)o;
21             Product2 product = new Product2();
22             product.Name =
(String)mapProduct.get('name');
23             product.Cost__c =
(Integer)mapProduct.get('cost');
24             product.Current_Inventory__c =
(Integer)mapProduct.get('quantity');
25             product.Maintenance_Cycle__c =
(Integer)mapProduct.get('maintenanceperiod');
26             product.Replacement_Part__c =
(Boolean)mapProduct.get('replacement');
27             product.Lifespan_Months__c =
(Integer)mapProduct.get('lifespan');
28             product.Warehouse_SKU__c =
(String)mapProduct.get('sku');
29             product.ProductCode =
(String)mapProduct.get('_id');
30             products.add(product);
31         }
32         if(products.size() > 0){
33             System.debug(products);
34             upsert products;
35         }
36     }
37 }
38 }

```

Step 4 - Schedule Synchronize

Apex code : WarehouseSyncSchedule

```
1 global with sharing class WarehouseSyncSchedule
  implements Schedulable{
2     global void execute(SchedulableContext ctx){
3         System.enqueueJob(new WarehouseCalloutService());
4     }
5 }
```

Step 5 - Test automation logic

Apex Trigger: MaintenanceRequest

```
1 trigger MaintenanceRequest on Case (before update, after
  update) {
2     if(Trigger.isUpdate && Trigger.isAfter){
3         MaintenanceRequestHelper.updateWorkOrders(Trigger.New,
4         Trigger.OldMap);
5     }
6 }
```

Apex Class : MaintenanceRequestHelper

```
1 public with sharing class MaintenanceRequestHelper {
2     public static void updateWorkOrders(List<Case>
3     updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
4         Set<Id> validIds = new Set<Id>();
5         For (Case c : updWorkOrders){
6             if (nonUpdCaseMap.get(c.Id).Status != 'Closed' &&
7             c.Status == 'Closed'){
8                 if (c.Type == 'Repair' || c.Type == 'Routine
9
10                    validIds.add(c.Id);
11                }
12            }
13        }
14
15        //When an existing maintenance request of type Repair
16        //or Routine Maintenance is closed,
17        //create a new maintenance request for a future
```

```

    routine checkup.
14     if (!validIds.isEmpty()){
15         Map<Id,Case> closedCases = new
Map<Id,Case>([SELECT Id, Vehicle__c, Equipment__c,
Equipment__r.Maintenance_Cycle__c,
16         (SELECT Id,Equipment__c,Quantity__c FROM
Equipment_Maintenance_Items__r)
17         Case WHERE Id IN :validIds]);
18         Map<Id,Decimal> maintenanceCycles = new
Map<ID,Decimal>();
19
20         //calculate the maintenance request due dates by
using the maintenance cycle defined on the related equipment
records.
21         AggregateResult[] results = [SELECT
Maintenance_Request__c,
22         MIN(Equipment__r.Maintenance_Cycle__c)cycle
23         FROM
Equipment_Maintenance_Item__c
24         WHERE
Maintenance_Request__c IN :ValidIds GROUP BY
Maintenance_Request__c];
25
26         for (AggregateResult ar : results){
27             maintenanceCycles.put((Id)
ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
28         }
29
30         List<Case> newCases = new List<Case>();
31         for(Case cc : closedCases.values()){
32             Case nc = new Case (
33                 ParentId = cc.Id,
34                 Status = 'New',
35                 Subject = 'Routine Maintenance',
36                 Type = 'Routine Maintenance',
37                 Vehicle__c = cc.Vehicle__c,
38                 Equipment__c =cc.Equipment__c,
39                 Origin = 'Web',
40                 Date_Reported__c = Date.Today()
41             );

```



```

42
43             //If multiple pieces of equipment are used in
the maintenance request,
44             //define the due date by applying the shortest
maintenance cycle to today's date.
45             //If (maintenanceCycles.containsKey(cc.Id)){
46                 nc.Date_Due__c =
Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
47             //} else {
48             //    nc.Date_Due__c =
Date.today().addDays((Integer)
cc.Equipment__r.maintenance_Cycle__c);
49             //}
50
51             newCases.add(nc);
52         }
53
54         insert newCases;
55
56         List<Equipment_Maintenance_Item__c> clonedList =
new List<Equipment_Maintenance_Item__c>();
57         for (Case nc : newCases){
58             for (Equipment_Maintenance_Item__c
clonedListItem :
closedCases.get(nc.ParentId).Equipment_Maintenance_Items__r){
59                 Equipment_Maintenance_Item__c item =
clonedListItem.clone();
60                 item.Maintenance_Request__c = nc.Id;
61                 clonedList.add(item);
62             }
63         }
64         insert clonedList;
65     }
66 }
}

```

Apex Test Class : MaintenanceRequestHelperTest

```

1  @isTest
2  public with sharing class MaintenanceRequestHelperTest {
3
4      // createVehicle

```

```

5     private static Vehicle__c createVehicle(){
6         Vehicle__c vehicle = new Vehicle__C(name = 'Testing

7         return vehicle;
8     }
9
10    // createEquipment
11    private static Product2 createEquipment(){
12        product2 equipment = new product2(name = 'Testing

13                                                lifespan_months__c =
14        10,
15                                                maintenance_cycle__c
16        = 10,
17                                                replacement_part__c
18        = true);
19        return equipment;
20    }
21
22    // createMaintenanceRequest
23    private static Case createMaintenanceRequest(id vehicleId,
24    id equipmentId){
25        case cse = new case(Type='Repair',
26                            Status='New',
27                            Origin='Web',
28                            Subject='Testing subject',
29                            Equipment__c=equipmentId,
30                            Vehicle__c=vehicleId);
31        return cse;
32    }
33
34    // createEquipmentMaintenanceItem
35    private static Equipment_Maintenance_Item__c
36    createEquipmentMaintenanceItem(id equipmentId,id requestId){
37        Equipment_Maintenance_Item__c equipmentMaintenanceItem
38        = new Equipment_Maintenance_Item__c(
39            Equipment__c = equipmentId,
40            Maintenance_Request__c = requestId);
41        return equipmentMaintenanceItem;
42    }
43
44    @isTest
45    private static void testPositive(){

```

```

40     Vehicle__c vehicle = createVehicle();
41     insert vehicle;
42     id vehicleId = vehicle.Id;
43
44     Product2 equipment = createEquipment();
45     insert equipment;
46     id equipmentId = equipment.Id;
47
48     case createdCase =
createMaintenanceRequest(vehicleId,equipmentId);
49     insert createdCase;
50
51     Equipment_Maintenance_Item__c equipmentMaintenanceItem
= createEquipmentMaintenanceItem(equipmentId,createdCase.id);
52     insert equipmentMaintenanceItem;
53
54     test.startTest();
55     createdCase.status = 'Closed';
56     update createdCase;
57     test.stopTest();
58
59     Case newCase = [Select id,
60                     subject,
61                     type,
62                     Equipment__c,
63                     Date_Reported__c,
64                     Vehicle__c,
65                     Date_Due__c
66                     from case
67                     where status ='New'];
68
69     Equipment_Maintenance_Item__c workPart = [select id
70                                               from
Equipment_Maintenance_Item__c
71                                               where
Maintenance_Request__c =:newCase.Id];
72     list<case> allCase = [select id from case];
73     system.assert(allCase.size() == 2);
74
75     system.assert(newCase != null);
76     system.assert(newCase.Subject != null);
77     system.assertEquals(newCase.Type, 'Routine

```

```

78         SYSTEM.assertEquals(newCase.Equipment__c,
equipmentId);
79         SYSTEM.assertEquals(newCase.Vehicle__c, vehicleId);
80         SYSTEM.assertEquals(newCase.Date_Reported__c,
system.today());
81     }
82
83     @isTest
84     private static void testNegative(){
85         Vehicle__C vehicle = createVehicle();
86         insert vehicle;
87         id vehicleId = vehicle.Id;
88
89         product2 equipment = createEquipment();
90         insert equipment;
91         id equipmentId = equipment.Id;
92
93         case createdCase =
createMaintenanceRequest(vehicleId,equipmentId);
94         insert createdCase;
95
96         Equipment_Maintenance_Item__c workP =
createEquipmentMaintenanceItem(equipmentId, createdCase.Id);
97         insert workP;
98
99         test.startTest();
100         createdCase.Status = 'Working';
101         update createdCase;
102         test.stopTest();
103
104         list<case> allCase = [select id from case];
105
106         Equipment_Maintenance_Item__c
equipmentMaintenanceItem = [select id
107                                     from
Equipment_Maintenance_Item__c
108                                     where
Maintenance_Request__c = :createdCase.Id];
109
110         system.assert(equipmentMaintenanceItem != null);
111         system.assert(allCase.size() == 1);
112     }
113

```

```

114     @isTest
115     private static void testBulk(){
116         list<Vehicle__C> vehicleList = new
list<Vehicle__C>();
117         list<Product2> equipmentList = new list<Product2>();
118         list<Equipment_Maintenance_Item__c>
equipmentMaintenanceItemList = new
list<Equipment_Maintenance_Item__c>();
119         list<case> caseList = new list<case>();
120         list<id> oldCaseIds = new list<id>();
121
122         for(integer i = 0; i < 300; i++){
123             vehicleList.add(createVehicle());
124             equipmentList.add(createEquipment());
125         }
126         insert vehicleList;
127         insert equipmentList;
128
129         for(integer i = 0; i < 300; i++){
130
131             caseList.add(createMaintenanceRequest(vehicleList.get(i).id,
equipmentList.get(i).id));
132             insert caseList;
133
134             for(integer i = 0; i < 300; i++){
135
136                 equipmentMaintenanceItemList.add(createEquipmentMaintenanceIte
137
138             }
139             insert equipmentMaintenanceItemList;
140
141             test.startTest();
142             for(case cs : caseList){
143                 cs.Status = 'Closed';
144                 oldCaseIds.add(cs.Id);
145             }
146             update caseList;
147             test.stopTest();
148
149             list<case> newCase = [select id
from case
where status = 'New'];

```

```

150
151
152
153         list<Equipment_Maintenance_Item__c> workParts =
            [select id
154
                                                    from
            Equipment_Maintenance_Item__c
155
            where Maintenance_Request__c in: oldCaseIds];
156
157         system.assert(newCase.size() == 300);
158
159         list<case> allCase = [select id from case];
160         system.assert(allCase.size() == 600);
161     }
162 }

```

Step 6 - Schedule Synchronize

Apex Calss : WarehouseCalloutService

```

1  public with sharing class WarehouseCalloutService {
2
3      private static final String WAREHOUSE_URL =
        'https://th-superbadge-apex.herokuapp.com/equipment';
4
5      // complete this method to make the callout (using
        @future) to the
6      // REST endpoint and update equipment on hand.
7      @future(callout=true)
8      public static void runWarehouseEquipmentSync(){
9          Http http = new Http();
10         HttpRequest request = new HttpRequest();
11         request.setEndpoint(WAREHOUSE_URL);
12         request.setMethod('GET');
13         HttpResponse response = http.send(request);
14         // If the request is successful, parse the
            JSON response.
15         if (response.getStatusCode() == 200) {

```

```

16         // Deserialize the JSON string into
    collections of primitive data types.
17         List<Object> equipments = (List<Object>)
    JSON.deserializeUntyped(response.getBody());
18
19         List<Product2> products = new
    List<Product2>();
20         for(Object o : equipments){
21             Map<String, Object> mapProduct =
    (Map<String, Object>)o;
22             Product2 product = new Product2();
23             product.Name =
    (String)mapProduct.get('name');
24             product.Cost__c =
    (Integer)mapProduct.get('cost');
25             product.Current_Inventory__c =
    (Integer)mapProduct.get('quantity');
26             product.Maintenance_Cycle__c =
    (Integer)mapProduct.get('maintenanceperiod');
27             product.Replacement_Part__c =
    (Boolean)mapProduct.get('replacement');
28             product.Lifespan_Months__c =
    (Integer)mapProduct.get('lifespan');
29             product.Warehouse_SKU__c =
    (String)mapProduct.get('sku');
30             product.ProductCode =
    (String)mapProduct.get('_id');
31             products.add(product);
32         }
33         if(products.size() > 0){
34             System.debug(products);
35             upsert products;
36         }
37     }
38 }
}

```

Apex Mock Class: WarehouseCalloutServiceMock

```
1 @isTest
2 global class WarehouseCalloutServiceMock implements
  HttpCalloutMock {
3     // implement http mock callout
4     global static HttpResponse respond(HttpRequest
  request) {
5
6         HttpResponse response = new HttpResponse();
7         response.setHeader('Content-Type',
  'application/json');
8         response.setBody(' [{"_id":"55d66226726b611100aaf741"},"rep
9
10        response.setStatusCode(200);
11
12        return response;
13    }
14 }
```

Apex Teast class : WarehouseCalloutServiceTest

```
1 @IsTest
2 private class WarehouseCalloutServiceTest {
3     // implement your mock callout test here
4     @isTest
5     static void testWarehouseCallout() {
6         test.startTest();
7         test.setMock(HttpCalloutMock.class, new
  WarehouseCalloutServiceMock());
8
9         WarehouseCalloutService.runWarehouseEquipmentSync();
10        test.stopTest();
11
12        List<Product2> product2List = new
  List<Product2>();
13        product2List = [SELECT ProductCode FROM
  Product2];
14    }
```



```

14         System.assertEquals(3, product2List.size());
15         System.assertEquals('55d66226726b611100aaf741',
    product2List.get(0).ProductCode);
16         System.assertEquals('55d66226726b611100aaf742',
    product2List.get(1).ProductCode);
17         System.assertEquals('55d66226726b611100aaf743',
    product2List.get(2).ProductCode);
18     }
19 }

```

Step 7 - Test Scheduling Logic

Apex Test Class : WarehouseSyncScheduleTest

```

1  @isTest
2  public with sharing class WarehouseSyncScheduleTest {
3      // implement scheduled code here
4      //
5      @isTest static void test() {
6          String scheduleTime = '00 00 00 * * ? *';
7          Test.startTest();
8          Test.setMock(HttpCalloutMock.class, new
    WarehouseCalloutServiceMock());
9          String jobId = System.schedule('Warehouse Time to
    ());
10         CronTrigger c = [SELECT State FROM CronTrigger WHERE
    Id =: jobId];
11         System.assertEquals('WAITING',
    String.valueOf(c.State), 'JobId does not match');
12
13         Test.stopTest();
14     }
15 }

```

Apex code : WarehouseSyncSchedule

```


1  global with sharing class WarehouseSyncSchedule
    implements Schedulable{
2      global void execute(SchedulableContext ctx){

```

```
3      System.enqueueJob(new WarehouseCalloutService());  
4  }
```

}

Process Automation Specialist :




Search

?

Rutuja Jarange
35 badges, 52,900 points

Today Learn Credentials Community For Companies



 App Builder Super Set

+10,000 POINTS

Superbadge





Process Automation Specialist

Showcase your mastery of business process automation without writing a line of code.



Completed 6/15/22

Prerequisites



Formulas and ValidationsSalesforce FlowLeads & Opportunities for Lightning ExperienceProcess Automation Specialist

Formula and Validation:

empathetic-fox-hlya25-dev-ed.lightning.force.com/lightning/setup/ObjectManager/Contract/FieldsAndRelationships/00N5i000004xnKF/view

Search Setup

Setup Home Object Manager

SETUP > OBJECT MANAGER

Contract

Contract Custom Field

Days Remaining

Back to Contract Fields

Custom Field Definition Detail

EditSet Field-Level SecurityView Field AccessibilityWhere is this used?

Field Information	
Field Label	Days Remaining
Field Name	Days_Remaining
API Name	Days_Remaining__c
Description	
Help Text	
Data Owner	
Field Usage	
Data Sensitivity Level	
Compliance Categorization	
Created By	Rutuja Jarange, 5/29/2022, 3:58 AM
Modified By	Rutuja Jarange, 5/29/2022, 3:58 AM

Formula Options	
Data Type	Formula
Decimal Places	2
Formula	EndDate - TODAY()

empathetic-fox-hlya25-dev-ed.lightning.force.com/lightning/setup/ObjectManager/Account/FieldsAndRelationships/00N5i000004xnLD/view

Search Setup

Setup Home Object Manager

SETUP > OBJECT MANAGER

Account

Account Custom Field
Potential Value
[Back to Account Fields](#)

Custom Field Definition Detail [Edit](#) [Set Field-Level Security](#) [View Field Accessibility](#) [Where is this used?](#)

Field Information

Field Label	Potential Value	Object Name	Account
Field Name	Potential_Value		
API Name	Potential_Value__c		
Description	To Calculate the total expected revenue of all the opportunities related to the account		
Help Text			
Data Owner			
Field Usage			
Data Sensitivity Level			
Compliance Categorization			
Created By	Rutuja Jarange , 5/29/2022, 4:07 AM	Modified By	Rutuja Jarange , 5/29/2022, 4:07 AM

Roll-Up Summary Options

Data Type	Roll-Up Summary	Summary Type	SUM
Summarized Object	Opportunity		
Field to Aggregate	Opportunity_Expected Revenue		
Filter Criteria			

Salesforce Flows:

Search Setup

Setup Home Object Manager

Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with Process Builder—and more! Salesforce plans to retire Process Builder and recommends building automation in Flow Builder. [Tell Me More](#) [Try Flow Builder](#)

Process Builder - Contact address change [Back To Setup](#) [Help](#)

[Expand All](#) [Collapse All](#) [View All Processes](#) [Clone](#) [View Properties](#) [Deactivate](#) [Read Only](#)

```

graph TD
    START([START]) --> Account[Account]
    Account --> AddressChange{Address Change}
    AddressChange -- TRUE --> IMMEDIATE_ACTIONS[IMMEDIATE ACTIONS]
    IMMEDIATE_ACTIONS --> STOP([STOP])
    IMMEDIATE_ACTIONS --> UpdateContactAdd[Update Contact Add...]
    IMMEDIATE_ACTIONS --> AddAction[+ Add Action]
    AddressChange -- FALSE --> IMMEDIATE_ACTIONS
  
```

Select Elements

↶ ↷ ⚙

Auto-Layout

Version 1: Active—Last modified 8 days ago

Run

Debug

Deactivate

Save As

Save

Toolbox

Manager

Search this flow...

New Resource

RESOURCES

Screen Components (3)

A Company_Name >

A Last_Name >

⚡ Upload_File >

Variables (1)

A LeadId from Create_Lead >

ELEMENTS

Create Records (1)

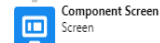
Create_Lead Q >

Screens (2)

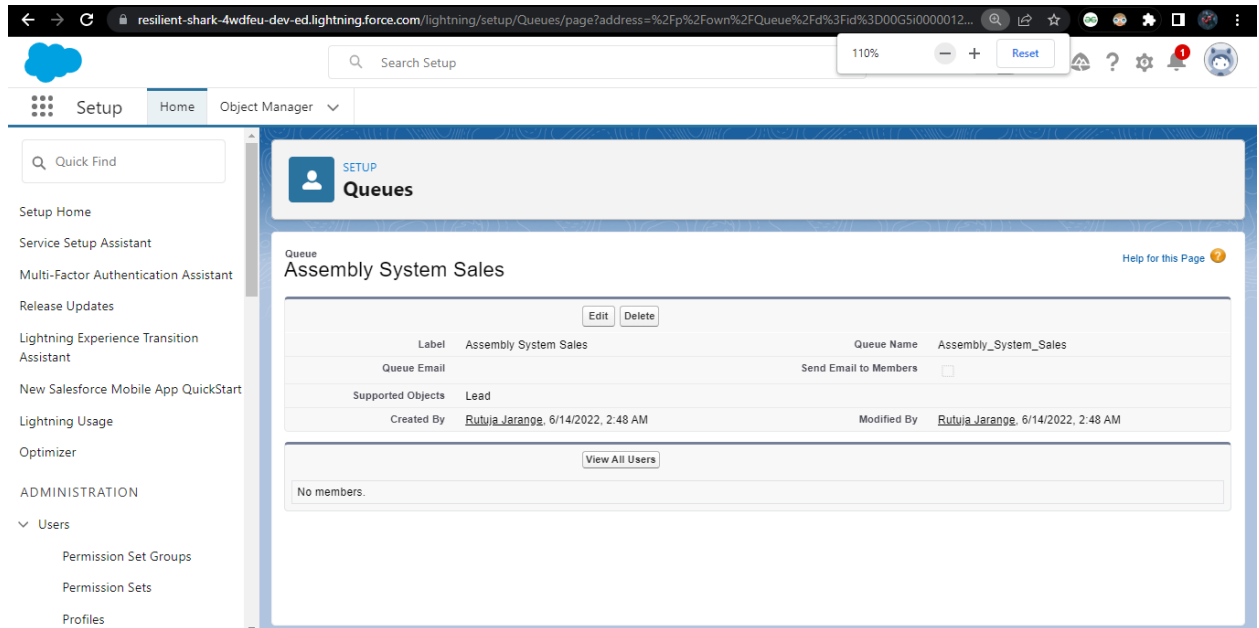
Component_Screen Q >

Lead_Screen Q >

− × +



Process Automation Specialist Challenges-



The screenshot shows the Salesforce Setup interface. The left sidebar contains navigation links: Setup Home, Service Setup Assistant, Multi-Factor Authentication Assistant, Release Updates, Lightning Experience Transition Assistant, New Salesforce Mobile App QuickStart, Lightning Usage, Optimizer, ADMINISTRATION, Users, Permission Set Groups, Permission Sets, and Profiles. The main content area is titled 'Queues' and shows a queue named 'Assembly System Sales'. The queue details include: Label: Assembly System Sales, Queue Name: Assembly_System_Sales, Queue Email: Send Email to Members, Supported Objects: Lead, Created By: Rutuja Jaranga, 6/14/2022, 2:48 AM, and Modified By: Rutuja Jaranga, 6/14/2022, 2:48 AM. There are buttons for 'Edit', 'Delete', and 'View All Users'. The queue currently has no members.

Setup Home

Service Setup Assistant

Multi-Factor Authentication Assistant

Release Updates

Lightning Experience Transition Assistant

New Salesforce Mobile App QuickStart

Lightning Usage

Optimizer

ADMINISTRATION

Users

Permission Set Groups

Permission Sets

Profiles

Queue

Assembly System Sales

Edit Delete

Label Assembly System Sales

Queue Name Assembly_System_Sales

Queue Email Send Email to Members

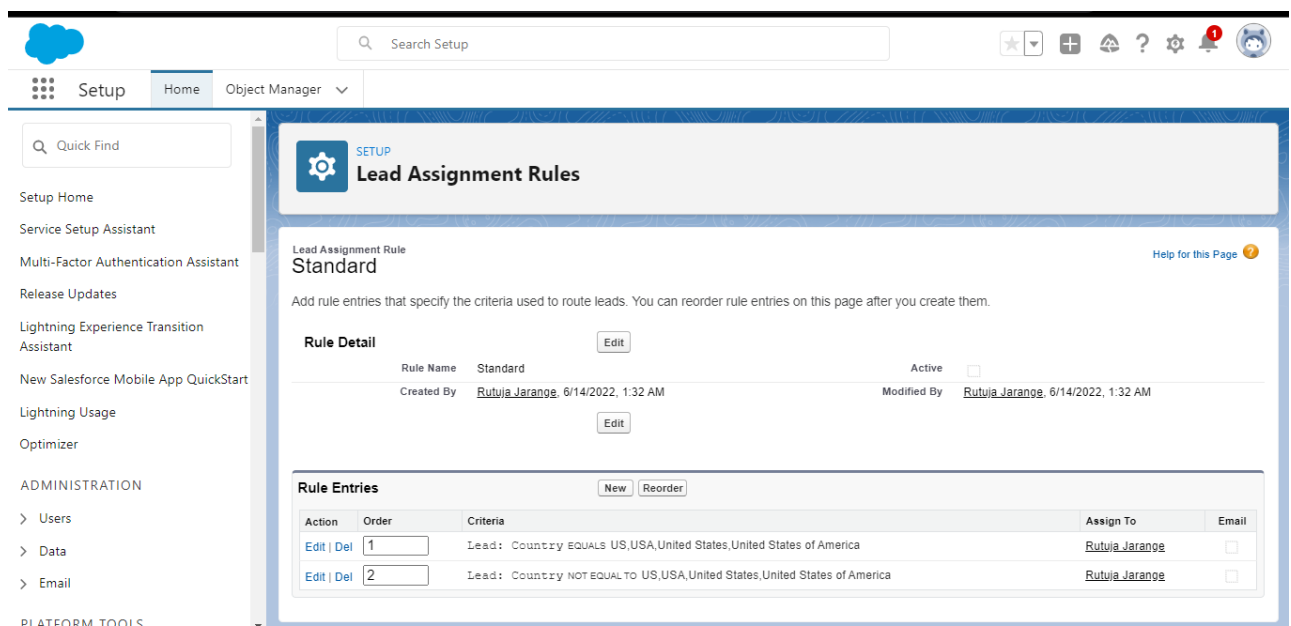
Supported Objects Lead

Created By Rutuja Jaranga, 6/14/2022, 2:48 AM

Modified By Rutuja Jaranga, 6/14/2022, 2:48 AM

View All Users

No members.



The screenshot shows the Salesforce Setup interface for Lead Assignment Rules. The left sidebar contains navigation links: Setup Home, Service Setup Assistant, Multi-Factor Authentication Assistant, Release Updates, Lightning Experience Transition Assistant, New Salesforce Mobile App QuickStart, Lightning Usage, Optimizer, ADMINISTRATION, Users, Data, Email, and PLATFORM TOOLS. The main content area is titled 'Lead Assignment Rules' and shows a rule named 'Standard'. The rule details include: Rule Name: Standard, Active: ☐, Created By: Rutuja Jaranga, 6/14/2022, 1:32 AM, and Modified By: Rutuja Jaranga, 6/14/2022, 1:32 AM. There are buttons for 'Edit' and 'New'. The rule entries table shows two entries:

Action	Order	Criteria	Assign To	Email
Edit Del	1	Lead: Country EQUALS US,USA,United States,United States of America	Rutuja Jaranga	<input type="checkbox"/>
Edit Del	2	Lead: Country NOT EQUAL TO US,USA,United States,United States of America	Rutuja Jaranga	<input type="checkbox"/>

Lightning Experience Transition Assistant

Move to the new, more productive Salesforce.

Get Started

Salesforce Mobile Quick Start

Home

Administer

Release Updates

Manage Users

Users

Mass Email Users

Roles

Permission Sets

Permission Set Groups

User Management Settings

Profiles

Public Groups

Queues

Login History

Identity Provider Event Log

Identity Verification History

Manage Apps

Manage Territories

Company Profile

Data Classification

User Detail

EditSharingReset PasswordFreeze

Name	Nushi Davoud	Role	
Alias	n dav	User License	Salesforce
Email	xxxxxxxx@xx.com	Profile	Contract Manager
Username	xxxxxxxx@xx.com	Active	<input checked="" type="checkbox"/>
Nickname	User16552078796508329349	Marketing User	<input type="checkbox"/>
Title	Nushi Davoud	Offline User	<input type="checkbox"/>
Company		Knowledge User	<input type="checkbox"/>
Department		Flow User	<input type="checkbox"/>
Division		Service Cloud User	<input type="checkbox"/>
Address		Site.com Contributor User	<input type="checkbox"/>
Time Zone	(GMT-07:00) Pacific Daylight Time (America/Los_Angeles)	Site.com Publisher User	<input type="checkbox"/>
Locale	English (United States)	WDC User	<input type="checkbox"/>
Language	English	Mobile Push Registrations	View
Delegated Approver		Data.com User Type	
Manager		Accessibility Mode (Classic Only)	<input type="checkbox"/>
Receive Approval Request Emails	Only if I am an approver	Debug Mode	<input type="checkbox"/>
Federation ID		High-Contrast Palette on Charts	<input type="checkbox"/>
App Registration: One-Time Password Authenticator		Load Lightning Pages While Scrolling	<input checked="" type="checkbox"/>
App Registration: Salesforce Authenticator		Salesforce CRM Content User	<input checked="" type="checkbox"/>
Security Key (U2F or WebAuthn)		Receive Salesforce CRM Content Email Alerts	<input checked="" type="checkbox"/>
Lightning Login		Receive Salesforce CRM Content Alerts as Daily Digest	<input checked="" type="checkbox"/>
Temporary Verification Code (Expires in 1 to 24 Hours)	[Generate]	Make Setup My Default Landing Page	<input type="checkbox"/>
		Allow Forecasting	<input type="checkbox"/>
		Call Center	
		Phone	
		Extension	

