Project Doc For Apex Modules

Apex Triggers

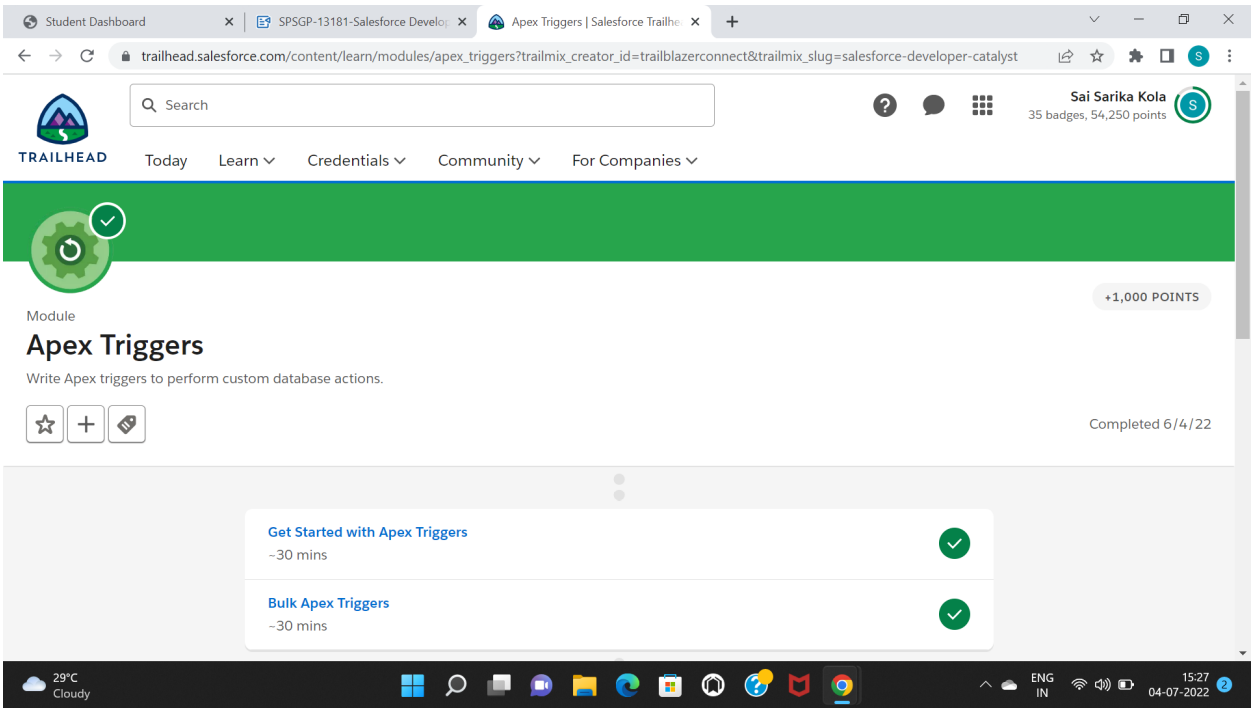
Get Started With Apex Triggers:

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

    for(Account account: Trigger.New){

        if(account.Match_Billing_Address__c == True){

            account.ShippingPostalCode=account.BillingPostalCode;

        }

    }

}
```

Bulk Apex Triggers:

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

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

    for(Opportunity opp : Trigger.new) {

    if(Trigger.isInsert) {
      if(Opp.StageName == 'Closed Won') {
        taskList.add(new Task(Subject = 'Follow Up Test Task', WhatId = opp.Id));
      }
    }

    if(Trigger.isUpdate) {
      if(Opp.StageName == 'Closed Won'
      && Opp.StageName != Trigger.oldMap.get(opp.Id).StageName) {
```

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

    if(taskList.size()>0) {
        insert taskList;
    }
}
```



Apex Testing

Get Started With Apex Units:

VerifyDate Class:

```
public class VerifyDate {

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

```
    return SetEndOfMonthDate(date1);
   }
  }
  private static Boolean DateWithin30Days(Date date1, Date date2) {

        if( date2 < date1) { return false; }

        Date date30Days = date1.addDays(30); //create a date 30 days away from date1
   if( date2 >= date30Days ) { return false; }
   else { return true; }
  }
  private static Date SetEndOfMonthDate(Date date1) {
   Integer totalDays = Date.daysInMonth(date1.year(), date1.month());
   Date lastDay = Date.newInstance(date1.year(), date1.month(), totalDays);
   return lastDay;
  }

}
```

TestVerifyDate:

```
@isTest
public class TestVerifyDate
{
   static testMethod void testMethod1()
   {
      Date d = VerifyDate.CheckDates(System.today(),System.today()+1);
      Date d1 = VerifyDate.CheckDates(System.today(),System.today()+60);
   }
}
```

Test Apex Triggers:

RestrictContactByName:

```
trigger RestrictContactByName on Contact (before insert, before update) {
```

```
  For (Contact c : Trigger.New) {
    if(c.LastName == 'INVALIDNAME') {  //invalidname is invalid
      c.AddError('The Last Name "'+c.LastName+'" is not allowed for DML');
    }

  }

}
```

TestRestrictContactByName:

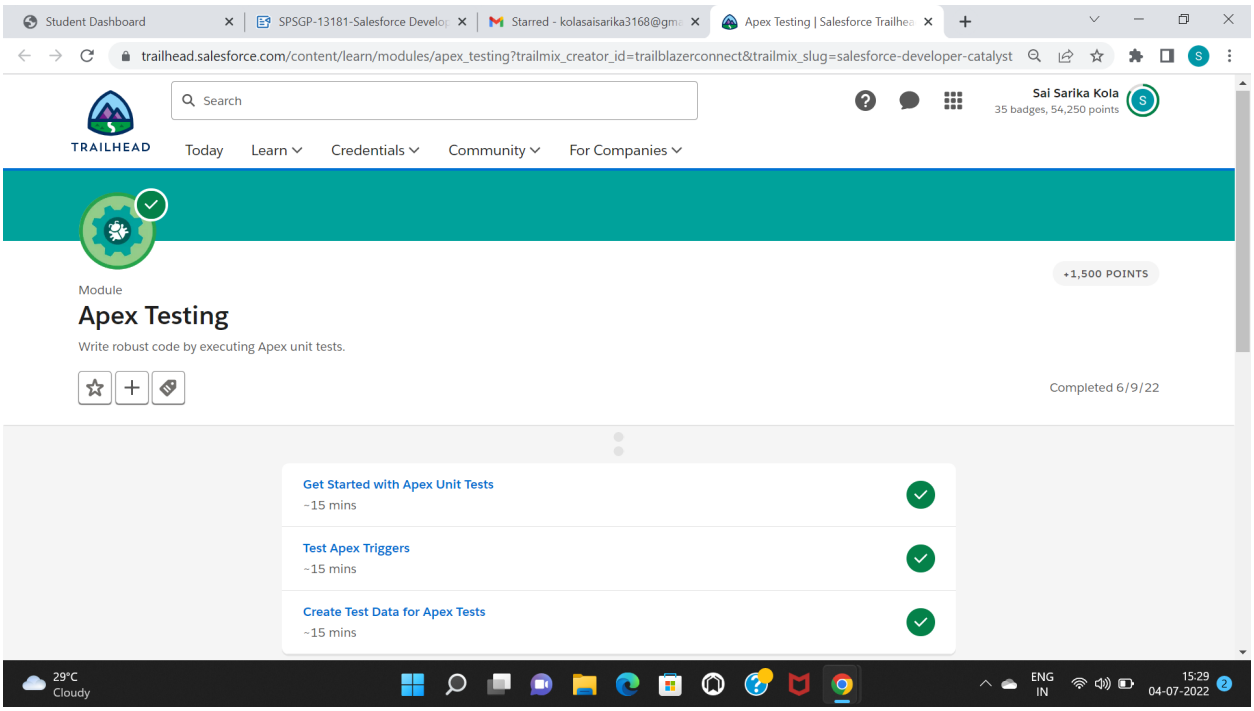
```
@isTest
private class TestRestrictContactByName {

    static testMethod void  metodoTest()
    {

        List<Contact> listContact= new List<Contact>();
        Contact c1 = new Contact(FirstName='Francesco', LastName='Riggio' ,
email='Test@test.com');
        Contact c2 = new Contact(FirstName='Francesco1', LastName =
'INVALIDNAME',email='Test@test.com');
        listContact.add(c1);
        listContact.add(c2);

        Test.startTest();
           try
           {
              insert listContact;
           }
           catch(Exception ee)
           {
           }

        Test.stopTest();

    }
```

```
}
```

Create Test Data For Apex Units:

RandomContactFactory Class:

```
@isTest
public class RandomContactFactory {
   public static List<Contact> generateRandomContacts(Integer
numContactsToGenerate, String FName) {
      List<Contact> contactList = new List<Contact>();

      for(Integer i=0;i<numContactsToGenerate;i++) {
         Contact c = new Contact(FirstName=FName + ' ' + i, LastName = 'Contact '+i);
         contactList.add(c);
         System.debug(c);
      }
      //insert contactList;
      System.debug(contactList.size());
      return contactList;
   }

}
```



Asynchronous Apex

Use Future Methods:

AccountProceesor:

```
public class AccountProcessor {
   @future
   public static void countContacts(List<Id> accountIds){
      List<Account> accounts = [Select Id, Name from Account Where Id IN : accountIds];
      List<Account> updatedAccounts = new List<Account>();
      for(Account account : accounts){
        account.Number_of_Contacts__c = [Select count() from Contact Where AccountId
=: account.Id];
         System.debug('No Of Contacts = ' + account.Number_of_Contacts__c);
         updatedAccounts.add(account);
      }
      update updatedAccounts;
   }
```

```
}
```

AccountProceesorTest:

```
@isTest
public class AccountProcessorTest {
   @isTest
   public static void testNoOfContacts(){
      Account a = new Account();
      a.Name
 = 'Test Account';
      Insert a;

      Contact c = new Contact();
      c.FirstName = 'Bob';
      c.LastName =  'Willie';
      c.AccountId = a.Id;

      Contact c2 = new Contact();
      c2.FirstName = 'Tom';
      c2.LastName = 'Cruise';
      c2.AccountId = a.Id;

      List<Id> acctIds = new List<Id>();
      acctIds.add(a.Id);

      Test.startTest();
      AccountProcessor.countContacts(acctIds);
      Test.stopTest();
   }

}
```

Use Batch Apex:

LeadProcessor:

```
public class LeadProcessor implements Database.Batchable<sObject> {

    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator([Select LeadSource From Lead ]);
    }
    public void execute(Database.BatchableContext bc, List<Lead> leads){
          for (Lead Lead : leads) {
             lead.LeadSource = 'Dreamforce';
          }
        update leads;
    }
    public void finish(Database.BatchableContext bc){
     }

}
```

LeadProcessorTest:

```
@isTest
public class LeadProcessorTest {

       @testSetup
    static void setup() {
       List<Lead> leads = new List<Lead>();
       for(Integer counter=0 ;counter <200;counter++){
          Lead lead = new Lead();
          lead.FirstName ='FirstName';
          lead.LastName ='LastName'+counter;
          lead.Company
 ='demo'+counter;
          leads.add(lead);
       }
       insert leads;
    }

    @isTest static void test() {
       Test.startTest();
```

```
        LeadProcessor leadProcessor = new LeadProcessor();
        Id batchId = Database.executeBatch(leadProcessor);
        Test.stopTest();
    }

}
```

Control Processes With Queueable Apex:

AddPrimaryContact:

```
public class AddPrimaryContact implements Queueable
{
    private Contact c;
    private String state;
    public  AddPrimaryContact(Contact c, String state)
    {
        this.c = c;
        this.state = state;
    }
    public void execute(QueueableContext context)
    {
        List<Account> ListAccount = [SELECT ID, Name ,(Select id,FirstName,LastName
from contacts ) FROM ACCOUNT WHERE BillingState = :state LIMIT 200];
        List<Contact> lstContact = new List<Contact>();
        for (Account acc:ListAccount)
        {
            Contact cont = c.clone(false,false,false,false);
            cont.AccountId =  acc.id;
            lstContact.add( cont );
        }

        if(lstContact.size() >0 )
        {
            insert lstContact;
        }
```

```
    }

}
```

AddPrimaryContactTest:

```
@isTest
public class AddPrimaryContactTest
{
    @isTest static void TestList()
    {
        List<Account> Teste = new List <Account>();
        for(Integer i=0;i<50;i++)
        {
            Teste.add(new Account(BillingState = 'CA', name = 'Test'+i));
        }
        for(Integer j=0;j<50;j++)
        {
            Teste.add(new Account(BillingState = 'NY', name = 'Test'+j));
        }
        insert Teste;

        Contact co = new Contact();
        co.FirstName='demo';
        co.LastName ='demo';
        insert co;
        String state = 'CA';

         AddPrimaryContact apc = new AddPrimaryContact(co, state);
         Test.startTest();
           System.enqueueJob(apc);
         Test.stopTest();
    }
}
```

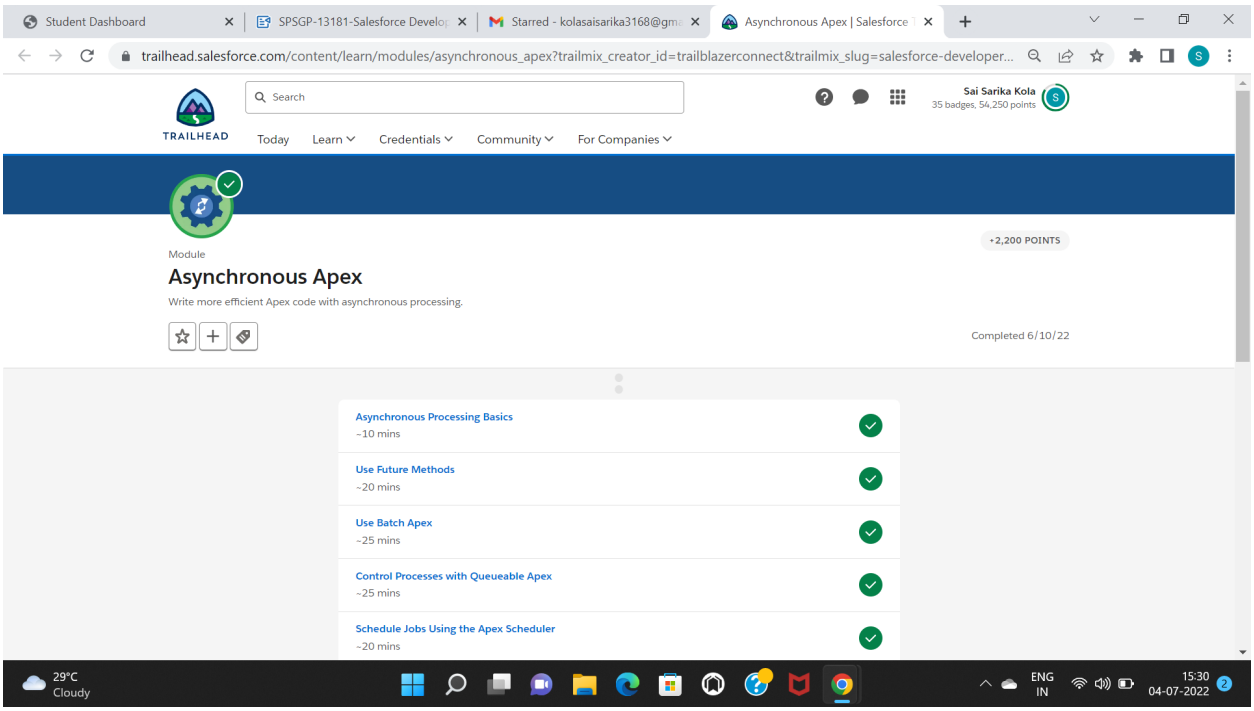
Schedule Jobs Using The Apex Scheduler:

DailyLeadProceesor:

```
public class DailyLeadProcessor implements Schedulable  {
   Public void execute(SchedulableContext SC){
     List<Lead> LeadObj=[SELECT Id from Lead where LeadSource=null limit 200];
      for(Lead l:LeadObj){
         l.LeadSource='Dreamforce';
         update l;
      }
   }
}
```

DailyLeadProcessorTest:

```
@isTest
private class DailyLeadProcessorTest {
          static testMethod void testDailyLeadProcessor() {
                    String CRON_EXP = '0 0 1 * * ?';
                    List<Lead> lList = new List<Lead>();
             for (Integer i = 0; i < 200; i++) {
                              lList.add(new Lead(LastName='Dreamforce'+i, Company='Test1 Inc.',
Status='Open - Not Contacted'));
                    }
                    insert lList;

                    Test.startTest();
                    String jobId = System.schedule('DailyLeadProcessor', CRON_EXP, new
DailyLeadProcessor());
          }
}
```


Student Dashboard
SPSGP-13181-Salesforce Develop
Starred - kolasaisarika3168@gm
Asynchronous Apex | Salesforce
trailhead.salesforce.com/content/learn/modules/asynchronous_apex?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer...
TRAILHEAD
Search
Today
Learn
Credentials
Community
For Companies
Sai Sarika Kola
35 badges, 54,250 points
Module
Asynchronous Apex
Write more efficient Apex code with asynchronous processing.
+2,200 POINTS
Completed 6/10/22
Asynchronous Processing Basics
~10 mins
Use Future Methods
~20 mins
Use Batch Apex
~25 mins
Control Processes with Queueable Apex
~25 mins
Schedule Jobs Using the Apex Scheduler
~20 mins
29°C
Cloudy
ENG
IN
15:30
04-07-2022