Apex Triggers

1. Get Started with Apex Triggers from (module - Apex Triggers)
2.
3. ---
4. Create an Apex trigger:
5. Name: AccountAddressTrigger
6. Object: Account
7.
8. SOURCE CODE:
9. trigger AccountAddressTrigger on Account (before insert, before update) {
10. for(Account a: Trigger.New){
11. if(a.Match_Billing_Address__c == true && a.BillingPostalCode!= null){
12. a.ShippingPostalCode=a.BillingPostalCode;
13. }
14. }
15. }

2)

1. Bulk Apex Triggers from (module - Apex Triggers)
2.
3.
4. ---
5. Create an Apex trigger:
6. Name: ClosedOpportunityTrigger
7. Object: Opportunity
8.

9. SOURCE CODE:
10. trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {
11. List<Task> taskList = new List<Task>();
12. for(Opportunity opp : [SELECT Id, StageName FROM Opportunity WHERE StageName='Closed Won' AND Id IN : Trigger.New]){
13. taskList.add(new Task(Subject='Follow Up Test Task', WhatId = opp.Id));
14. }
15. if(taskList.size()>0){
16. insert tasklist;
17. } }

3)

1. Get Started with Apex Unit Tests from (module - Apex Testing)
2.
3. ----------
4. Name: TestVerifyDate
5.
6. @isTest
7. public class TestVerifyDate
8. {
9. static testMethod void testMethod1()
10. {
11. Date d = VerifyDate.CheckDates(System.today(),System.today()+1);
12. Date d1 = VerifyDate.CheckDates(System.today(),System.today()+60);
13. } }
14.

4)

1. Test Apex Triggers from (module - Apex Testing)
2.
3.
4. ----------
5. Name: TestRestrictContactByName
6.
7. SOURCE CODE:
8. @isTest
9. private class TestRestrictContactByName {
10. static testMethod void metodoTest()
11. {
12. List<Contact> listContact= new List<Contact>();
13. Contact c1 = new Contact(FirstName='Francesco', LastName='Riggio' , email='Test@test.com');
14. Contact c2 = new Contact(FirstName='Francesco1', LastName = 'INVALIDNAME',email='Test@test.com');
15. listContact.add(c1);
16. listContact.add(c2);
17. Test.startTest();
18. try
19. {
20. insert listContact;
21. }
22. catch(Exception ee)
23. {
24. }
25. Test.stopTest();
26. } }

Apex Testing

1. Create Test Data for Apex Tests from (module - Apex Testing)
2.
3.
4. ---
5. Create an Apex class in the public scope
6. Name: RandomContactFactory
7.
8.
9. SOURCE CODE:
10. //@isTest
11. public class RandomContactFactory {
12. public static List<Contact> generateRandomContacts(Integer numContactsToGenerate, String FName) {
13. List<Contact> contactList = new List<Contact>();
14. for(Integer i=0;i<numContactsToGenerate;i++) {
15. Contact c = new Contact(FirstName=FName + ' ' + i, LastName = 'Contact '+i);
16. contactList.add(c);
17. System.debug(c);
18. }
19. //insert contactList;
20. System.debug(contactList.size());
21. return contactList;
22. } }

Asynchronous Apex

1)

1. Use Future Methods from (module - Asynchronous Apex)
2.
3. ----------
4. SOURCE CODE1:
5.
6. public class AccountProcessor {
7. @future
8. public static void countContacts(List<Id> accountIds){
9. List<Account> accounts = [Select Id, Name from Account Where Id IN : accountIds];
10. List<Account> updatedAccounts = new List<Account>();
11. for(Account account : accounts){
12. account.Number_of_Contacts__c = [Select count() from Contact Where AccountId =: account.Id];
13. System.debug('No Of Contacts = ' + account.Number_of_Contacts__c);
14. updatedAccounts.add(account);
15. }
16. update updatedAccounts;
17. }}
18.
19. ----------
20. Source CODE2:
21.
22. @isTest
23. public class AccountProcessorTest {
24. @isTest
25. 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();
    } }
```

2)

1. Use Batch Apex from (module - Asynchronous Apex)
2.
3. ----------
4. SOURCE CODE1: LeadProcessor
5.
6. public class LeadProcessor implements Database.Batchable<sObject> {
7. public Database.QueryLocator start(Database.BatchableContext bc) {
8. return Database.getQueryLocator([Select LeadSource From Lead]);

```
9.   }
10. public void execute(Database.BatchableContext bc, List<Lead> leads){
11. for (Lead Lead : leads) {
12. lead.LeadSource = 'Dreamforce';
13. }
14. update leads;
15. }
16. public void finish(Database.BatchableContext bc){
17. } }
18.
19. ----------------------------------------------
20. SOURCE CODE2: LeadProcessorTest
21.
22. @isTest
23. public class LeadProcessorTest {
24. @testSetup
25. static void setup() {
26. List<Lead> leads = new List<Lead>();
27. for(Integer counter=0 ;counter <200;counter++){
28. Lead lead = new Lead();
29. lead.FirstName ='FirstName';
30. lead.LastName ='LastName'+counter;
31. lead.Company ='demo'+counter;
32. leads.add(lead);
33. }
34. insert leads;
35. }
36. @isTest static void test() {
37. Test.startTest();
38. LeadProcessor leadProcessor = new LeadProcessor();
39. Id batchId = Database.executeBatch(leadProcessor);
```

```
40. Test.stopTest();
41.     } }
42.
```

3)

```
1.  Use Batch Apex from (module - Asynchronous Apex)
2.
3.  ---------------------------------------------------
4.  SOURCE CODE1: LeadProcessor
5.
6.  public class LeadProcessor implements Database.Batchable<sObject> {
7.  public Database.QueryLocator start(Database.BatchableContext bc) {
8.  return Database.getQueryLocator([Select LeadSource From Lead ]);
9.  }
10. public void execute(Database.BatchableContext bc, List<Lead> leads){
11. for (Lead Lead : leads) {
12. lead.LeadSource = 'Dreamforce';
13. }
14. update leads;
15. }
16. public void finish(Database.BatchableContext bc){
17. } }
18.
19. ---------------------------------------------------
20. SOURCE CODE2: LeadProcessorTest
21.
22. @isTest
23. public class LeadProcessorTest {
24. @testSetup
```

```
25. static void setup() {
26. List<Lead> leads = new List<Lead>();
27. for(Integer counter=0 ;counter <200;counter++){
28. Lead lead = new Lead();
29. lead.FirstName ='FirstName';
30. lead.LastName ='LastName'+counter;
31. lead.Company ='demo'+counter;
32. leads.add(lead);
33. }
34. insert leads;
35. }
36. @isTest static void test() {
37. Test.startTest();
38. LeadProcessor leadProcessor = new LeadProcessor();
39. Id batchId = Database.executeBatch(leadProcessor);
40. Test.stopTest();
41.     } }
42.
```

4)

```
1.  Control Processes with Queueable Apex from (module - Asynchronous Apex)
2.
3.  ----------------------------------------------
4.  SOURCE CODE1: AddPrimaryContact
5.
6.  public class AddPrimaryContact implements Queueable
7.  {
8.  private Contact c;
```

```
9.  private String state;
10. public AddPrimaryContact(Contact c, String state)
11. {
12. this.c = c;
13. this.state = state;
14. }
15. public void execute(QueueableContext context)
16. {
17. List<Account> ListAccount = [SELECT ID, Name ,(Select id,FirstName,LastName from contacts
    ) FROM ACCOUNT WHERE BillingState = :state LIMIT 200];
18. List<Contact> lstContact = new List<Contact>();
19. for (Account acc:ListAccount)
20. {
21. Contact cont = c.clone(false,false,false,false);
22. cont.AccountId = acc.id;
23. lstContact.add( cont );
24. }
25. if(lstContact.size() >0 )
26. {
27. insert lstContact;
28. }} }
29.
30. ------------------------------------------------
31. SOURCE CODE2: AddPrimaryContactTest
32.
33. @isTest
34. public class AddPrimaryContactTest {
35. @isTest static void TestList()
36. {
37. List<Account> Teste = new List <Account>();
38. for(Integer i=0;i<50;i++)
```

```
39. {
40. Teste.add(new Account(BillingState = 'CA', name = 'Test'+i));
41. }
42. for(Integer j=0;j<50;j++)
43. {
44. Teste.add(new Account(BillingState = 'NY', name = 'Test'+j));
45. }
46. insert Teste;
47. Contact co = new Contact();
48. co.FirstName='demo';
49. co.LastName ='demo';
50. insert co;
51. String state = 'CA';
52. AddPrimaryContact apc = new AddPrimaryContact(co, state);
53. Test.startTest();
54. System.enqueueJob(apc);
55. Test.stopTest();
56.      } }
```

5)

```
1. Schedule Jobs Using the Apex Scheduler from (module - Asynchronous Apex)
2.
3. ---------------------------------------------------
4. SOURCE CODE1: DailyLeadProcessor
5.
6. public class DailyLeadProcessor implements Schedulable {
7. Public void execute(SchedulableContext SC){
8. List<Lead> LeadObj=[SELECT Id from Lead where LeadSource=null limit 200];
9. for(Lead l:LeadObj){
```

```
l.LeadSource='Dreamforce';
update l;
    } }}

----------------------------------------------
SOURCE CODE1: 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());
} }
```

Apex Integration Services

1)

1. Apex REST Callouts from (module - Apex Integration Services)
2.
3. ----------
4. SOURCE CODE: AnimalLocator
5.
6. public class AnimalLocator {
7. public static String getAnimalNameById (Integer id) {
8. String AnimalName = '';
9. Http http = new Http();
10. HttpRequest request = new HttpRequest();
11. request.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/'+id);
12. request.setMethod('GET');
13. HttpResponse response = http.send(request);
14. if (response.getStatusCode() == 200) {
15. Map<String,Object> results = (Map<String,Object>)
    JSON.deserializeUntyped(response.getBody());
16. Map<String, Object> animal = (Map<String, Object>) results.get('animal');
17. animalName = (String) animal.get('name');
18. }
19. return animalName;
20. } }
21.
22. ----------
23. SOURCE CODE: AnimalLocatorTest
24.
25. @isTest

```
26. private class AnimalLocatorTest {
27. @isTest static void testGet() {
28. Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
29. // Call method to test
30. String result = AnimalLocator.getAnimalNameById (7);
31. // Verify mock response is not null
32. System.assertNotEquals(null,result,
33. 'The callout returned a null response.');
34. System.assertEquals('dog', result,
35. 'The animal name should be \'dog\'');
36. } }
37.
38. ----------------------------------------------
39. SOURCE CODE: AnimalLocatorMock
40.
41. @isTest
42. global class AnimalLocatorMock implements HttpCalloutMock{
43.
44. // Implement this interface method
45. global HTTPResponse respond(HTTPRequest request) {
46. // Create a fake response
47. HttpResponse response = new HttpResponse();
48. response.setHeader('Content-Type', 'application/json');
49. response.setBody('{"animal":{"id":7,"name":"dog","eats":"meat","says":"i am a lovely pet
    animal"}}');
50. response.setStatusCode(200);
51. return response;
52. } }
```

2)

1. Apex SOAP Callouts from (module - Apex Integration Services)
2.
3.
4. Remote Site URL : https://th-apex-soap-service.herokuapp.com
5. ---
6. SOURCE CODE: ParkLocator
7.
8. public class ParkLocator {
9. public static string[] country(string theCountry) {
10. ParkService.ParksImplPort parkSvc = new ParkService.ParksImplPort();
11. return parkSvc.byCountry(theCountry);
12. } }
13.
14. ---
15. SOURCE CODE: ParkLocatorTest
16. @isTest
17. private class ParkLocatorTest {
18. @isTest static void testCallout() {
19. Test.setMock(WebServiceMock.class, new ParkServiceMock ());
20. String country = 'United States';
21. List<String> result = ParkLocator.country(country);
22. List<String> parks = new List<String>{'Kaziranga National Park', 'Gir National Park', 'Deer Park'};
23. System.assertEquals(parks, result);
24. } }
25.
26. ---
27. SOURCE CODE: ParkServiceMock
28. @isTest
29. global class ParkServiceMock implements WebServiceMock {

30. global void doInvoke(
31. Object stub,
32. Object request,
33. Map<String, Object> response,
34. String endpoint,
35. String soapAction,
36. String requestName,
37. String responseNS,
38. String responseName,
39. String responseType) {
40. // start - specify the response you want to send
41. ParkService.byCountryResponse response_x = new ParkService.byCountryResponse();
42. response_x.return_x = new List<String>{'Kaziranga National Park', 'Gir National Park', 'Deer Park'};
43. // end
44. response.put('response_x', response_x);
45. } }

3)

1. Apex Web Services from (module - Apex Integration Services)
2.
3. ---
4. SOURCE CODE: AccountManager
5. @RestResource(urlMapping='/Accounts/*/contacts')
6. global class AccountManager {
7. @HttpGet
8. global static Account getAccount() {
9. RestRequest req = RestContext.request;

```
String accId = req.requestURI.substringBetween('Accounts/', '/contacts');
Account acc = [SELECT Id, Name, (SELECT Id, Name FROM Contacts)
FROM Account WHERE Id = :accId];
return acc;
    }
}

----------------------------------------------
SOURCE CODE: AccountManagerTest
@isTest
private class AccountManagerTest {
private static testMethod void getAccountTest1() {
Id recordId = createTestRecord();
// Set up a test request
RestRequest request = new RestRequest();
request.requestUri = 'https://na1.salesforce.com/services/apexrest/Accounts/'+ recordId
+'/contacts' ;
request.httpMethod = 'GET';
RestContext.request = request;
// Call the method to test
Account thisAccount = AccountManager.getAccount();
// Verify results
System.assert(thisAccount != null);
System.assertEquals('Test record', thisAccount.Name);

}
// Helper method
static Id createTestRecord() {
// Create test record
Account TestAcc = new Account(
Name='Test record');
```

```
40. insert TestAcc; Contact TestCon= new Contact(
41. LastName='Test',
42. AccountId = TestAcc.id);
43. return TestAcc.Id;
44. }}
```