Creating an apex class consumer record:-

```
class ConsumerRecord {
   public static void sendEmailNotification (List<consumer__c> con){
      for(consumer__c c:con)
      {
         Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
            email.setToAddresses( new List<String>{c.email__c});
            email.setSubject('Welcome to our company');
            email.setPlainTextBody('Dear '  + ' '+ ',\n\nWelcome to MY RICE!'+'You have been seen as a valuable customer to us. PLease continue your journey with us, while we try to provide you with good quality resources.'+'\n'+
                                   "We are proud to associate with valuable customers like you and we look forward to collaborating with you by providing more and more exciting discounts or even product offers too.' + '\n'
                                   +'So why taking a step back, take a leap of faith and shop with us more, while we provide with the valuable products and offers'+'\n'+'\n'+'\n'+
                                   'Thankyou for buying '+ '' +'Here are some of the products that are brought by the customers who similarly bought products like this'+'\n\n');
            Messaging.sendEmail(new List<Messaging.SingleEmailMessage>{email});


      }
   }
}
```

Creating an apex trigger:-

```
trigger consumerTrigger on consumer__c (After insert) {
   if(trigger.isAfter && trigger.isInsert) {
      ConsumerRecord.sendEmailNotification(trigger.new);
   }
}
```

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

```
1.  Name: TestVerifyDate
2.
3.  @isTest
4.  public class TestVerifyDate
5.  {
6.  static testMethod void testMethod1()
7.  {
```

```
8.  Date d = VerifyDate.CheckDates(System.today(),System.today()+1);
9.  Date d1 = VerifyDate.CheckDates(System.today(),System.today()+60);
10.     } }
```

```
1.  Create an Apex trigger:
2.  Name: ClosedOpportunityTrigger
3.  Object: Opportunity
4.
5.  SOURCE CODE:
6.  trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {
7.  List<Task> taskList = new List<Task>();
8.  for(Opportunity opp : [SELECT Id, StageName FROM Opportunity WHERE StageName='Closed
    Won' AND Id IN : Trigger.New]){
9.  taskList.add(new Task(Subject='Follow Up Test Task', WhatId = opp.Id));
10. }
11. if(taskList.size()>0){
12. insert tasklist;
13.     } }
14.
15. SOURCE CODE: AnimalLocator
16.
17. public class AnimalLocator {
18. public static String getAnimalNameById (Integer id) {
19. String AnimalName = '';
20. Http http = new Http();
21. HttpRequest request = new HttpRequest();
22. request.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/'+id);
23. request.setMethod('GET');
24. HttpResponse response = http.send(request);
25. if (response.getStatusCode() == 200) {
26. Map<String,Object> results = (Map<String,Object>)
    JSON.deserializeUntyped(response.getBody());
27. Map<String, Object> animal = (Map<String, Object>) results.get('animal');
28. animalName = (String) animal.get('name');
29. }
30. return animalName;
31. } }
32.
33. ----------------------------------------------
34. SOURCE CODE: AnimalLocatorTest
35.
36. @isTest
37. private class AnimalLocatorTest {
38. @isTest static void testGet() {
39. Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
40. // Call method to test
41. String result = AnimalLocator.getAnimalNameById (7);
42. // Verify mock response is not null
43. System.assertNotEquals(null,result,
44. 'The callout returned a null response.');
45. System.assertEquals('dog', result,
46. 'The animal name should be \'dog\'');
47. } }
48.
49. ----------------------------------------------
50. SOURCE CODE: AnimalLocatorMock
51.
52. @isTest
53. global class AnimalLocatorMock implements HttpCalloutMock{
54.
```

```
// Implement this interface method
global HTTPResponse respond(HTTPRequest request) {
// Create a fake response
HttpResponse response = new HttpResponse();
response.setHeader('Content-Type', 'application/json');
response.setBody('{"animal":{"id":7,"name":"dog","eats":"meat","says":"i am a lovely pet animal"}}');
response.setStatusCode(200);
return response;
} }

Apex SOAP Callouts from (module - Apex Integration Services)


Remote Site URL : https://th-apex-soap-service.herokuapp.com
-----------------------------------------------
SOURCE CODE: ParkLocator

public class ParkLocator {
public static string[] country(string theCountry) {
ParkService.ParksImplPort parkSvc = new ParkService.ParksImplPort();
return parkSvc.byCountry(theCountry);
} }

-----------------------------------------------
SOURCE CODE: ParkLocatorTest
@isTest
private class ParkLocatorTest {
@isTest static void testCallout() {
Test.setMock(WebServiceMock.class, new ParkServiceMock ());
String country = 'United States';
List<String> result = ParkLocator.country(country);
List<String> parks = new List<String>{'Kaziranga National Park', 'Gir National Park', 'Deer Park'};
System.assertEquals(parks, result);
} }

-----------------------------------------------
SOURCE CODE: ParkServiceMock
@isTest
global class ParkServiceMock implements WebServiceMock {
global void doInvoke(
Object stub,
Object request,
Map<String, Object> response,
String endpoint,
String soapAction,
       String requestName,
       String responseNS,
       String responseName,
       String responseType) {
       // start - specify the response you want to send
       ParkService.byCountryResponse response_x = new ParkService.byCountryResponse();
       response_x.return_x = new List<String>{'Kaziranga National Park', 'Gir National Park', 'Deer Park'};
       // end
       response.put('response_x', response_x);
       } }
       Apex Web Services from (module - Apex Integration Services)

       -----------------------------------------------
```

```
SOURCE CODE: AccountManager
@RestResource(urlMapping='/Accounts/*/contacts')
global class AccountManager {
@HttpGet
global static Account getAccount() {
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');
insert TestAcc; Contact TestCon= new Contact(
LastName='Test',
AccountId = TestAcc.id);
return TestAcc.Id;
}}
```

```
SOURCE CODE: DisplayImage(Visualforce page)

<apex:page showHeader="false" sidebar="false">
<apex:image url="https://developer.salesforce.com/files/salesforce-developer-network-
    logo.png"/>
</apex:page>
```