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
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. }
13. Name: TestVerifyDate
15. @isTest
16. public class TestVerifyDate
17. {
18. static testMethod void testMethod1()
19. {
20. Date d = VerifyDate.CheckDates(System.today(),System.today()+1);
21. Date d1 = VerifyDate.CheckDates(System.today(),System.today()+60);
22. } }
23. Create an Apex trigger:
24. Name: ClosedOpportunityTrigger
25. Object: Opportunity
27. SOURCE CODE:
28. trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {
29. List<Task> taskList = new List<Task>();
30. for(Opportunity opp : [SELECT Id, StageName FROM Opportunity WHERE StageName='Closed Won' AND Id IN : Trigger.New]){
31. taskList.add(new Task(Subject='Follow Up Test Task', WhatId = opp.Id));
32. }
33. if(taskList.size()>0){
34. insert tasklist;
35. } }
37. SOURCE CODE: AnimalLocator
39. public class AnimalLocator {
40. public static String getAnimalNameById (Integer id) {
41. String AnimalName = '';
42. Http http = new Http();
43. HttpRequest request = new HttpRequest();
44. request.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/'+id);
45. request.setMethod('GET');
46. HttpResponse response = http.send(request);
47. if (response.getStatusCode() == 200) {
48. Map<String,Object> results = (Map<String,Object>) JSON.deserializeUntyped(response.getBody());
49. Map<String, Object> animal = (Map<String, Object>) results.get('animal');
50. animalName = (String) animal.get('name');
51. }
52. return animalName;
53. } }
55. -------------------------------------------------
56. SOURCE CODE: AnimalLocatorTest
58. @isTest
59. private class AnimalLocatorTest {
60. @isTest static void testGet() {
61. Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
62. // Call method to test
63. String result = AnimalLocator.getAnimalNameById (7);
64. // Verify mock response is not null
65. System.assertNotEquals(null,result,
66. 'The callout returned a null response.');
67. System.assertEquals('dog', result,
68. 'The animal name should be \'dog\'');
69. } }
71. -------------------------------------------------
72. SOURCE CODE: AnimalLocatorMock
74. @isTest
75. global class AnimalLocatorMock implements HttpCalloutMock{
77. // Implement this interface method
78. global HTTPResponse respond(HTTPRequest request) {
79. // Create a fake response
80. HttpResponse response = new HttpResponse();
81. response.setHeader('Content-Type', 'application/json');
82. response.setBody('{"animal":{"id":7,"name":"dog","eats":"meat","says":"i am a lovely pet animal"}}');
83. response.setStatusCode(200);
84. return response;
85. } }
87. Apex SOAP Callouts from (module - Apex Integration Services)

90. Remote Site URL : https://th-apex-soap-service.herokuapp.com
91. -------------------------------------------------
92. SOURCE CODE: ParkLocator
94. public class ParkLocator {
95. public static string[] country(string theCountry) {
96. ParkService.ParksImplPort parkSvc = new ParkService.ParksImplPort();
97. return parkSvc.byCountry(theCountry);
98. } }
100. -------------------------------------------------
101. SOURCE CODE: ParkLocatorTest
102. @isTest
103. private class ParkLocatorTest {
104. @isTest static void testCallout() {
105. Test.setMock(WebServiceMock.class, new ParkServiceMock ());
106. String country = 'United States';
107. List<String> result = ParkLocator.country(country);
108. List<String> parks = new List<String>{'Kaziranga National Park', 'Gir National Park', 'Deer Park'};
109. System.assertEquals(parks, result);
110. } }
112. -------------------------------------------------
113. SOURCE CODE: ParkServiceMock
114. @isTest
115. global class ParkServiceMock implements WebServiceMock {
116. global void doInvoke(
117. Object stub,
118. Object request,
119. Map<String, Object> response,
120. String endpoint,
121. String soapAction,
122. String requestName,
123. String responseNS,
124. String responseName,
125. String responseType) {
126. // start - specify the response you want to send
127. ParkService.byCountryResponse response\_x = new ParkService.byCountryResponse();
128. response\_x.return\_x = new List<String>{'Kaziranga National Park', 'Gir National Park', 'Deer Park'};
129. // end
130. response.put('response\_x', response\_x);
131. } }
132. Apex Web Services from (module - Apex Integration Services)
134. -------------------------------------------------
135. SOURCE CODE: AccountManager
136. @RestResource(urlMapping='/Accounts/\*/contacts')
137. global class AccountManager {
138. @HttpGet
139. global static Account getAccount() {
140. RestRequest req = RestContext.request;
141. String accId = req.requestURI.substringBetween('Accounts/', '/contacts');
142. Account acc = [SELECT Id, Name, (SELECT Id, Name FROM Contacts)
143. FROM Account WHERE Id = :accId];
144. return acc;
145. }
146. }
148. -------------------------------------------------
149. SOURCE CODE: AccountManagerTest
150. @isTest
151. private class AccountManagerTest {
152. private static testMethod void getAccountTest1() {
153. Id recordId = createTestRecord();
154. // Set up a test request
155. RestRequest request = new RestRequest();
156. request.requestUri = 'https://na1.salesforce.com/services/apexrest/Accounts/'+ recordId +'/contacts' ;
157. request.httpMethod = 'GET';
158. RestContext.request = request;
159. // Call the method to test
160. Account thisAccount = AccountManager.getAccount();
161. // Verify results
162. System.assert(thisAccount != null);
163. System.assertEquals('Test record', thisAccount.Name);
165. }
166. // Helper method
167. static Id createTestRecord() {
168. // Create test record
169. Account TestAcc = new Account(
170. Name='Test record');
171. insert TestAcc; Contact TestCon= new Contact(
172. LastName='Test',
173. AccountId = TestAcc.id);
174. return TestAcc.Id;
175. }}
176. SOURCE CODE: DisplayImage(Visualforce page)
178. <apex:page showHeader="false" sidebar="false">
179. <apex:image url="https://developer.salesforce.com/files/salesforce-developer-network-logo.png"/>
180. </apex:page>