**Apex integration services**

**Animal locator:**

**public class AnimalLocator{**

**public static String getAnimalNameById(Integer x){**  
**Http http = new Http();**  
**HttpRequest red = new HttpRequest();**  
**req.setEndpoint('**[**https://th-apex-http-callout.herokuapp.com/animals/'**](https://www.youtube.com/redirect?event=comments&redir_token=QUFFLUhqazcyalhSTU1LVTFwS3hxT2ZycFhGcVlmTzYwZ3xBQ3Jtc0trZlY4MU9xWUtjV2ZqaU5RNU5jNmxyM1I2cmpMWnJDQVpHR1NCTE5YWmdNUnhHRVJpUkdWbGRBeUJqb2s3MVViTS1ySlNpWmFlOTZUeThXZnRKV3pNZVpxQUdWT24wTE1jQV9PdmhiV3pDSlJSMzFjMA&q=https%3A%2F%2Fth-apex-http-callout.herokuapp.com%2Fanimals%2F%27&stzid=UgxmbnC5fXnY2npTqYZ4AaABAg) **+ x);**

**req.setMethod('GET');**

**Map<String, Object> animal= new Map<String, Object>();**  
**HttpResponse res = http.send(req);**  
**if (res.getStatusCode() == 200) }**

**Map<String, Object> results = (Map<String, Object>)JSON.deserializeUntyped(res.getBody());**  
**animal = (Map<String, Object>) results.get('animal');**  
**}**  
**return (String)animal.get('name');**  
**}**  
**}**

**Animal locator test:**

**@isTest**  
  
**private class AnimalLocatorTest{**  
  
**@isTest static void AnimalLocatorMock1() {**  
  
**Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());**  
  
**string result = AnimalLocator.getAnimalNameById(3);**  
  
**String expectedResult = 'chicken';**  
  
**System.assertEquals(result,expectedResult );**  
  
**}**  
  
**}**

**Animal locator mock:**

**@isTest**  
**global class AnimalLocatorMock implements HttpCalloutMock {**  
**// 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('{"animals": ["majestic badger", "fluffy bunny", "scary bear", "c**hicken", "mighty **moose"]}');**  
**response.setStatusCode(200);**  
  
**return response;**   
  
**}**  
  
**}**

**park locator:**

**public class ParkLocator {**

**public static string[] country(string theCountry) {**

**ParkService.ParksImplPort  parkSvc = new  ParkService.ParksImplPort(); // remove space**  
**return parkSvc.byCountry(theCountry);**

**}**

**}**

**Park locator test :**

**@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>{'Yellowstone', 'Mackinac National Park', 'Yosemite'};**  
**System.assertEquals(parks, result);**

**}**  
**}**

**park locator mock:**

**@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>{'Yellowstone', 'Mackinac National Park', 'Yosemite'};**  
**// end**  
**response.put('response\_x', response\_x);**   
**}**

**}**

**account manager:**

**@RestResource(urlMapping='/Accounts/\*/contacts')**  
**global class AccountManager {**  
**@HttpGet**

**global static Account getAccount() {**  
**RestRequest re q = 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;**  
**}**  
**}**

**Account manager test:**

**@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/'+**](https://www.youtube.com/redirect?event=comments&redir_token=QUFFLUhqa3BIdmVJSnBpYm0yVzlFX215OHI1NXNScjdhZ3xBQ3Jtc0tseHRsV05PWnlqWlFtRWpZWi05VmFscEhBMzhaeXhUcUY4c3ExYWU2dll3MW9nRVNldlBhTU04UmNhelN2Yms3WENuZEJ5LWJMUHp5X2hGWHRkSlR3a1ptbHVuQ1A0ZEcxUHZuS1dfSW93SkxKbGQ4OA&q=https%3A%2F%2Fna1.salesforce.com%2Fservices%2Fapexrest%2FAccounts%2F%27%2B&stzid=UgydtaZ7DrxVzx0JUJl4AaABAg) **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**](https://www.youtube.com/redirect?event=comments&redir_token=QUFFLUhqbVlaVTlmM05BNGF0X3gzRmxNcV9odVJ6TE9DUXxBQ3Jtc0trZTR6RTBLWkpVRmZXUEctTGxBTWJRaUpsOGlCMFJJTWh0WGhkTlM2SkdqNDBQT0ZzZm1jd3lOTnZqS2xya3VOelo0TWY5QmhIUmZPdG4ycmdmOW1odTVrLWZsb1pjeldHTWFPdnhwX0NVU2tMRS1tYw&q=http%3A%2F%2Ftestacc.id%2F&stzid=UgydtaZ7DrxVzx0JUJl4AaABAg)**;**  
**}**   
**}**