AnimalLocator:-

public class AnimalLocator

{

public static String getAnimalNameById(Integer id)

{

Http http = new Http();

HttpRequest request = new HttpRequest();

request.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/'+id);

request.setMethod('GET');

HttpResponse response = http.send(request);

String strResp = '';

system.debug('\*\*\*\*\*\*response '+response.getStatusCode());

system.debug('\*\*\*\*\*\*response '+response.getBody());

// If the request is successful, parse the JSON response.

if (response.getStatusCode() == 200)

{

// Deserializes the JSON string into collections of primitive data types.

Map<String, Object> results = (Map<String, Object>) JSON.deserializeUntyped(response.getBody());

// Cast the values in the 'animals' key as a list

Map<string,object> animals = (map<string,object>) results.get('animal');

System.debug('Received the following animals:' + animals );

strResp = string.valueof(animals.get('name'));

System.debug('strResp >>>>>>' + strResp );

}

return strResp ;

}

}

AnimalLocatorTest class:-

@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);

}

}

AnimalLocatorMock :-

@isTest

global class AnimalLocatorMock implements HttpCalloutMock {

global HTTPResponse respond(HTTPRequest request) {

HttpResponse response = new HttpResponse();

response.setHeader('Content-Type', 'application/json');

response.setBody('{"animal":{"id":1,"name":"chicken","eats":"chicken food","says":"cluck cluck"}}');

response.setStatusCode(200);

return response;

}

}

ParkLocator Class:-

public class ParkLocator {

public static String[] country(String ctry) {

ParkService.ParksImplPort prk =

new ParkService.ParksImplPort();

return prk.byCountry(ctry);

}

}

ParklocatorTest:-

@isTest

private class ParkLocatorTest {

@isTest static void testCallout() {

// This causes a fake response to be generated

Test.setMock(WebServiceMock.class, new ParkServiceMock());

// Call the method that invokes a callout

List<String> result = new List<String>();

List<String> expectedvalue = new List<String>{'Park1','Park2','Park3'};

result = ParkLocator.country('India');

// Verify that a fake result is returned

System.assertEquals(expectedvalue, result);

}

}

ParkSErviceMock:-

AccountManager:-

@RestResource(urlMapping='/Accounts/\*/contacts')

global with sharing class AccountManager{

@HttpGet

global static Account getAccount(){

RestRequest request = RestContext.request;

String accountId = request.requestURI.substringBetween('Accounts/','/contacts');

system.debug(accountId);

Account objAccount = [SELECT Id,Name,(SELECT Id,Name FROM Contacts) FROM Account WHERE Id = :accountId LIMIT 1];

return objAccount;

}

AccountManagerTest:-

@isTest

private class AccountManagerTest{

static testMethod void testMethod1(){

Account objAccount = new Account(Name = 'test Account');

insert objAccount;

Contact objContact = new Contact(LastName = 'test Contact',

AccountId = objAccount.Id);

insert objContact;

Id recordId = objAccount.Id;

RestRequest request = new RestRequest();

request.requestUri =

'https://sandeepidentity-dev-ed.my.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 Account', thisAccount.Name);

}

}