Apex Integration Services

Apex Integration Overview

Apex REST Callouts

-

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 ;
    }

}
```

<u>AnimalLocatorTest</u>

```
@isTest

private class AnimalLocatorTest {

    @isTest static void AnimalLocatorMock1(){

        Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());

        string result = AnimalLocator.getAnimalNameById(1);

        string expectedResult = 'chicken';

        System.assertEquals(result, expectedResult);

    }


}
```

<u>AnimalLocatorMock</u>

```
@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":"chiken food","says":"cluck
cluck"}}');

        response.setStatusCode(200);

        return response;

    }


}
```

<u>AnimalsCalloutsTest</u>

```
@isTest
private class AnimalsCalloutsTest {
    @isTest static  void testGetCallout() {
        // Create the mock response based on a static resource
        StaticResourceCalloutMock mock = new StaticResourceCalloutMock();
        mock.setStaticResource('GetAnimalResource');
        mock.setStatusCode(200);
        mock.setHeader('Content-Type', 'application/json;charset=UTF-8');
        // Associate the callout with a mock response
        Test.setMock(HttpCalloutMock.class, mock);
        // Call method to test
        HttpResponse result = AnimalsCallouts.makeGetCallout();
        // Verify mock response is not null
        System.assertNotEquals(null,result, 'The callout returned a null response.');
        // Verify status code
        System.assertEquals(200,result.getStatusCode(), 'The status code is not 200.');
        // Verify content type
        System.assertEquals('application/json;charset=UTF-8',
          result.getHeader('Content-Type'),
          'The content type value is not expected.');
        // Verify the array contains 3 items
        Map<String, Object> results = (Map<String, Object>)
```

```
            JSON.deserializeUntyped(result.getBody());
        List<Object> animals = (List<Object>) results.get('animals');
        System.assertEquals(3, animals.size(), 'The array should only contain 3 items.');
    }
    @isTest
    static void testPostCallout() {
    // Set mock callout class
    Test.setMock(HttpCalloutMock.class, new AnimalsHttpCalloutMock());
    // This causes a fake response to be sent
    // from the class that implements HttpCalloutMock.
    HttpResponse response = AnimalsCallouts.makePostCallout();
    // Verify that the response received contains fake values
    String contentType = response.getHeader('Content-Type');
    System.assert(contentType == 'application/json');
    String actualValue = response.getBody();
    System.debug(response.getBody());
    String expectedValue = '{"animals": ["majestic badger", "fluffy bunny", "scary bear", "chicken",
"mighty moose"]}';
    System.assertEquals(expectedValue, actualValue);
    System.assertEquals(200, response.getStatusCode());
}
}
```

Apex SOAP Callouts

ParkService

```
public class ParkService {
    public class byCountryResponse {
        public String[] return_x;
        private String[] return_x_type_info = new String[]{'return','http://parks.services/',null,'0','-1','false'};
        private String[] apex_schema_type_info = new String[]{'http://parks.services/','false','false'};
        private String[] field_order_type_info = new String[]{'return_x'};
    }
    public class byCountry {
        public String arg0;
        private String[] arg0_type_info = new String[]{'arg0','http://parks.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new String[]{'http://parks.services/','false','false'};
        private String[] field_order_type_info = new String[]{'arg0'};
    }
    public class ParksImplPort {
        public String endpoint_x = 'https://th-apex-soap-service.herokuapp.com/service/parks';
        public Map<String,String> inputHttpHeaders_x;
        public Map<String,String> outputHttpHeaders_x;
        public String clientCertName_x;
        public String clientCert_x;
        public String clientCertPasswd_x;
        public Integer timeout_x;
        private String[] ns_map_type_info = new String[]{'http://parks.services/', 'ParkService'};

        public String[] byCountry(String arg0) {

            ParkService.byCountry request_x = new ParkService.byCountry();

            request_x.arg0 = arg0;

            ParkService.byCountryResponse response_x;

            Map<String, ParkService.byCountryResponse> response_map_x = new Map<String,
ParkService.byCountryResponse>();

            response_map_x.put('response_x', response_x);

            WebServiceCallout.invoke(

              this,

              request_x,

              response_map_x,

              new String[]{endpoint_x,

              '',

              'http://parks.services/',

              'byCountry',

              'http://parks.services/',

              'byCountryResponse',

              'ParkService.byCountryResponse'}

            );

            response_x = response_map_x.get('response_x');

            return response_x.return_x;

        }

    }

}
```

ParkLocator

```
public class ParkLocator {
    public static String[] country(String country){
        ParkService.ParksImplPort Locator = new ParkService.ParksImplPort();
        return Locator.byCountry(country);
    }
}
```

<u>ParkLocatorTest</u>

```
@isTest

private class ParkLocatorTest {

    testMethod static void testCallout(){

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

         String country = 'United States';

         String[] result = ParkLocator.country(country);

        System.assertEquals(new List<String>{'Garner State Park', 'Fowler Park', 'Hoosier National Forest
Park'}, result);

    }

}
```

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) {

            ParkService.byCountryResponse response_x = new ParkService.byCountryResponse();

            response_x.return_x = new List<String>{'Garner State Park', 'Fowler Park', 'Hoosier National
Forest Park'};

                response.put('response_x',response_x);

        }


}
```

Apex Web Services

AccountManager

```
@RestResource(urlMapping='/Accounts/*/contacts')
global with sharing class AccountManager {

    @HttpGet
    global static account getAccount() {

        RestRequest request = RestContext.request;

        String accountId = request.requestURI.substring(request.requestURI.lastIndexOf('/')-18,
          request.requestURI.lastIndexOf('/'));
        List<Account> a = [select id, name, (select id, name from contacts) from account where id =
:accountId];
        List<contact> co = [select id, name from contact where account.id = :accountId];
        system.debug('** a[0]= '+ a[0]);
        return a[0];

    }

}
```

<u>AccountManagerTest</u>

```
@istest

public class AccountManagerTest {

@istest static void testGetContactsByAccountId() {

Id recordId = createTestRecord();

// Set up a test request

RestRequest request = new RestRequest();

request.requestUri =

'https://yourInstance.salesforce.com/services/apexrest/Accounts/'+ recordId+'/Contacts';

request.httpMethod = 'GET';

RestContext.request = request;


Account thisAccount = AccountManager.getAccount();

System.assert(thisAccount!= null);

System.assertEquals('Test record', thisAccount.Name);

}


// Helper method

static Id createTestRecord() {


// Create test record

Account accountTest = new Account(

Name='Test record');

insert accountTest;
Contact contactTest = new Contact(

FirstName='John',

LastName='Doe',

AccountId=accountTest.Id

);

return accountTest.Id;

}

}
```