

# Testing REST services and APIs in Java using REST Assured and JUnit

- [Overview](#)
  - [REST Assured concepts](#)
- [Description](#)
- [Tips](#)
- [References](#)

## Overview

Services are usually accessed using APIs. REST APIs are common both in internal services and in public facing services.

Whenever testing REST APIs, we can either be targeting the client/consumer or the server/producer.

In this tutorial we'll show you how a REST API can be tested using Java and the REST Assured library, together with JUnit and WireMock.

[REST Assured](#) is a Java library that implements a DSL in order to easily validate REST API based services by abstracting low-level details of HTTP requests and of parsing responses.

It provides a [syntax](#) based on Gherkin (i.e. given, when, then) for better readability.

From REST Assured documentation, imagine that you have a REST based service accessible in the `http://localhost:8080/lotto/{id}` endpoint, and returning a JSON content.

```
{
  "lotto": {
    "lottoId": 5,
    "winning-numbers": [2, 45, 34, 23, 7, 5, 3],
    "winners": [
      {
        "winnerId": 23,
        "numbers": [2, 45, 34, 23, 3, 5]
      },
      {
        "winnerId": 54,
        "numbers": [52, 3, 12, 11, 18, 22]
      }
    ]
  }
}
```

Also ensure that you have a JUnit test to check that the HTTP response code is OK (i.e. 200) and that the "lotto" object returned is correct. This could be something as follows.

```
@Test public void
lotto_resource_returns_200_with_expected_id_and_winners() {

    when().
        get("/lotto/{id}", 5).
    then().
        statusCode(200).
        body("lotto.lottoId", equalTo(5),
            "lotto.winners.winnerId", hasItems(23, 54));

}
```

This takes advantage of the [JsonPath](#), a simple JSON parser (there's also an equivalent for XML: [XmlPath](#)).

If required, it's also possible to perform JSON Schema validation.

## REST Assured concepts

In REST Assured it's all about being able to quickly perform HTTP REST requests and to verify the response easily, no matter if it's JSON or XML based.

Main concepts include:

- **request specification** - the definition of the request (e.g. URL, authentication, etc); the object **RequestSpecification** is usually specified using *given()* and *when()*.
- **response specification** - the definition of the response (e.g. body, status code); the object **ResponseSpecification** is usually specified using *then()*
- **parser** - an entity responsible for processing and abstract the body content of the response, so that data can easily be referred in the matchers /assertions
- **matchers** - used to match either built-in ones (io.restassured.matcher.RestAssuredMatchers.\*) or from **Hamcrest** (org.hamcrest.Matchers.\*).

Note: it's possible to reuse request specifications or response specifications between multiple requests/responses for different tests (more info [here](#)). This can be extremely useful if you need to perform a set of common assertions (e.g. status code being OK) for a bunch of tests.

## Description

In this tutorial, we'll use the code from [Bas Dijkstra open-source workshop on REST Assured](#); you may want to check the [workshop presentation](#) for further detail.

We aim to test a service that provides an API (i.e. <http://api.zippopotam.us/>) to obtain location data based on postal/zip codes on several countries.

### Please note

To be clear, in this project, tests are performed against the server provided by WireMock that mocks responses from the API; this was done just by convenience of the workshop, so it's not dependent on the public server availability. In a real scenario, you would perform tests against the target service or you use WireMock to validate your client/consumer library implementation.

The workshop provides some exercises, stored under [src/test/java/exercises](#); it also provides the answers.

As an example, let's implement a test for checking that the API returns the correct state, in the correct order, for a given zip code of US.

### example of a test from RestAssuredAnswers1Test.java

```
/* *****
 * Send a GET request to /us/90210 and check
 * that the state associated with the first place
 * in the list returned is equal to 'California'
 *
 * Use the GPath expression "places[0].state" to
 * extract the required response body element
 * ***** */

@Test
public void requestUsZipCode90210_checkStateForFirstPlace_expectCalifornia() {

    given().
        spec(requestSpec).
    when().
        get("/us/90210").
    then().
        assertThat().
            body("places[0].state", equalTo("California"));
}
```

Another variant of it could be performing data-driven testing to check the returned "state" for a set of input data (e.g. country+zip code).

#### code snippet from RestAssuredAnswers2Test.java

```
..
/*****
 * Create a DataProvider with three test data rows:
 * -----
 * country code | zip code | state
 * -----
 * us           | 90210   | California
 * us           | 12345   | New York
 * ca           | Y1A     | Yukon
 *****/

@DataProvider
public static Object[][] zipCodeData() {
    return new Object[][] {
        { "us", "90210", "California" },
        { "us", "12345", "New York" },
        { "ca", "Y1A", "Yukon" }
    };
}

/*****
 * Request zip code data for the given country / zip
 * combinations by sending a GET to /<countryCode>/<zipCode>.
 *
 * Use the test data collection created
 * above. Check that the state returned by the API
 * matches the expected value.
 *
 * Use the GPath expression "places[0].state" to
 * extract the required response body element
 *****/

@Test
@UseDataProvider("zipCodeData")
public void checkStateForCountryCodeAndZipCode(String countryCode, String zipCode, String
expectedState) {

    given().
        spec(requestSpec).
    and().
        pathParam("countryCode", countryCode).
        pathParam("zipCode", zipCode).
    when().
        get("/{countryCode}/{zipCode}").
    then().
        assertThat().
        body("places[0].state", equalTo(expectedState));
}

...
```

We'll see ahead how this specific data-driven test will be mapped to Jira.

Tests can be run using Maven; in this case, we'll be only executing the ones inside the "answers" package.

#### example of a Bash script to run the tests

```
mvn clean compile test -Dtest=answers.*
```

Since the previous command generates multiple JUnit XML files, we may need to merge them into a single XML file so it can be submitted into a Test Execution more easily. That can be achieved by using the [junit-merge](#) utility.

```
junit-merge -d target/surefire-reports/ -o merged-test-results.xml
```

After successfully running the tests and generating the aggregated JUnit XML report (e.g., merged-test-results.xml), it can be imported to Xray (either by the [REST API](#) or through the **Import Execution Results** action within the Test Execution, or even by using a [CI tool of your choice](#)).

Calculator / CALC-8028  
Execution results - merged-test-results.xml - [1606402588763]

Edit Comment Assign More Start Progress Resolve Issue Close Issue Admin

**Details**

Type: Test Execution  
Priority: Major  
Affects Version/s: None  
Component/s: None  
Labels: None  
Test Environments: None  
Test Plan: None

Status: OPEN (View Workflow)  
Resolution: Unresolved  
Fix Version/s: None

**Description**  
Execution results imported from external source

**Tests**

Overall Execution Status

19 PASS

Total Tests: 19

Filter(s)

Apply Rank

Show 100 entries Columns

Rank	Key	Summary	Test Type	#Req	#Def	Assignee	Status
1	CALC-8029	checkStateForCountryCodeAndZipCode[1: us, 12345, New York]	Generic	0	0	Administrator	PASS
2	CALC-8030	requestUsZipCode90210_checkStateForFirstPlace_expectCalifornia	Generic	0	0	Administrator	PASS
3	CALC-8031	extractCountryFromResponse	Generic	0	0	Administrator	PASS
4	CALC-8032	requestUsZipCode99999_checkResponseCode_expect404	Generic	0	0	Administrator	PASS
5	CALC-8033	getDeZipCode24848_checkNumberOfPlacesStartingWithKlein_expect2	Generic	0	0	Administrator	PASS
6	CALC-8034	requestUsZipCode90210_checkResponseCode_expect200	Generic	0	0	Administrator	PASS
7	CALC-8035	requestUsZipCode90210_checkContentType_expectApplicationJson	Generic	0	0	Administrator	PASS
8	CALC-8036	checkStateForCountryCodeAndZipCode[2: ca, Y1A, Yukon]	Generic	0	0	Administrator	PASS
9	CALC-8037	useResponseSpecification	Generic	0	0	Administrator	PASS
10	CALC-8038	getDeZipCode24848_checkThirdPlaceInList_expectKropp	Generic	0	0	Administrator	PASS

Each JUnit test is mapped to a Generic Test in Jira, and the **Generic Test Definition** field contains the name of the package, the class and the method name that implements the Test Case. The summary of each Test issue is filled out with the name of the method corresponding to the JUnit Test.

The Execution Details page also shows information about the Test Suite, which in this case corresponds to the Test class, including its namespace.

Calculator

Calculator Board Enhance...

Backlog

Active sprints

Releases

Reports

Issues

Components

Structure

Xray Reports

Xray Test Repository

Xray Test Plan Board

Automated Steps Library

Add-ons

PROJECT SHORTCUTS

Add a link to useful information for your whole team to see.

Add link

Calculator / Test Execution: CALC-8028 / Test: CALC-8030

requestUsZipCode90210\_checkStateForFirstPlace\_expectCalifornia

Export Test as Text

Return to Test Execution

Execute with Exploratory App

Previous

Next

Execution Status PASS

Started On: 26/Nov/20 2:56 PM

Finished On: 26/Nov/20 2:56 PM

Assignee: Administrator

Executed By: Administrator

Tests -

environments:

Versions: -

Revision: -

Affected Requirements

None

Comment

Preview Comment

Execution Defects (0)

Create Defect

Create Sub-Task

Add Defects

Execution Evidence (0)

Add Evidence

Execution Details

Test Description

None

Custom Fields

There are no Test Run Custom Fields defined.

Test Details

Test Type: Generic

Definition: answers.RestAssuredAnswersTest.requestUsZipCode90210\_checkStateForFirstPlace\_expectCalifornia

Results

Context	Output	Duration	Status
TestSuite answers.RestAssuredAnswersTest	-	59.000 ms	PASS

Activity



### Please note

In the case of data-driven tests, as the JUnit XML report treats each row of the data provider as a different "test case," you will end up with several Tests in Xray.

	Rank	Key	Summary	Test Type	#Req	#Def	Assignee	Status
<input type="checkbox"/>	14	CALC-8042	checkStateForCountryCodeAndZipCode[0: us, 90210, California]	Generic	0	0	Administrator	PASS
<input type="checkbox"/>	1	CALC-8029	checkStateForCountryCodeAndZipCode[1: us, 12345, New York]	Generic	0	0	Administrator	PASS
<input type="checkbox"/>	8	CALC-8036	checkStateForCountryCodeAndZipCode[2: ca, Y1A, Yukon]	Generic	0	0	Administrator	PASS

Calculator / Test Execution: CALC-8028 / Test: CALC-8042

#### checkStateForCountryCodeAndZipCode[0: us, 90210, California]

Execution Status **PASS**

Started On: 26/Nov/20 2:56 PM Finished On: 26/Nov/20 2:56 PM

#### Affected Requirements

None

Comment

Preview Comment

Execution Defects (0)

### Execution Details

#### Test Description

None

#### Custom Fields

There are no Test Run Custom Fields defined.

#### Test Details

Test Type: Generic

Definition: answers.RestAssuredAnswers2Test.checkStateForCountryCodeAndZipCode[0: us, 90210, California]

#### Results

Context	Output
TestSuite answers.RestAssuredAnswers2Test	-

## Tips

- Multiple runs of your tests can be grouped and consolidated in a Test Plan, so you can have an updated overview of their current state
- After importing the results, you can link the corresponding Test issues with an existing requirement or user story and thus track coverage directly on the respective issue, or even on a Agile board

Calculator / CALC-8048

As a user, I can obtain location data for a given country's zipcode

Edit

Q Comment

Assign

More

Start Progress

Close Issue

Admin

Details

Type: Story

Priority: Major

Affects Version/s: None

Component/s: None

Labels: None

Requirement Status: UNCOVERED

Status: OPEN (View Workflow)

Resolution: Unresolved

Fix Version/s: None

People

Assign

Report

Votes

Watch

Dates

Create

Update

Agile

View c

Description

As a user, I can obtain location data for a given country's zipcode

Test Coverage

No Tests were found testing the requirement.

Attachments

Create Test

Create Sub-Test Execution

+ Link

Tests...

Test Sets...

Calculator / CALC-8048

As a user, I can obtain location data for a given country's zipcode

Edit Comment Assign More Start Progress Resolve Issue Close Issue Admin

**Details**

Type: Story  
Priority: Major  
Affects Version/s: None  
Component/s: None  
Labels: None  
Requirement Status: OK

Status: OPEN (View Workflow)  
Resolution: Unresolved  
Fix Version/s: None

**Description**

As a user, I can obtain location data for a given country's zipcode

**Test Coverage**

Create Test Create Sub-Test Execution + Link

TEST COVERAGE FOR THE FOLLOWING ANALYSIS SCOPE

Scope: Version: Version: None - latest execution; Environment: All Environments OK

Filter(s)

P	Status	Resolution	Key	Summary	Test Runs	Test Status
<input type="checkbox"/>	OPEN	Unresolved	CALC-8029	checkStateForCountryCodeAndZipCode[1: us, 12345, New York]	ID	PASS
<input type="checkbox"/>	OPEN	Unresolved	CALC-8030	requestUsZipCode90210_checkStateForFirstPlace_expectCalifornia	ID	PASS
<input type="checkbox"/>	OPEN	Unresolved	CALC-8031	extractCountryFromResponse	ID	PASS
<input type="checkbox"/>	OPEN	Unresolved	CALC-8032	requestUsZipCode99999_checkResponseCode_expect404	ID	PASS
<input type="checkbox"/>	OPEN	Unresolved	CALC-8033	getDeZipCode24848_checkNumberOfPlacesStartingWithKlein_expect2	ID	PASS
<input type="checkbox"/>	OPEN	Unresolved	CALC-8034	requestUsZipCode90210_checkResponseCode_expect200	ID	PASS
<input type="checkbox"/>	OPEN	Unresolved	CALC-8035	requestUsZipCode90210_checkContentType_expectApplicationJson	ID	PASS
<input type="checkbox"/>	OPEN	Unresolved	CALC-8036	checkStateForCountryCodeAndZipCode[2: ca, Y1A, Yukon]	ID	PASS
<input type="checkbox"/>	OPEN	Unresolved	CALC-8037	useResponseSpecification	ID	PASS
<input type="checkbox"/>	OPEN	Unresolved	CALC-8038	getDeZipCode24848_checkThirdPlaceInList_expectKropp	ID	PASS

Showing 1 to 10 of 10 entries First Previous 2 Next Last

Calculator

Calculator Board Enhance...

Backlog

Active sprints

Releases

Reports

Issues

Components

Structure

Xray Reports

Xray Test Repository

Xray Test Plan Board

Automated Steps Library

Add-ons

All sprints Switch sprint

QUICK FILTERS: Only My Issues Recently Updated

TO DO

CALC-962  
As a user, I can calculate add positive numbers  
NOK

CALC-983  
As a user, I can calculate the sum of 2 numbers  
Sub Test Execution for CALC-983  
OK

CALC-970  
calculator screen does not show anything  
NOK

CALC-1200

IN PROGRESS

CALC-8048  
As a user, I can obtain location data for a given country's zipcode  
OK

CALC-3208  
As a user, I can calculate the sum of two numbers  
NOK

CALC-3214  
all my sum related tests for v3.0  
NOK

CALC-983  
As a user, I can calculate the sum of 2 numbers  
NOK

DONE

CALC-1076  
As a user, I can calculate the sum of 2 numbers  
NOK

CALC-1086  
Sub Test Execution for CALC-1086  
OK

CALC-1184  
Test Plan with sprints  
OK

CALC-1186  
Test Execution for CALC-1186  
OK

## References

- <https://rest-assured.io/>
- <https://github.com/rest-assured/rest-assured>
- REST Assured Usage Guide
- Bas Dijkstra open-source workshop on REST Assured
- Bas Dijkstra blog entry about WireMock
- WireMock