Contents

Revision History

Version	Date	Author
1.0	20/1/2017	Nishant Makkar Nimish Kumar

1. Introduction

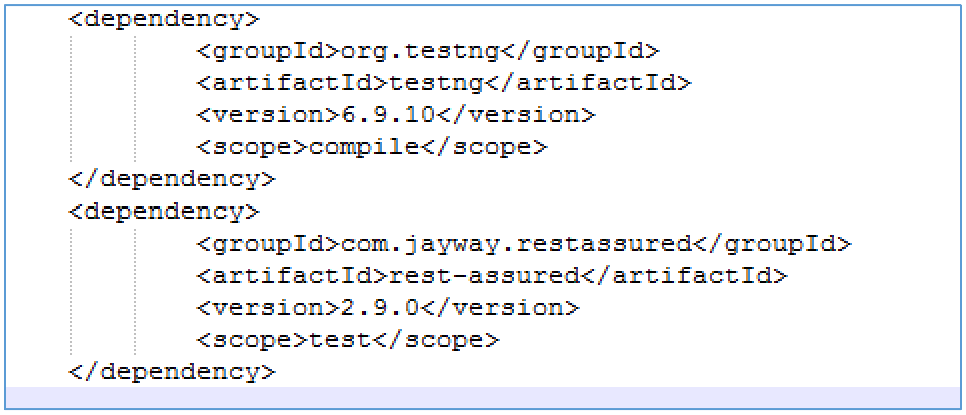
REST Assured library is used to test REST APIs. It is developed by **JayWay** Company and it is a really powerful catalyzer for automation testing of REST-services. REST-assured provides a variety of features, such as XPath-Validation, Specification Reuse, and easy file uploads.

2. Objective

This document introduce Rest-Assured framework with sample code and good examples. After understanding it, we can decide how Rest-Assured is useful in terms of Automation in different kinds of project.

3. Pre-Requisites

- **i.** IDE.
- **ii.** Java should be installed and build path should be configured.
- **iii.** Maven with configured build path. (Need Rest-Assured jar if not using maven).



```
<dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>6.9.10</version>
        <scope>compile</scope>
</dependency>
<dependency>
        <groupId>com.jayway.restassured</groupId>
        <artifactId>rest-assured</artifactId>
        <version>2.9.0</version>
        <scope>test</scope>
</dependency>
```

- **iv.** TestNG (optional)
- **v.** Add below mentioned maven dependencies in pom.xml:

4. Scenarios

Source Code is present in following Git Repo:

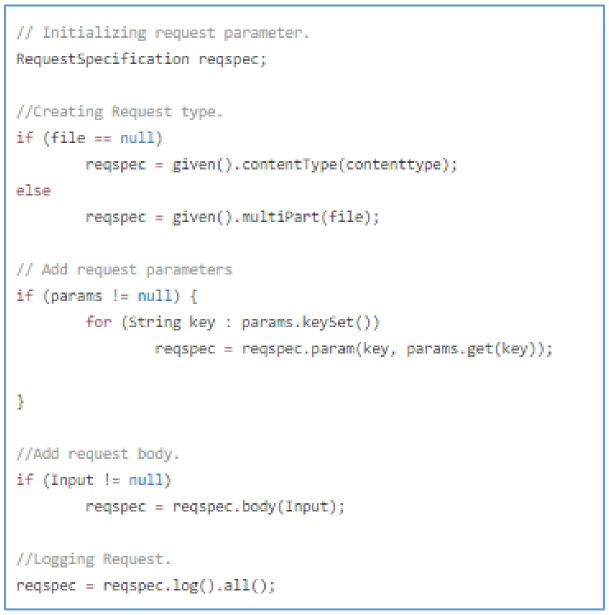
https://github.com/ggarg1xav/Rest-POC/

Scenarios covered are:

i. Performing API calls with :
   a. Input Body.
   b. Parameters.
   c. Files.

ii. Getting response and deciding its format as :
   a. String
   b. JSON Object
   c. JSON Array.

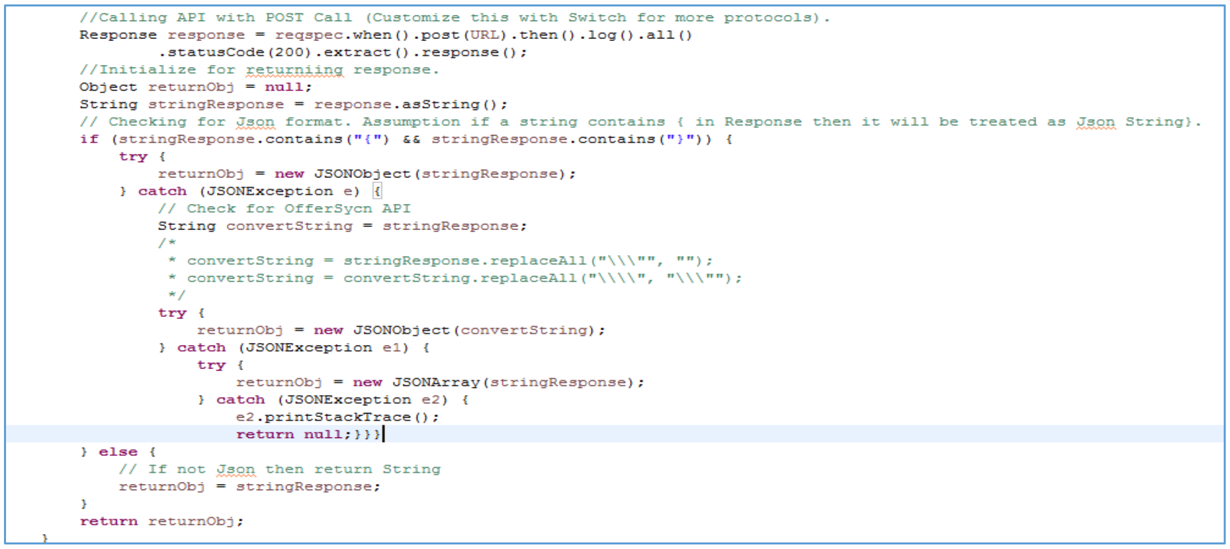
Following Repository contains ApiHelper which is performing all API calls in following manner:



```
// Initializing request parameter.
RequestSpecification reqspec;

//Creating Request type.
if (file == null)
        reqspec = given().contentType(contenttype);
else
        reqspec = given().multiPart(file);

// Add request parameters
if (params != null) {
        for (String key : params.keySet())
                reqspec = reqspec.param(key, params.get(key));

}

//Add request body.
if (Input != null)
        reqspec = reqspec.body(Input);

//Logging Request.
reqspec = reqspec.log().all();
```

Here idea is to collate all request parameters (Request Spec Object) in one object in order to make API call.

Calling API & deciding Response type:



```
        //Calling API with POST Call (Customize this with Switch for more protocols).
        Response response = reqspec.when().post(URL).then().log().all()
                .statusCode(200).extract().response();
        //Initialize for returniing response.
        Object returnObj = null;
        String stringResponse = response.asString();
        // Checking for Json format. Assumption if a string contains { in Response then it will be treated as Json String}.
        if (stringResponse.contains("{") && stringResponse.contains("}")) {
            try {
                returnObj = new JSONObject(stringResponse);
            } catch (JSONException e) {
                // Check for OfferSycn API
                String convertString = stringResponse;
                /*
                 * convertString = stringResponse.replaceAll("\\\"", "");
                 * convertString = convertString.replaceAll("\\\\", "\\\"");
                 */
                try {
                    returnObj = new JSONObject(convertString);
                } catch (JSONException e1) {
                    try {
                        returnObj = new JSONArray(stringResponse);
                    } catch (JSONException e2) {
                        e2.printStackTrace();
                        return null;}}}
        } else {
            // If not Json then return String
            returnObj = stringResponse;
        }
        return returnObj;
    }
```

After Calling API we are:

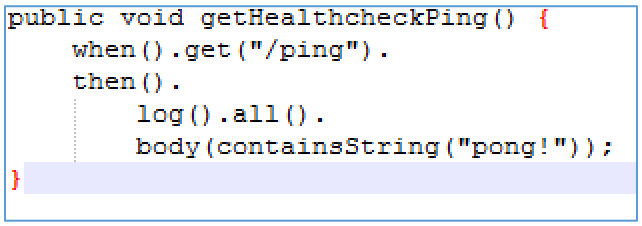
- Validating response code.
- Logging Response.
- Deciding type of response and returning it.

Method Return type is Object.

5. Advantages

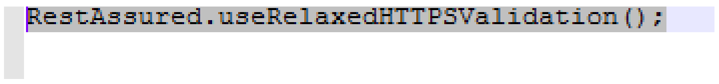
Below are some additional advantages of Rest Assured library:

i. Rest-Assured framework provides built-in logger functionality by using "log().all()" method:



```
public void getHealthcheckPing() {
    when().get("/ping").
    then().
        log().all().
        body(containsString("pong!"));
}
```

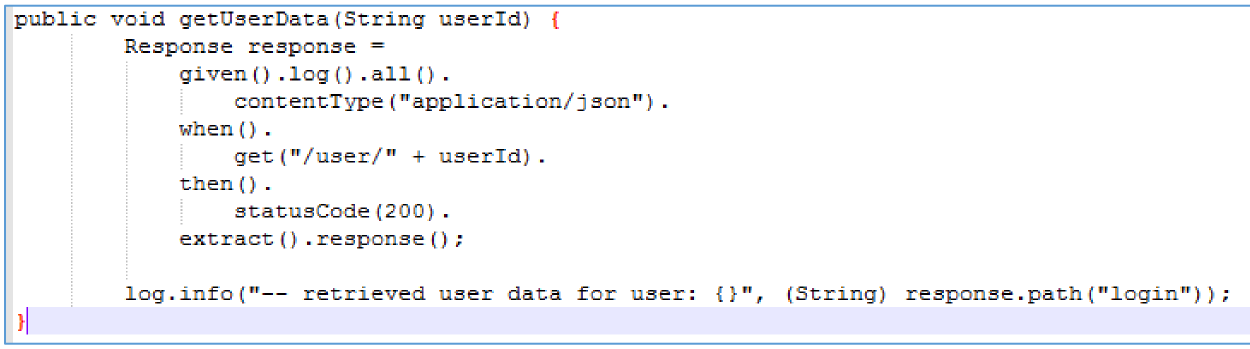
ii. HTTPS Validation :If you have some HTTPS issues and test flow doesn't check any security cases :



```
RestAssured.useRelaxedHTTPSValidation();
```

iii. Assertion: Rest Assured is also providing the assertion functionality for this we have to add TestNg or Junit maven dependencies as mentioned above in step 3(ii).

iv. One of the most obvious use case would be extracting values from response. Below is an example of extracting login value from getUserData() method response:



```
public void getUserData(String userId) {
        Response response =
            given().log().all().
                contentType("application/json").
            when().
                get("/user/" + userId).
            then().
                statusCode(200).
            extract().response();

        log.info("-- retrieved user data for user: {}", (String) response.path("login"));
}
```

v. File Handling is possible.

6. Limitations

Below are some limitations of Rest Assured library:

i. It could be used only for Rest API validations not for SOAP.

ii. TestNg or Junit libraries/ maven dependencies should be added in your framework to use Rest Assured libraries.

7. Conclusion

Rest Assured is among the most popular tools used for automating Rest API with a base level knowledge of supported programming language.

If customized properly as per requirement then it can be used for all kind of Rest Services.