

Handling the Test Data and Executing It in Multiple Environments for User Module of an Application

Project Setup and Configuration

1. Create a new Maven project in Eclipse.
2. Configure the project to use Java 1.8.
3. Set up the project structure with feature files, step definitions, and supporting Java classes.

Add Dependencies

1. Add TestNG dependencies to the project's "pom.xml" file.
2. Configure Maven to handle dependencies.

```
<dependencies>
  <dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>5.3.0</version>
    <scope>test</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured-common -->
  <dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured-common</artifactId>
    <version>5.3.0</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured-all -->
  <dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured-all</artifactId>
    <version>5.3.0</version>
    <scope>test</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/io.rest-assured/json-path -->
  <dependency>
    <groupId>io.rest-assured</groupId>
```

```
<artifactId>json-path</artifactId>

<version>5.3.0</version>

<scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/org.json/json -->
<dependency>

  <groupId>org.json</groupId>

  <artifactId>json</artifactId>

  <version>20180813</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.hamcrest/hamcrest -->
<dependency>

  <groupId>org.hamcrest</groupId>

  <artifactId>hamcrest</artifactId>

  <version>2.2</version>

  <scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
<dependency>

  <groupId>com.fasterxml.jackson.core</groupId>

  <artifactId>jackson-databind</artifactId>

  <version>2.15.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>

  <groupId>org.testng</groupId>

  <artifactId>testng</artifactId>

  <version>7.7.1</version>

  <scope>test</scope>

</dependency>

<dependency>

  <groupId>io.cucumber</groupId>

  <artifactId>cucumber-java</artifactId>

  <version>7.10.1</version>
```

```
</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
<dependency>
  <groupId>io.cucumber</groupId>
  <artifactId>cucumber-junit</artifactId>
  <version>7.10.1</version>
  <scope>compile</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
<dependency>
  <groupId>io.cucumber</groupId>
  <artifactId>cucumber-core</artifactId>
  <version>7.10.1</version>
</dependency>
</dependency>
</dependencies>
```

Feature File Creation

1. Write Gherkin feature files for user registration, login, and user details.
2. Define scenarios with Given-When-Then steps to interact with the specified endpoints.

Step Definitions

- Create step definition classes in Java corresponding to the steps in your feature files. Implement Java methods to interact with the web services and perform data validation

- ▼ API Test Reques
 - src/main/java
 - ▼ src/test/java
 - ▼ definition.reqres.in
 - > reqres.java
 - ▼ feature.reqres.in
 - reqres.feature
 - ▼ runner.reqres.in
 - > testrunner.java
 - > JRE System Library [JavaSE-1.7]
 - > Maven Dependencies
 - > src
 - > target
 - > test-output-thread
 - pom.xml

```

reqres.java ×
1 package definition.reqres.in;
2 import java.io.ObjectInputFilter.Config;
12 public class reqres {
13     @Given("User send a Post request to create a user and validates status")
14     public void user_send_a_post_request_to_create_a_user_and_validates_status() {
15
16         JSONObject body = new JSONObject();
17         body.put("name", "meghna");
18         body.put("job", "developer");
19
20         RestAssured.given()
21             .baseUrl("https://reqres.in")
22             .basePath("/api/users")
23             .contentType(ContentType.JSON)
24             .body(body.toString())
25             .when().post()
26             .then().statusCode(201).log().ifError(); // log if there an error
27
28     }
29
30
31
32     @Given("User sends a Get request to get a user and validates status")
33     public void user_sends_a_get_request_to_get_a_user_and_validates_status() {
34
35         JSONObject body = new JSONObject();
36         body.put("email", "meghna@gmail.com");
37         body.put("password", "pasl23");
38         RestAssured.given().baseUrl("https://reqres.in")
39             .contentType(ContentType.JSON) .body(body.toString())
40             .when().post("/api/register")
41             .then().statusCode(400);
42
43     }
44
45
46
47     @Given("User sends a get request to get list of users and validates status")
48     public void user_sends_a_get_request_to_get_list_of_users_and_validates_status() {
49
50         RestAssured.given()
51             .baseUrl("https://reqres.in")
52             .basePath("/api/unknown")
53             .when().get()
54             .then().statusCode(200).log().all();
55     }
56
57 }

```

The screenshot displays the Eclipse IDE interface with three tabs: `reqres.java`, `reqres.feature`, and `testrunner.java`.

reqres.feature:

```
1 Feature: Implement The Lesson End Project
2
3 Scenario: Rest API testing on reqres.in
4   Given User send a Post request to create a user and validates status
5   Given User sends a Get request to get a user and validates status
6   Given User sends a get request to get list of users and validates status
```

testrunner.java:

```
1 package runner.reqres.in;
2 import org.junit.runner.RunWith;
3
4 @RunWith(Cucumber.class)
5 @CucumberOptions(features="src\\test\\java\\feature\\reqres\\in\\reqres.feature",
6                   glue={"definition.reqres.in"},
7
8                   plugin= {"html:target/Cucumberreport.html",
9                           "pretty",
10                          "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:",
11                          "timeline:test-output-thread/"})
12
13 public class testrunner {
14 }
15
16
17
18
```

Console Output:

```
Finished after 4.907 seconds
Runs: 1/1 Errors: 0 Failures: 0
> runner.reqres.in.testrunner [Runner: JUnit 4] (4.623 s)
Failure Trace
Scenario: Rest API testing on reqres.in
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Given User send a Post request to create a user and validates status # definition.reqres.in.reqres.user_send_a_post_request_to_crea
Given User sends a Get request to get a user and validates status # definition.reqres.in.reqres.user_send_a_get_request_to_get_
HTTP/1.1 200 OK
Date: Thu, 09 Nov 2023 15:04:42 GMT
Content-Type: application/json; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Report-To: {"group":"heroku-nel","max_age":3600,"endpoints":[{"url":"https://nel.heroku.com/reports?ts=1698558123&sid=c4c9725f-1ab0-44d8-
Reporting-Endpoints: heroku-nel=https://nel.heroku.com/reports?ts=1698558123&sid=c4c9725f-1ab0-44d8-820f-430df2718e11s=tntMYChrvMsoeufec
Nel: {"report_to":"heroku-nel","max_age":3600,"success_fraction":0.005,"failure_fraction":0.005,"response_headers":["Via"]}
X-Powered-By: Express
Access-Control-Allow-Origin: *
Etag: W/"2c1-N68qerxqu2kgqL5IKig4x0R8"
Via: 1.1 vegur
Cache-Control: max-age=14400
CF-Cache-Status: HIT
Age: 1712
Vary: Accept-Encoding
Server: cloudflare
CF-RAY: 8236f3236829604d-SIN
Content-Encoding: gzip
{
  "page": 1,
  "per_page": 6,
  "total": 12,
  "total_pages": 2,
  "data": [
    {
      "id": 1,
      "name": "cerulean",
      "year": 2000,
      "color": "#98FB98",
      "pantone_value": "15-4020"
    },
    {
      "id": 2,
```

GitHub repository : <https://github.com/Karthick-Office/Project01.git>

Handling the Test Data and Executing It in Multiple Environments for User Module of an Application Using REST Assured is present under the “REST Assured Practice Project” folder in the My GitHub repository