Writing Rest API Tests

The VeriSoft framework easily supports writing API tests. We implement API testing using RestAssured .

In writing API tests we need to mention numerous components:

- Project Lombok - Project Lombok is a Java library that automatically plugs into your editor and, using annotations automatically writes boilerplate code.
- RestAssured - RestAssured is a Java package, with abilities to test and validate Rest APIs
- Jackson - Jackson is a very popular and efficient Java-based library to serialize or map Java objects to JSON and vice versa.
- The builder design pattern - Builder is a creational design pattern, which allows the construction of complex objects step by step

The Example API

The API we are going to use as an example is taken from the website

https://dummy.restapiexample.com

In the following example, we will examine the get all employees API. The result of the API is a JSON file in the following way:

```
{
"status": "success",
"data": [
    {
    "id": "1",
    "employee_name": "Tiger Nixon",
    "employee_salary": "320800",
    "employee_age": "61",
    "profile_image": ""
    },
    ....
    ]
}
```

Our test will call the get all employees API, deserialize the JSON response and make sure there is at lease one employee

The Test

Here is the test case code:

```
@Test
public void getAllEmployees() throws JsonProcessingException {

    // Create the API request
    ApiRequest apiRequest = ApiRequest.builder()
            .endpoint("https://dummy.restapiexample.com")
            .method(Method.GET.toString())
            .build();

    // Build a RestAssured request
    RestAssured.baseURI = apiRequest.getEndpoint();
    RequestSpecification request = prepareRequest(apiRequest);

    // Send the request and get the response
    Response response = request.get("/api/v1/employees");
```

```
    ObjectMapper objectMapper = new ObjectMapper();
    EmployeesWrapper employeesWrapper =
      objectMapper.readValue(response.getBody().asString(), EmployeesWrapper.class);

    // Assert that the response is successful and the response body is not empty
    Assertions.assertTrue(!employeesWrapper.data.isEmpty());
}
```

We use the builder pattern to create an API request. The @Data and @Builder annotations are annotations from Project Lombok. The API request has the following structure:

```
@Data
@Builder
public class ApiRequest {
    private String endpoint;
    private String method;
    private String body;
    private String headers;
    private String authToken;
}
```

On line 12 we use a private util method to prepare the request:

- Create a new RestAssured object
- Add the appropriate headers and authorization
- Returns the created RestAssured object

Here is the implementation of the method:

```
private RequestSpecification prepareRequest(ApiRequest apiRequest) {
    RequestSpecification request = RestAssured.given();
    request.header("Content-Type", ContentType.XML);
    request.header("Accept", ContentType.XML);
    if (apiRequest.getAuthToken() != null) {
        request.header("Authorization", "Bearer " + apiRequest.getAuthToken());
    }
    if (apiRequest.getBody() != null) {
        request.body(apiRequest.getBody());
    }
    return request;
}
```

Next in the test, we call the Rest API in line 15:

```
Response response = request.get("/api/v1/employees");
```

And now the magic happens - we use the Jackson library in order to deserialize the JSON response:

```
ObjectMapper objectMapper = new ObjectMapper();
    EmployeesWrapper employeesWrapper =
      objectMapper.readValue(response.getBody().asString(), EmployeesWrapper.class);
```

And the JSON is packed into the EmployeeWrapper object.

The EmployeeWrapper object

We unfold the JSON object and use JSON to POJO website to create the class structure. You can find the full class structure here.