Example - Integrate Allure Report

In this lesson, we will understand the integration of Allure in a test.

We'll cover the following

- Attaching request and response in allure report
  - Rest Assured
    - Gradle
    - Maven
  - Spring WS SOAP client

Attaching request and response in allure report

In addition to the already-discussed dependencies for Allure like `allure-testng`, which adds every step of the test execution to the report, we can customize the report further and add a few additional capabilities to the report, like attaching the request and response.

`Rest Assured`

The following dependency will help to attach the requests and responses to the report while using `Rest Assured`:

Gradle

```
compile 'io.qameta.allure:allure-rest-assured:2.13.2'
```

Maven

```
<dependency>
    <groupId>io.qameta.allure</groupId>
    <artifactId>allure-rest-assured</artifactId>
    <version>2.13.2</version>
</dependency>
```

With this dependency being added, we need to configure a `Rest Assured` client to log the requests and responses so that the `allure-rest-assured` library can attach that to the report.

It can be configured in either of the following two ways:

- **Globally**

  To intercept all the requests and responses and attach it to the `Allure` report, the following piece of code can be added anywhere in our test code:

  ```
  static {
      RestAssured.filters(new io.qameta.allure.restassured.AllureRestAssured
  ());
  }
  ```

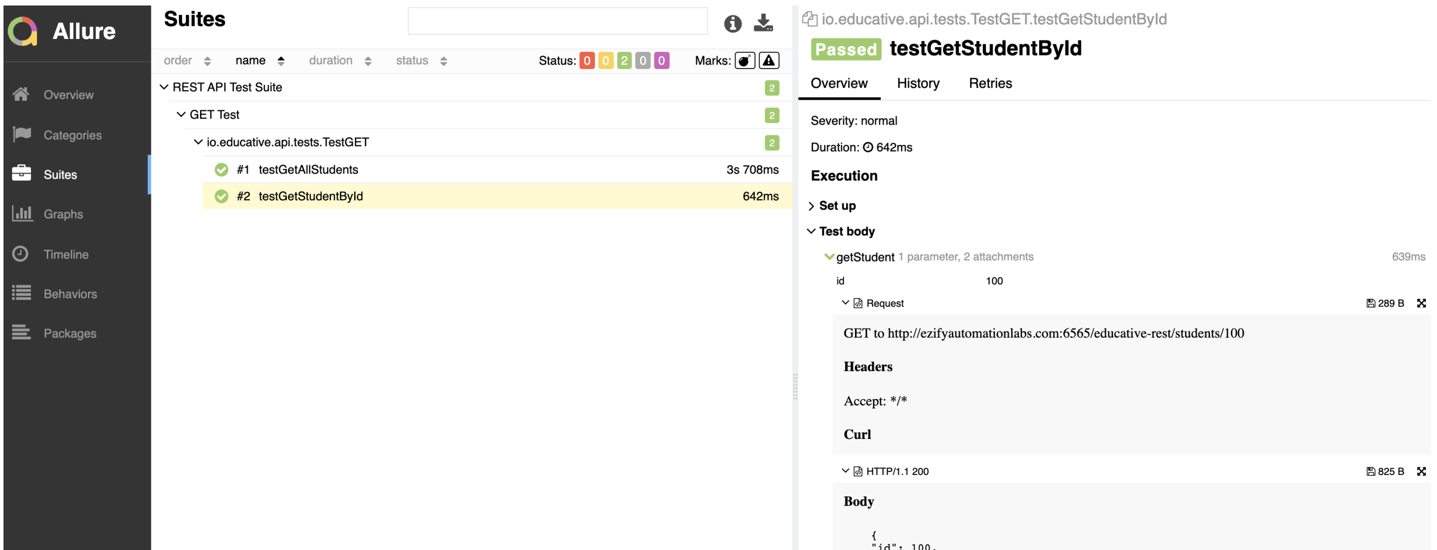
- **Only for certain requests**

  We can configure a certain API's requests and responses to reports by adding `filter(...)`. This will ensure that only this API's request and response are logged.

  ```
  @Step
  public void getAllStudents() {

      Response response = RestAssured.given()
              .filter(new io.qameta.allure.restassured.AllureRestAssured())
              .get("http://ezifyautomationlabs.com:6565/educative/students")
              .andReturn();
      assertEquals(response.getStatusCode(), HttpStatus.SC_OK, "http status"
  );
  }
  ```

The generated report will have both the requests and responses attached to it and look something like so:



REST API Test Suite 2
GET Test 2
io.educative.api.tests.TestGET 2
#1 testGetAllStudents 3s 708ms
#2 testGetStudentById 642ms

io.educative.api.tests.TestGET.testGetStudentById
Passed testGetStudentById

Severity: normal
Duration: 642ms
Execution
Set up
Test body
getStudent 1 parameter, 2 attachments 639ms
id 100
Request 289 B
GET to http://ezifyautomationlabs.com:6565/educative-rest/students/100
Headers
Accept: */*
Curl
HTTP/1.1 200 825 B
Body

```
{
"id": 100,
```

```
    "first_name": "John",
    "last_name": "Doe",
    "gender": "male"
}
```

Spring WS SOAP client

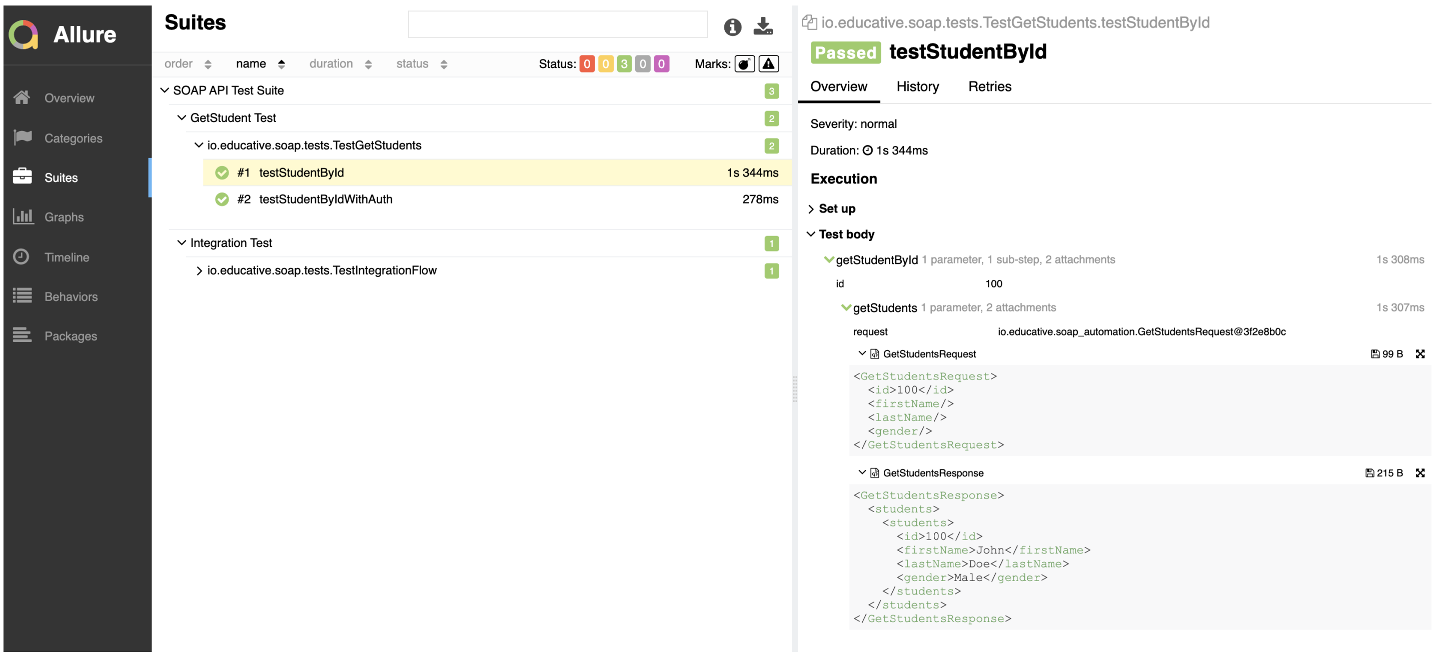
Unlike `Rest Assured`, here, we do not have any library that automatically attaches the requests and responses to the report. We have to do it ourselves.

```
@Step
public GetStudentsResponse getStudents(GetStudentsRequest request) {
    addAttachment(request);
    GetStudentsResponse response = (GetStudentsResponse) webServiceTemplate.marshalSendAndReceive(SERVICE_URL,
            request, getAuthRequestCallback());
    addAttachment(response);
    return response;
}

// add attachment to allure report
private void addAttachment(Object obj) {
    try {
    String xml = new XmlMapper().writerWithDefaultPrettyPrinter().writeValueAsBytes(obj);
    Allure.getLifecycle().addAttachment(obj.getClass().getSimpleName(), "text/xml", ".xml", xml);
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
}
```

In the code above, we can see the `GetStudentsRequest` object being added to report using the `addAttachment` method and the same is used for adding `GetStudentsResponse` after we receive the response. `XmlMapper` is used to convert the Java object to XML.

The generated report will have both the requests and responses attached to it and look something like so:



Suites

SOAP API Test Suite 3
GetStudent Test 2
io.educative.soap.tests.TestGetStudents 2
#1 testStudentById 1s 344ms
#2 testStudentByIdWithAuth 278ms
Integration Test 1
io.educative.soap.tests.TestIntegrationFlow 1

io.educative.soap.tests.TestGetStudents.testStudentById

Passed testStudentById

Severity: normal

Duration: 1s 344ms

Execution

Set up

Test body

getStudentById 1 parameter, 1 sub-step, 2 attachments 1s 308ms

id 100

getStudents 1 parameter, 2 attachments 1s 307ms

request io.educative.soap_automation.GetStudentsRequest@3f2e8b0c

GetStudentsRequest 99 B

```
<GetStudentsRequest>
  <id>100</id>
  <firstName/>
  <lastName/>
  <gender/>
</GetStudentsRequest>
```

GetStudentsResponse 215 B

```
<GetStudentsResponse>
  <students>
    <students>
      <id>100</id>
      <firstName>John</firstName>
      <lastName>Doe</lastName>
      <gender>Male</gender>
    </students>
  </students>
</GetStudentsResponse>
```

Note:

- Please note that the annotation `@Step` indicating the call to this method will be added as a step in the report.
- All the annotation-based things (`@Step`, `@Attachment`), apart from the ones programmatically added using `Allure.getLifeCycle().add...`, will only work when run from the command line, as `Allure` needs the `aspectjweaver` library to be attached as a Java agent when launching JVM. When running your code from IDEs (like Eclipse, IntelliJ, etc.) this does not happen.

In the next chapter, we will learn about using Postman and SoapUI to manually test REST API and SOAP API.