File Uploads

In this lesson, we will see how to handle file uploads using Rest Assured.

We'll cover the following

What is file upload?

There are use cases where you need to send a file in the message body with a `POST` or `PUT` *request*.

In the example below, we will see a `POST` *request* that takes a file stream as an input and creates a list of `Student`.

Example: Upload a JSON file to create a list of studentm

- HTTP Method: **POST**
- Target URL: `http://ezifyautomationlabs.com:6565`
- Resource path: `/educative-rest/students/upload`
- Message body: `multipart/form-data`

Take a look at the code below:

```
import static org.testng.Assert.assertEquals;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.testng.annotations.Test;

import io.restassured.RestAssured;
import io.restassured.response.Response;

public class APIDemo {

        private static final Logger LOG = LoggerFactory.getLogger(APIDemo.class);
        @Test
        public void fileUpload() throws IOException {


                String url = "http://ezifyautomationlabs.com:6565/educative-rest/students/upload";

        // creating JSON content to write to file
                String json = "[{\"first_name\":\"Sam\",\"last_name\":\"Bailey\",\"gender\":\"Fema

        // write the JSON string to File
                File file = new File("students.json");
                Files.write(file.toPath(), json.getBytes());

        // make api call to create list of `Student`
                Response response = RestAssured.given()
                                .multiPart("file", file)
                                .log().all()
                                .post(url)
                                .thenReturn();

                // validate the http status code of the response
                assertEquals(response.getStatusCode(), 201, "http status code");

                LOG.info("response body => {}", response.getBody().prettyPrint());
        }
}
```

Running the code above returns the following response:

```
[
    {
        "id": 103,
        "first_name": "Sam",
        "last_name": "Bailey",
        "gender": "Female"
    },
    {
        "id": 104,
        "first_name": "Sam",
        "last_name": "Hudson",
        "gender": "Male"
    }
]
```

> **Note:** The ids in the code snippet might be different from the results of the actual code.

Let's understand this example code

The code above uses the `TestNG` and `RestAssured` libraries to automate the HTTP POST file upload API and creates a list of `Student`.

- **Step 1** – we create the JSON file to be uploaded using the following code:

```
String json = "[{\"first_name\":\"Sam\",\"last_name\":\"Bailey\",\"gender
\":\"Female\"},{\"first_name\":\"Sam\",\"last_name\":\"Hudson\",\"gender
\":\"Male\"}]";

File file = new File("students.json");
Files.write(file.toPath(), json.getBytes());
```

- **Step 2** – we create a `multipart` entity with the file object. The control name of the `multipart` entity is ***file*** using `Rest Assured` constructs like the following:

```
// make api call to create list of `Student`
Response response = RestAssured.given()
    .multiPart("file", file)
    .log().all()
    .post(url)
    .thenReturn();
```

  The multipart does not have to be **file**. It can be of any type and `Rest Assured` provides us with overloaded methods to add any type of `multipart` data.

```
// make api call to create list of `Student`
Response response = RestAssured.given()
    .multiPart("file", file)
    .multipart("name", "Educative")
    .log().all()
    .post(url)
    .thenReturn();
```

- **Step 3** – we verify the HTTP status code to ensure the creation is successful

```
assertEquals(response.getStatusCode(), 201, "http status code");
```

  Furthermore, we can make a `GET` API call to ensure it is created successfully.

In the next lesson, we will learn how to download a file using the `Rest Assured` library.