What is Payload and body() in Rest Assured?

In the context of API automation testing using Rest Assured in Java, the **payload** refers to the actual data or content that is sent in the body of an HTTP request. When you make a request to a server (especially in POST, PUT, or PATCH requests), you often need to send some data. This data is called the payload. It can be in various formats like JSON, XML, or plain text.

The **body()** method in Rest Assured is used to set the payload in the request body. It is commonly used when sending data to an API endpoint, such as creating a new resource (using POST) or updating an existing resource (using PUT).

Different body() Methods in Rest Assured

The body() method in Rest Assured comes with several overloaded versions that allow you to pass the payload in different ways. Here are the different methods with their parameters and return types:

1. **body(String body)**
   - **Parameter:** A string representing the raw body content.
   - **Return Type:** RequestSpecification
   - **Example:** .body("{\"name\":\"John\",\"age\":30}")
2. **body(Object object)**
   - **Parameter:** An object that will be serialized to JSON or XML.
   - **Return Type:** RequestSpecification
   - **Example:** .body(new Person("John", 30)) (Assuming Person is a POJO)
3. **body(byte[] bytes)**
   - **Parameter:** A byte array representing the body content.
   - **Return Type:** RequestSpecification
   - **Example:** .body("some text".getBytes())
4. **body(File file)**
   - **Parameter:** A file containing the body content.
   - **Return Type:** RequestSpecification
   - **Example:** .body(new File("path/to/file.json"))
5. **body(InputStream inputStream)**

- **Parameter:** An InputStream representing the body content.
- **Return Type:** RequestSpecification
- **Example:** .body(new FileInputStream("path/to/file.json"))

6. **body(JsonNode node)**
   - **Parameter:** A JsonNode object from the Jackson library.
   - **Return Type:** RequestSpecification
   - **Example:** .body(jsonNode)
7. **body(Object object, ObjectMapper mapper)**
   - **Parameter:** An object to be serialized and an ObjectMapper instance.
   - **Return Type:** RequestSpecification
   - **Example:** .body(person, new ObjectMapper())
8. **body(Object object, ObjectMapperType mapperType)**
   - **Parameter:** An object to be serialized and an ObjectMapperType enumeration (e.g., ObjectMapperType.JACKSON_2).
   - **Return Type:** RequestSpecification
   - **Example:** .body(person, ObjectMapperType.JACKSON_2)
9. **body(Object object, Class<?> objectType)**
   - **Parameter:** An object and its type to be serialized.
   - **Return Type:** RequestSpecification
   - **Example:** .body(person, Person.class)

Example 1: Sending a Simple JSON String

```
given()
   .header("Content-Type", "application/json")
   .body("{\"name\":\"Mike\",\"age\":25}")
.when()
   .post("/createUser")
.then()
```

```
    .statusCode(201);
```

Explanation:

- **Step 1:** We use the body() method to send a simple JSON string as the payload.
- **Step 2:** The header() method sets the content type to application/json.
- **Step 3:** The post() method sends the request to the /createUser endpoint.
- **Step 4:** The statusCode(201) assertion verifies that the user was created successfully.

Example 2: Sending a Java Object (POJO) as Payload

```
Person person = new Person("Mahii", 30);


given()
    .header("Content-Type", "application/json")
    .body(person)
.when()
    .post("/createUser")
.then()
    .statusCode(201);
```

Explanation:

- **Step 1:** We create a Person object with name "Mahii" and age 30.
- **Step 2:** The body() method serializes the Person object into JSON.
- **Step 3:** The request is sent to the /createUser endpoint.
- **Step 4:** We check that the server responds with a 201 status code.

Example 3: Sending a Byte Array

```
byte[] payload = "{\"name\":\"Sweta\",\"age\":28}".getBytes();


given()
    .header("Content-Type", "application/json")
    .body(payload)
```

```
.when()
   .post("/createUser")
.then()
   .statusCode(201);
```

Explanation:

- **Step 1:** Convert the JSON string into a byte array.
- **Step 2:** The body() method sends the byte array as the payload.
- **Step 3:** The rest of the process remains the same as in the previous examples.

Example 4: Sending Payload from a File

```
File jsonFile = new File("path/to/user.json");


given()
   .header("Content-Type", "application/json")
   .body(jsonFile)
.when()
   .post("/createUser")
.then()
   .statusCode(201);
```

Explanation:

- **Step 1:** We create a File object representing the JSON file.
- **Step 2:** The body() method reads the content from the file and sends it as the payload.
- **Step 3:** We verify that the user creation is successful.

Example 5: Sending InputStream as Payload

```
InputStream jsonStream = new FileInputStream("path/to/user.json");


given()
   .header("Content-Type", "application/json")
```

```
    .body(jsonStream)
.when()
    .post("/createUser")
.then()
    .statusCode(201);
```

Explanation:

- **Step 1:** Create an InputStream from the JSON file.
- **Step 2:** The body() method sends the content of the stream as the payload.
- **Step 3:** Similar to the file example, we check for a successful response.

Example 6: Sending JsonNode as Payload

```
ObjectMapper mapper = new ObjectMapper();
JsonNode jsonNode = mapper.createObjectNode()
                         .put("name", "Mike")
                         .put("age", 25);


given()
    .header("Content-Type", "application/json")
    .body(jsonNode)
.when()
    .post("/createUser")
.then()
    .statusCode(201);
```

Explanation:

- **Step 1:** Create a JsonNode object using Jackson's ObjectMapper.
- **Step 2:** The body() method sends the JsonNode as the payload.
- **Step 3:** We proceed with the API request and validate the response.

Example 7: Sending Object with Custom ObjectMapper

```
Person person = new Person("Mahii", 30);
ObjectMapper customMapper = new ObjectMapper(); // You can configure it


given()
   .header("Content-Type", "application/json")
   .body(person, customMapper)
.when()
   .post("/createUser")
.then()
   .statusCode(201);
```

Explanation:

- **Step 1:** We use a custom-configured ObjectMapper to serialize the Person object.
- **Step 2:** The body() method serializes the object and sends it as the payload.
- **Step 3:** The request is sent and validated.

Example 8: Sending Object with ObjectMapperType

```
Person person = new Person("Sweta", 28);


given()
   .header("Content-Type", "application/json")
   .body(person, ObjectMapperType.JACKSON_2)
.when()
   .post("/createUser")
.then()
   .statusCode(201);
```

Explanation:

Step 1: Use the ObjectMapperType.JACKSON_2 to specify the serialization type.

Step 2: The body() method uses the specified mapper type to serialize and send the payload.

Step 3: We verify the response.

Example 9: Creating a New User

Objective: To create a new user by sending a JSON payload.

```
// Create a new user using JSON payload

String payload = "{ \"name\": \"Sweta\", \"job\": \"Lead QA\" }";


Response response = given()

    .contentType(ContentType.JSON)

    .body(payload)

    .when()

    .post("https://reqres.in/api/users")

    .then()

    .statusCode(201)

    .extract()

    .response();


System.out.println("User ID: " + response.jsonPath().getString("id"));
```

Explanation:

- The body() method is used to pass the JSON payload.
- The POST request is sent to create a new user, and the response is validated for a successful creation.

Example 10: Updating a User with POJO

Objective: To update an existing user's information using a POJO.

```
// Define a User class

class User {
    private String name;
    private String job;
    // constructors, getters, setters
}

// Update user information
User user = new User("Mahii", "Senior Developer");

Response response = given()
    .contentType(ContentType.JSON)
    .body(user)
    .when()
    .put("https://reqres.in/api/users/2")
    .then()
    .statusCode(200)
    .extract()
    .response();

System.out.println("Updated At: " + response.jsonPath().getString("updatedAt"));
```

Explanation:

- The body() method serializes the POJO into JSON format.
- The PUT request updates the user’s information, and the response is validated.

Example 11: Sending Binary Data

Objective: To send binary data as the payload.

```
// Sending binary data in the request body
byte[] payload = "Binary data".getBytes();
```

```
Response response = given()
    .contentType(ContentType.BINARY)
    .body(payload)
    .when()
    .post("https://api.example.com/upload")
    .then()
    .statusCode(200)
    .extract()
    .response();

System.out.println("Response: " + response.asString());
```

Explanation:

- The body() method sends the binary data as the payload.
- The POST request uploads the binary data, and the response is validated.

Example 12: Sending File Data

Objective: To send a file as the payload.

```
// Sending a file in the request body
File jsonFile = new File("path/to/file.json");

Response response = given()
    .contentType(ContentType.JSON)
    .body(jsonFile)
    .when()
    .post("https://reqres.in/api/users")
    .then()
    .statusCode(201)
    .extract()
```

```
   .response();


System.out.println("User ID: " + response.jsonPath().getString("id"));
```

Explanation:

- The body() method is used to send a file as the request body.
- The POST request creates a new user from the file’s content, and the response is validated.

Example 13: Using a Custom Object Mapper

Objective: To serialize a POJO with a custom object mapper.

```
// Define a custom object mapper

ObjectMapper customMapper = new CustomObjectMapper();


// Define the User object

User user = new User("Mike", "Project Manager");


Response response = given()

   .contentType(ContentType.JSON)

   .body(user, customMapper)

   .when()

   .post("https://reqres.in/api/users")

   .then()

   .statusCode(201)

   .extract()

   .response();


System.out.println("User ID: " + response.jsonPath().getString("id"));
```

Explanation:

- The body() method uses a custom object mapper for serializing the POJO.

- The POST request creates a new user, and the response is validated using the custom serialization.

These examples should give you a good understanding of how to use the body() method in different scenarios while working with API automation in Rest Assured. Each example is aimed at handling different types of payloads, whether it's a simple string, an object, a byte array, or a file.