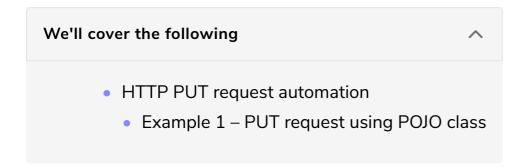
PUT Request

In this lesson, we will learn how to automate the PUT request to update an existing record.



HTTP PUT request automation

In this lesson, we will discuss updating an existing record using the **PUT** request method.

Example 1 – PUT request using POJO class

- HTTP Method: PUT
- Target URL: http://ezifyautomationlabs.com:6565
- Resource path: /educative-rest/students
- Message body: As a Java object
- Take a look at the code below:

```
import static org.testng.Assert.assertTrue;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.testng.annotations.Test;

import io.restassured.RestAssured;
import io.restassured.response.Response;
import com.fasterxml.jackson.annotation.JsonProperty;

public class PUTRequestTest {

    private static Logger LOG = LoggerFactory.getLogger(PUTRequestTest.class);

    @Test
    public void testPUTusingPOJO() {

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

        LOG.info("Step - 1 : Create a new Student [POST]");

        Student body = new Student("Ryan", "Jackson", "Male");
```

```
Response response = RestAssured.given()
                                 .header("accept", "application/json")
                                 .header("content-type", "application/json")
                                 .body(body)
                                 .post(url)
                                 .andReturn();
                LOG.info("Created Student Record");
                response.getBody().prettyPrint();
                String id = response.getBody().jsonPath().getString("id");
                LOG.info("Get the created Student ID: " + id);
                LOG.info("Step - 2 : Update Student's record [PUT]");
        Student bodyUpdate = new Student("John", "LP", "Male");
        bodyUpdate.id = Long.parseLong(id);
        String url1 = url + "/" + id;
        Response response1 = RestAssured.given()
                .header("accept", "application/json")
                .header("content-type", "application/json")
                .body(bodyUpdate)
                .put(url1)
                .andReturn();
        LOG.info("Step - 3 : Print the response message and assert the status");
        response1.getBody().prettyPrint();
        LOG.info("Status " + response.getStatusCode());
                assertTrue(response.getStatusCode() == 201);
        }
}
class Student {
    public Student(String firstName, String lastName, String gender) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.gender = gender;
    @JsonProperty("id")
    Long id;
    @JsonProperty("first_name")
    String firstName;
    @JsonProperty("last_name")
    String lastName;
    @JsonProperty("gender")
    String gender;
```





 \leftarrow

زا

Let's understand this example code.

HTTP PUT request and sends a Student id along with the message body which has to be updated.

• **Step 1** – creates a new student using POST request by sending a new Student object. The method post(url) is used for sending a new resource creation request along with headers using header(key,value) method and it returns a Response object using andReturn() method. Also, it fetches the created Student id (line no 34 in the code example).

• **Step 2** – makes a PUT request using {id} in the path param to update the data of the particular Student. We are also sending headers and a new message body in the request, which returns a Response object using the andReturn() method.

• **Step 3** – verifies that the Student's record is updated successfully

```
LOG.info("Step - 3 : Print the response message and assert the status");
response1.getBody().prettyPrint();
LOG.info("Status " +response getStatusCode());
```

assertTrue(response.getStatusCode()==201);

In the next lesson, we'll learn about deleting a record.