API Validation Testing Using Karate

Niculiță Cristiana Teodora, Grupa 344
Duca Cosmina Elena, Grupa 333
Voinea Ana-Maria, Grupa 333

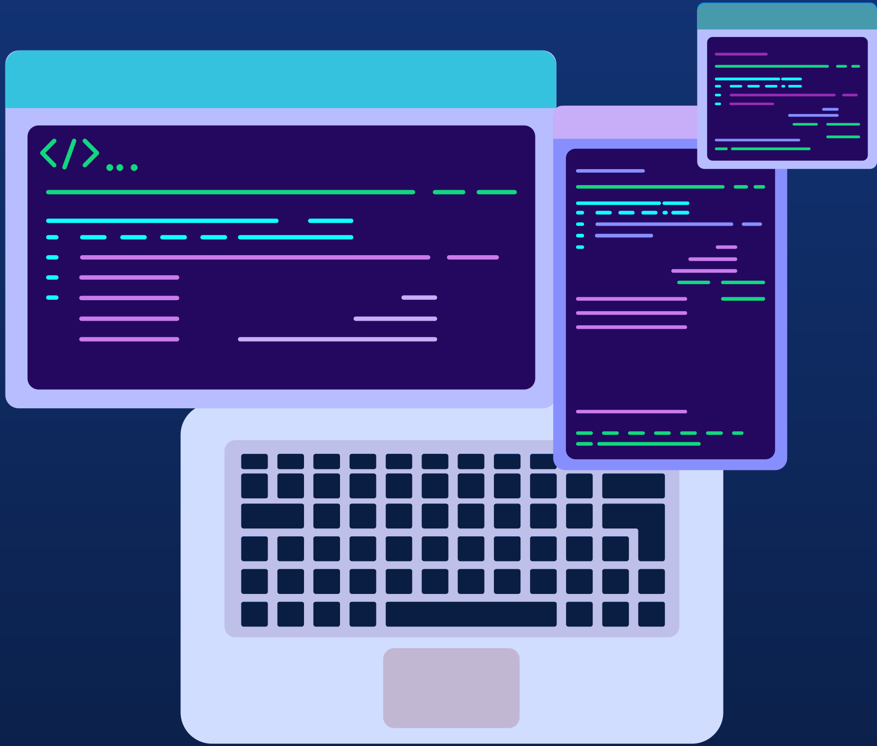


TABLE OF CONTENTS

01 Introduction

02 APIs

03 Automation and Manual Testing

04 Karate Framework

05 Cucumber Framework

06 Karate vs Cucumber

07 How Karate Works

08 Conclusion

01

Introduction

INTRODUCTION

The article we are studying is about the **Karate automation framework**, developed in 2017, which aims to reduce testing costs and increase time efficiency in development and deployment phases. It is designed for API testing, performance testing, and UI automation, offering a user-friendly language, powerful assertions, and built-in multithreading capabilities. We will analyze the development of Karate for **API testing**, evaluating its performance and comparing it with the Cucumber BDD framework.

Testing APIs

APIs or Endpoints serve as a message transfer link between back-end and front-end systems. When using APIs over a server, we need to ensure that the API is functioning as per contract.

- Validate the API returns the expected status codes.
  - Every API upon execution returns a Status code. Status codes in the range 200-299 depict a successful request. Status codes in 400-499 and 500-599 represent client and server-side errors respectively.
- Ensure API responses match predefined contracts.These contracts specify data formats, mandatory fields, and expected behaviors.
  - For example, we can specify in a response body that fields like Username and Email must be in string format, Age as an integer within a specific range, and that mandatory fields like Email must be included in every response.
- Regularly monitor API performance and functionality.

02

Karate Framework

- Karate is based on Cucumber, but doesn't require Java step definitions.
- It has a similar syntax, but it's easier to use and integrates well with other softwares. This makes it a simple, powerful and flexible option for API validation testing.
- Can handle complex schemas and quickly test endpoints using multiple threads. Its simple syntax helps developers write tests faster. This efficiency, along with parallel execution, makes Karate a suitable choice for regular testing on cloud platforms.

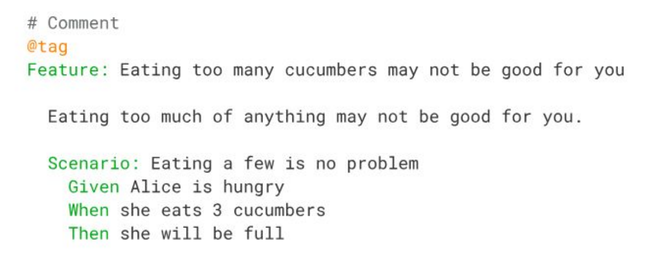
Karate Labs

03

Cucumber Framework

CUCUMBER FRAMEWORK

The Cucumber framework is a Behavior Driven Development (BDD) tool that uses natural language syntax to create test scripts. It helps product owners or business analysts to understand better what are the tests about. It follows a *given, when, then syntax*, which is written using natural language. It might looks very easy to understand, but behind those words, the developer must supply more definitions in the backend code in Java to automate these steps.



```
# Comment
@tag
Feature: Eating too many cucumbers may not be good for you

  Eating too much of anything may not be good for you.

  Scenario: Eating a few is no problem
    Given Alice is hungry
    When she eats 3 cucumbers
    Then she will be full
```

Figure 1. Cucumber test script

04

Karate VS Cucumber

CUCUMBER VS KARATE

FRAMEWORK	CUCUMBER	KARATE
Syntax	✗	✓
Backend Integration	✓	✗
Parallelism	✗	✓
Step Definitions	✓	✗

The tests using Karate framework has much many advantages than Cucumber framework.

CUCUMBER VS KARATE FRAMEWORK

Syntax and Language

- **Cucumber** tests are written in the Gherkin language, based on a schema like Given, When, Then to describe test scenarios in a human-readable format.
- **Karate** tests are written in a domain-specific language (DSL) inspired by Gherkin but with its syntax for expressing test scenarios. Karate allows you to write tests directly in the feature files, eliminating the need for separate step definition files.

Parallelism

- **Cucumber** itself doesn't provide built-in support for parallel test execution.
- **Karate** supports parallel execution of tests out-of-the-box. It automatically runs feature files concurrently, making it easy to execute tests in parallel without any additional configuration.

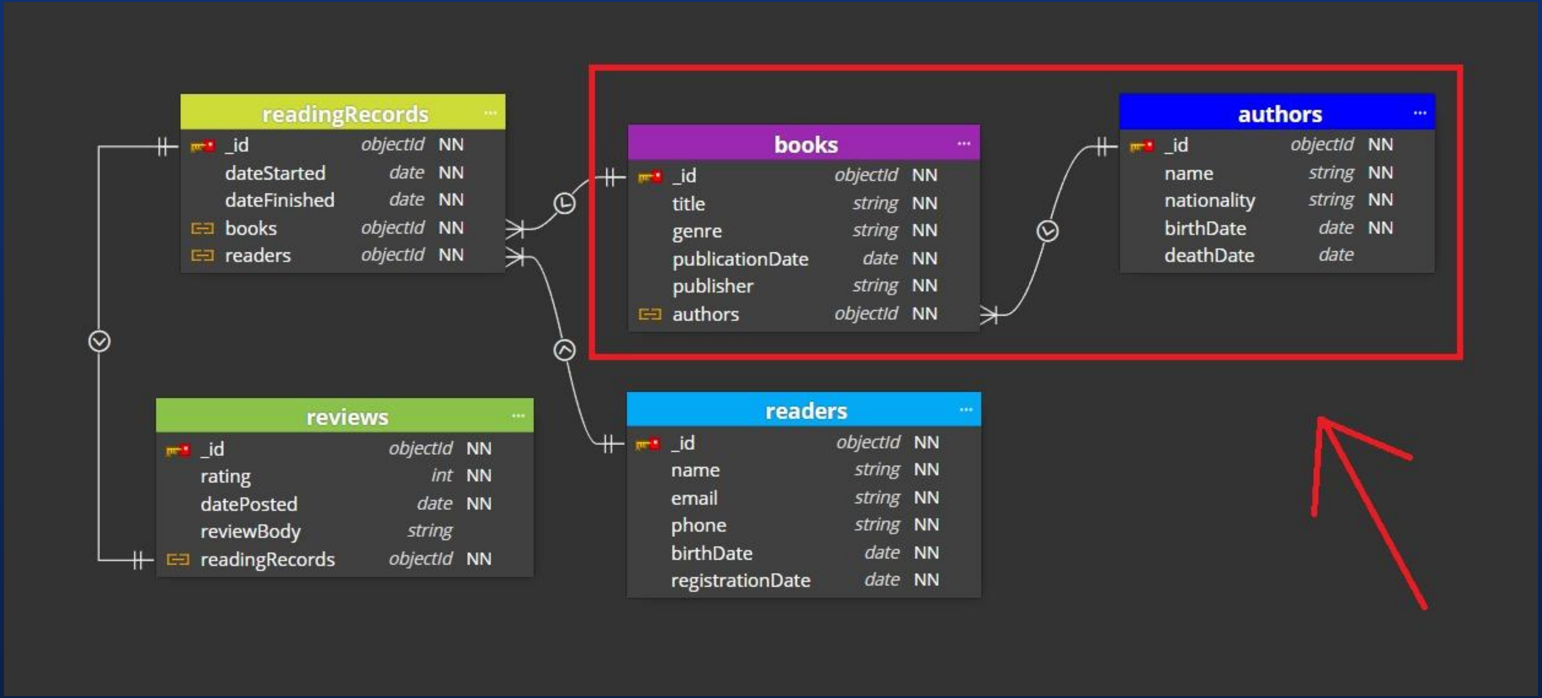
Step definition

- In **Cucumber**, step definitions are separate code files where the actual implementation of test steps is written in programming languages like Java.
- **Karate** eliminates the need for separate step definition files by allowing you to define test steps directly in the feature files.

05

Results & More

Diagram of the Tested Project



readingRecords

_id	objectId	NN
dateStarted	date	NN
dateFinished	date	NN
books	objectId	NN
readers	objectId	NN

books

_id	objectId	NN
title	string	NN
genre	string	NN
publicationDate	date	NN
publisher	string	NN
authors	objectId	NN

authors

_id	objectId	NN
name	string	NN
nationality	string	NN
birthDate	date	NN
deathDate	date	

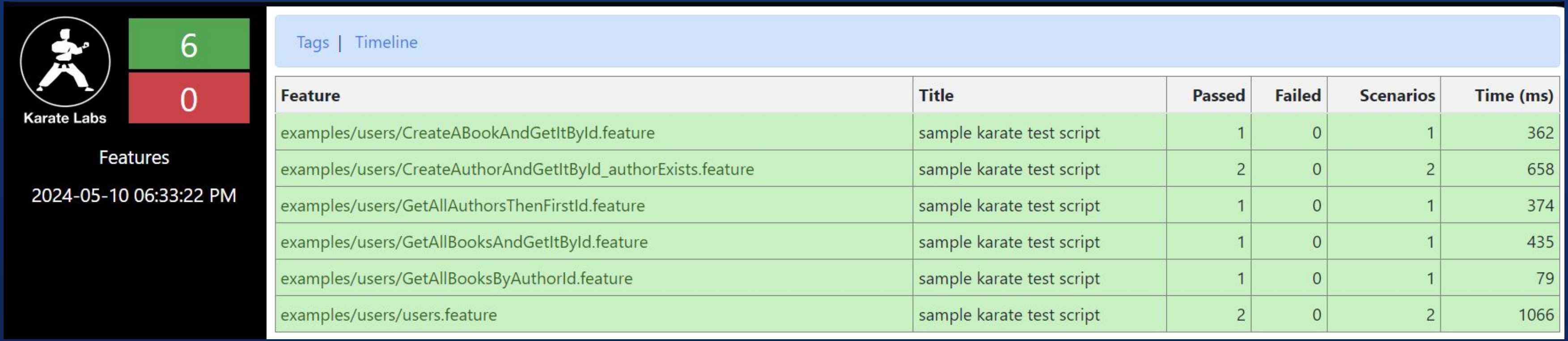
reviews

_id	objectId	NN
rating	int	NN
datePosted	date	NN
reviewBody	string	
readingRecords	objectId	NN

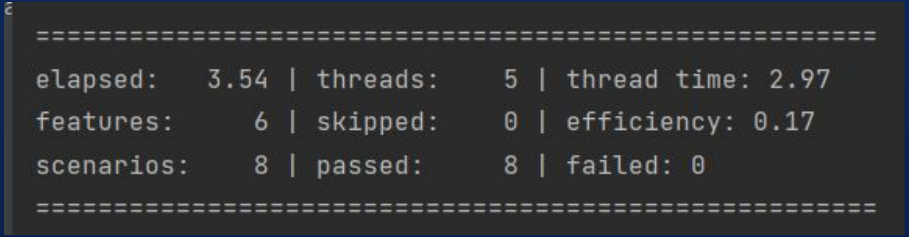
readers

_id	objectId	NN
name	string	NN
email	string	NN
phone	string	NN
birthDate	date	NN
registrationDate	date	NN

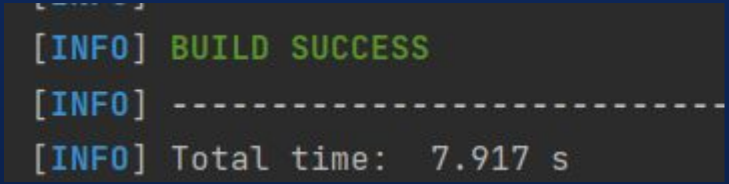
Karate results



Feature	Title	Passed	Failed	Scenarios	Time (ms)
examples/users/CreateABookAndGetItById.feature	sample karate test script	1	0	1	362
examples/users/CreateAuthorAndGetItById_authorExists.feature	sample karate test script	2	0	2	658
examples/users/GetAllAuthorsThenFirstId.feature	sample karate test script	1	0	1	374
examples/users/GetAllBooksAndGetItById.feature	sample karate test script	1	0	1	435
examples/users/GetAllBooksByAuthorId.feature	sample karate test script	1	0	1	79
examples/users/users.feature	sample karate test script	2	0	2	1066

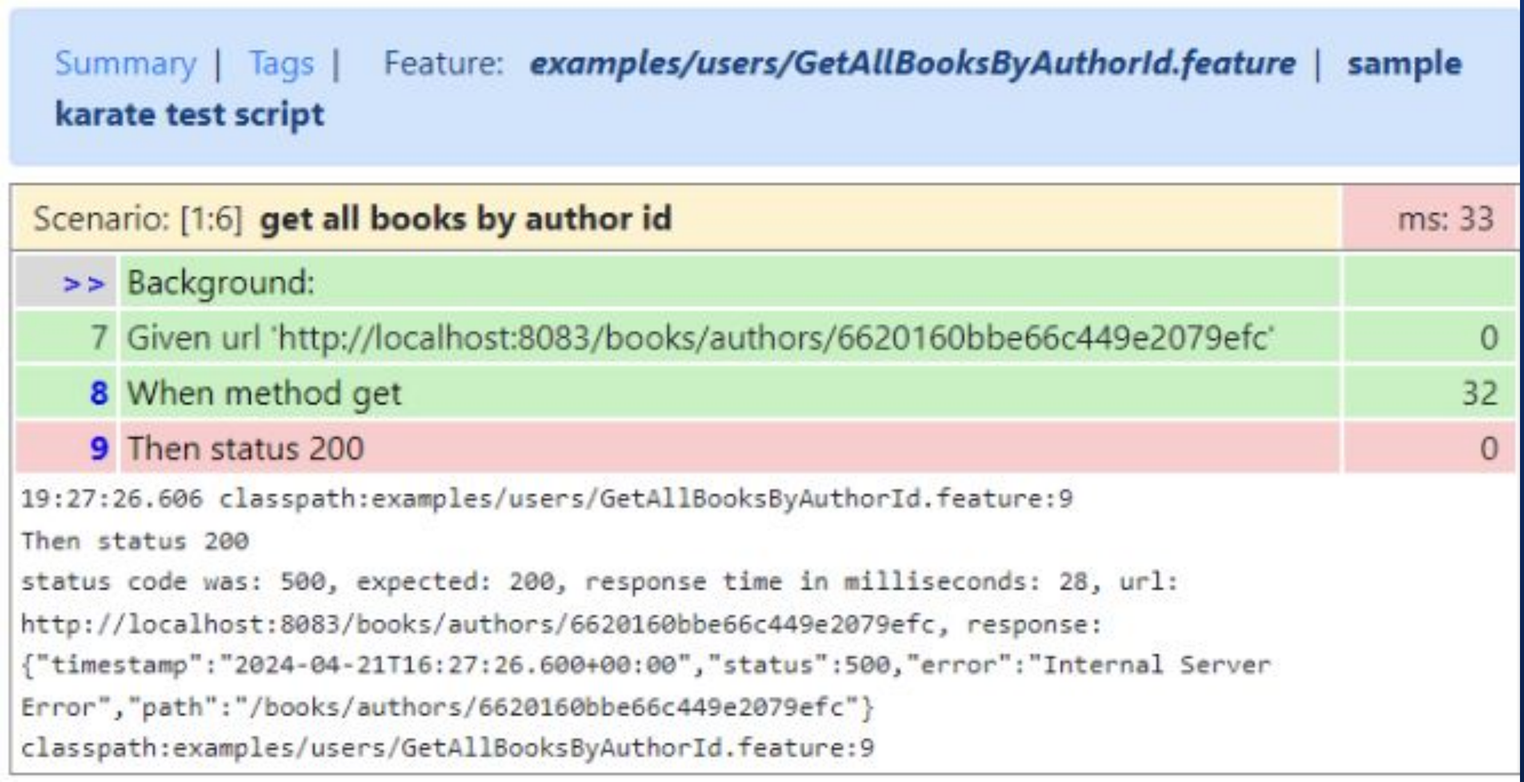


```
==================================================================
elapsed:   3.54 | threads:    5 | thread time: 2.97
features:     6 | skipped:    0 | efficiency: 0.17
scenarios:    8 | passed:     8 | failed: 0
==================================================================
```



```
[INFO] BUILD SUCCESS
[INFO] ------------------------------
[INFO] Total time:  7.917 s
```

Karate results

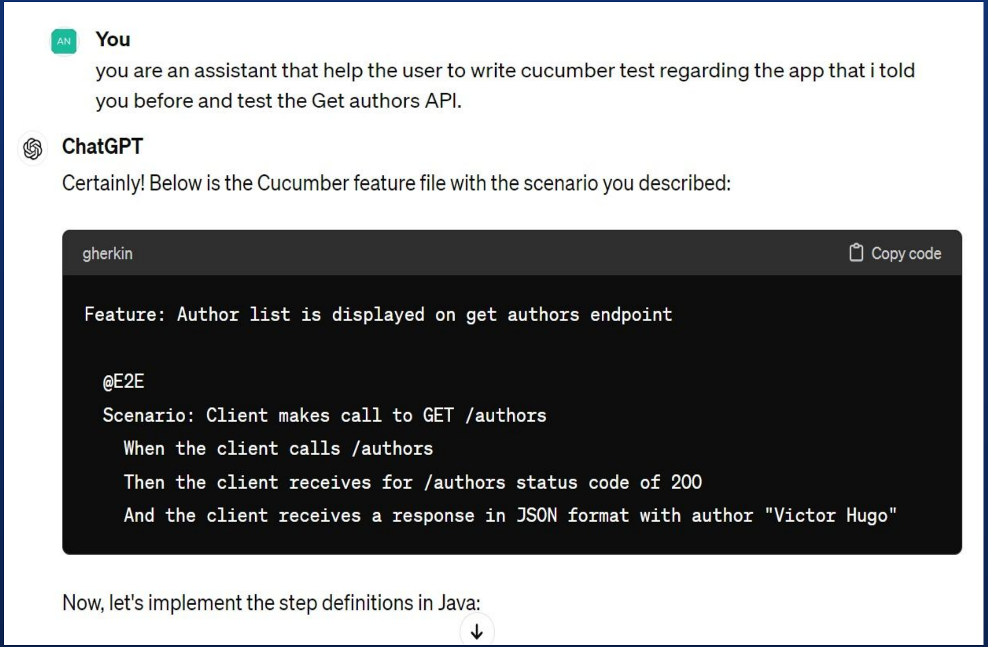


Summary | Tags | Feature: ***examples/users/GetAllBooksByAuthorId.feature*** | **sample karate test script**

Scenario: [1:6] **get all books by author id**		ms: 33
>>	Background:	
7	Given url 'http://localhost:8083/books/authors/6620160bbe66c449e2079efc'	0
8	When method get	32
9	Then status 200	0

```
19:27:26.606 classpath:examples/users/GetAllBooksByAuthorId.feature:9
Then status 200
status code was: 500, expected: 200, response time in milliseconds: 28, url:
http://localhost:8083/books/authors/6620160bbe66c449e2079efc, response:
{"timestamp":"2024-04-21T16:27:26.600+00:00","status":500,"error":"Internal Server
Error","path":"/books/authors/6620160bbe66c449e2079efc"}
classpath:examples/users/GetAllBooksByAuthorId.feature:9
```

Cucumber scenario generated by ChatGPT



You

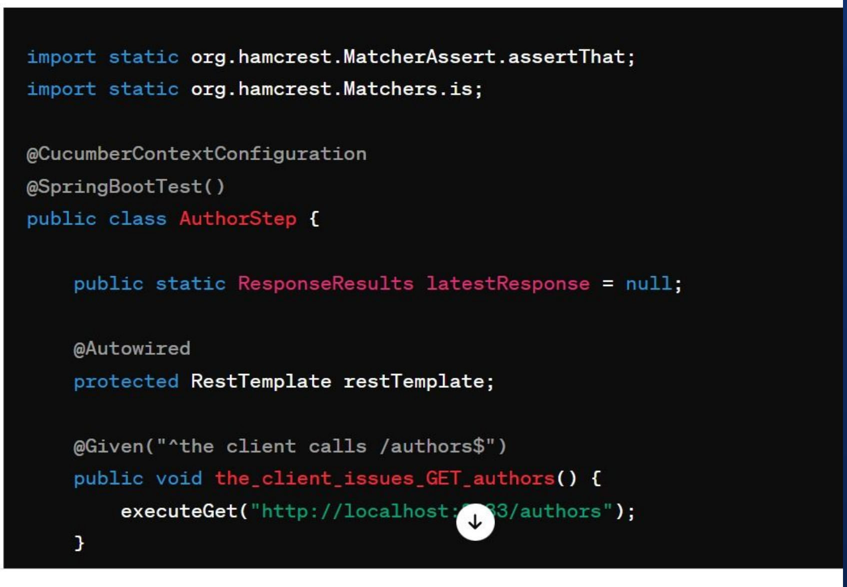
you are an assistant that help the user to write cucumber test regarding the app that i told you before and test the Get authors API.

ChatGPT

Certainly! Below is the Cucumber feature file with the scenario you described:

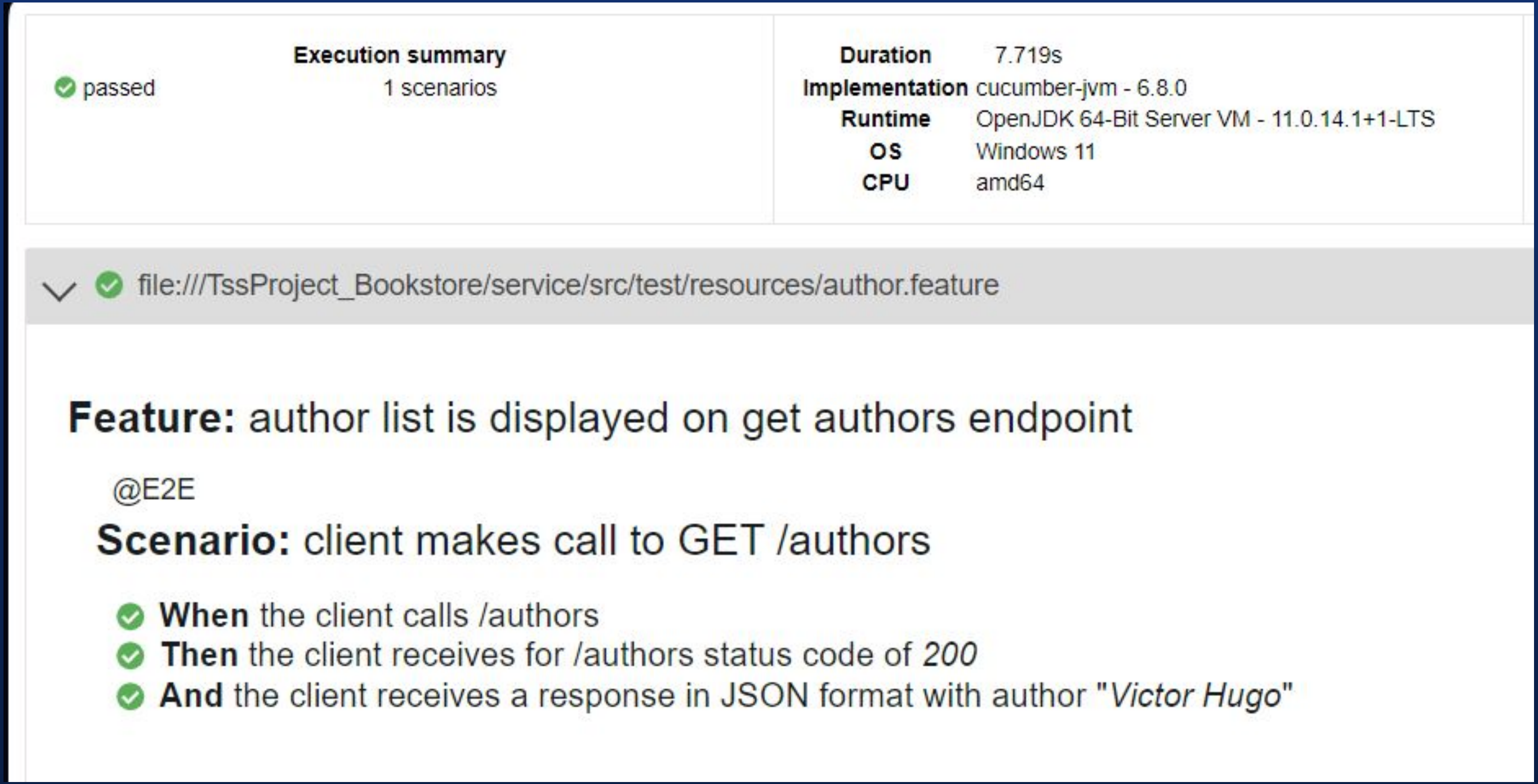
```gherkin
Feature: Author list is displayed on get authors endpoint

  @E2E
  Scenario: Client makes call to GET /authors
    When the client calls /authors
    Then the client receives for /authors status code of 200
    And the client receives a response in JSON format with author "Victor Hugo"
```

Now, let's implement the step definitions in Java:



```java
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.is;

@CucumberContextConfiguration
@SpringBootTest()
public class AuthorStep {

    public static ResponseResults latestResponse = null;

    @Autowired
    protected RestTemplate restTemplate;

    @Given("^the client calls /authors$")
    public void the_client_issues_GET_authors() {
        executeGet("http://localhost:83/authors");
    }
```

Cucumber Results



✓ passed

Execution summary
1 scenarios

Duration	7.719s
Implementation	cucumber-jvm - 6.8.0
Runtime	OpenJDK 64-Bit Server VM - 11.0.14.1+1-LTS
OS	Windows 11
CPU	amd64

✓ file:///TssProject_Bookstore/service/src/test/resources/author.feature

Feature: author list is displayed on get authors endpoint

@E2E

Scenario: client makes call to GET /authors

- ✓ **When** the client calls /authors
- ✓ **Then** the client receives for /authors status code of *200*
- ✓ **And** the client receives a response in JSON format with author "*Victor Hugo*"

06

Conclusion

MULTUMIM