Generate Spring boot REST API stubs using OpenAPI generator maven plugin

Tapan Adhikari · Follow
3 min read · Sep 18, 2022

50 3

One of the important aspects of microservices is inter-service communication. When making a call to another service, the caller should know the exact request and response. It's always better to have a contract between the provider and consumer of the APIs in the design phase itself.

In my project, I have used the OpenAPI specifications to have a contract between the providers and consumers. To ensure consistency, some parts of the code can be generated so that the developers need to focus only on the implementation part.

In this article i am going to explain how to generate REST API stubs using the OpenAPI generator maven plugin.

· · ·

The first step is to write the API specs. We can use the swagger editor to write the API specs and validate the same. For this example, I have created the specification document employee.yaml as below.

```yaml
openapi: 3.0.3
info:
  title: Employee - OpenAPI 3.0
  description: |-
    Employee API Spec
  version: 1.0.0
servers:
  - url: api/v1
paths:
  /employee:
    post:
      tags:
        - employee
      summary: Add a new employee
      description: Add a new employee
      operationId: addEmployee
      requestBody:
        description: Create a new employee
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Employee'
        required: true
      responses:
        '201':
          description: Created
          content:
            application/json:
              schema:
                type: string
        '400':
          description: Invalid request
    get:
      tags:
        - employee
      summary: Get all employees
      description: Get all employees
      operationId: getEmployees
      responses:
        '200':
          description: successful operation
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Employee'
        '400':
          description: Invalid request
components:
  schemas:
    Employee:
      type: object
      properties:
        id:
          type: integer
          format: int64
          example: 10
        name:
          type: string
          example: John
        age:
          type: integer
          format: int
          example: 25
```

I have added one POST API to add an employee and one GET API to get all the employees.

. . .

The next step is to create a spring boot project and add the OpenAPI generator maven plugin to the pom.xml. I have also specified the path to the API spec file.

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.openapitools</groupId>
            <artifactId>openapi-generator-maven-plugin</artifactId>
            <version>6.1.0</version>
            <executions>
                <execution>
                    <goals>
                        <goal>generate</goal>
                    </goals>
                    <configuration>
                        <inputSpec>
                            ${project.basedir}/src/main/resources/employee.yaml
                        </inputSpec>
                        <generatorName>spring</generatorName>
                        <apiPackage>org.SwaggerCodeGenExample.api</apiPackage>
                        <modelPackage>org.SwaggerCodeGenExample.model</modelPackage>
                        <configOptions>
                            <interfaceOnly>true</interfaceOnly>
                        </configOptions>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

pom.xml hosted with ❤ by GitHub   view raw

I have only added a few basic configs. You can refer to the full list here.

We must add a few dependencies for the generated code to compile.

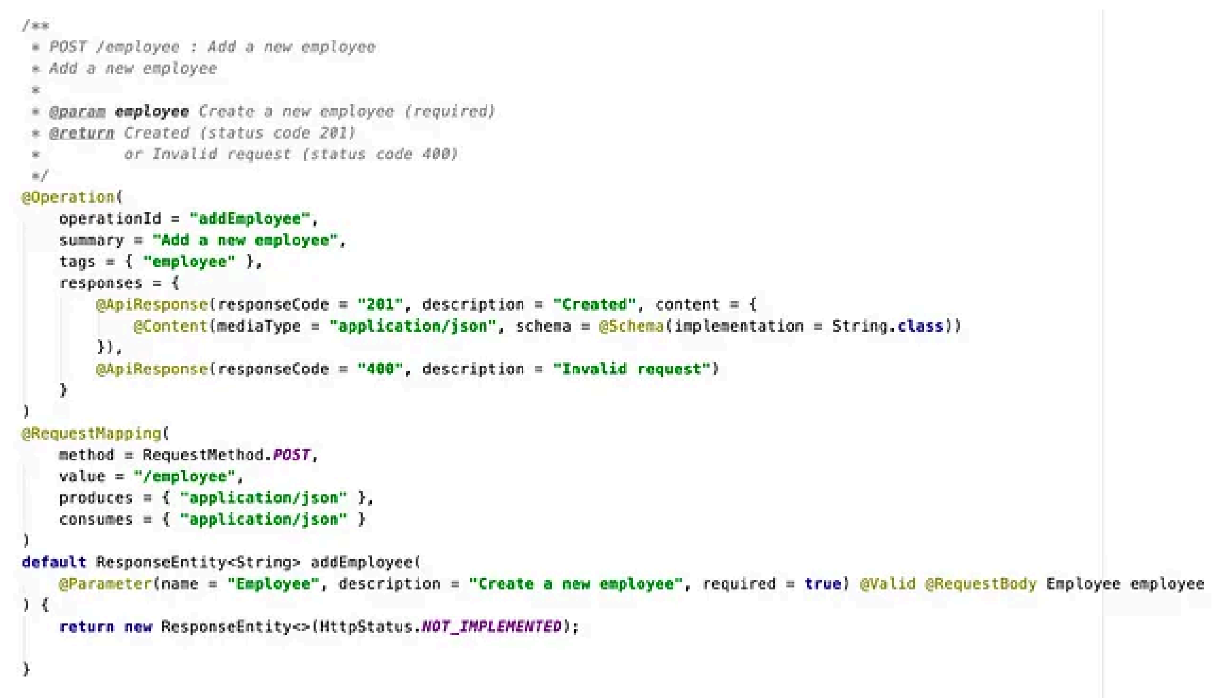
```xml
    <dependencies>
        <dependency>
            <groupId>io.swagger.core.v3</groupId>
            <artifactId>swagger-annotations</artifactId>
            <version>2.2.2</version>
        </dependency>
        <dependency>
            <groupId>org.openapitools</groupId>
            <artifactId>jackson-databind-nullable</artifactId>
            <version>0.2.2</version>
        </dependency>
        <dependency>
            <groupId>javax.validation</groupId>
            <artifactId>validation-api</artifactId>
            <version>2.0.1.Final</version>
        </dependency>
    </dependencies>
```

pom.xml hosted with ❤ by GitHub view raw

Then execute the below maven goal.

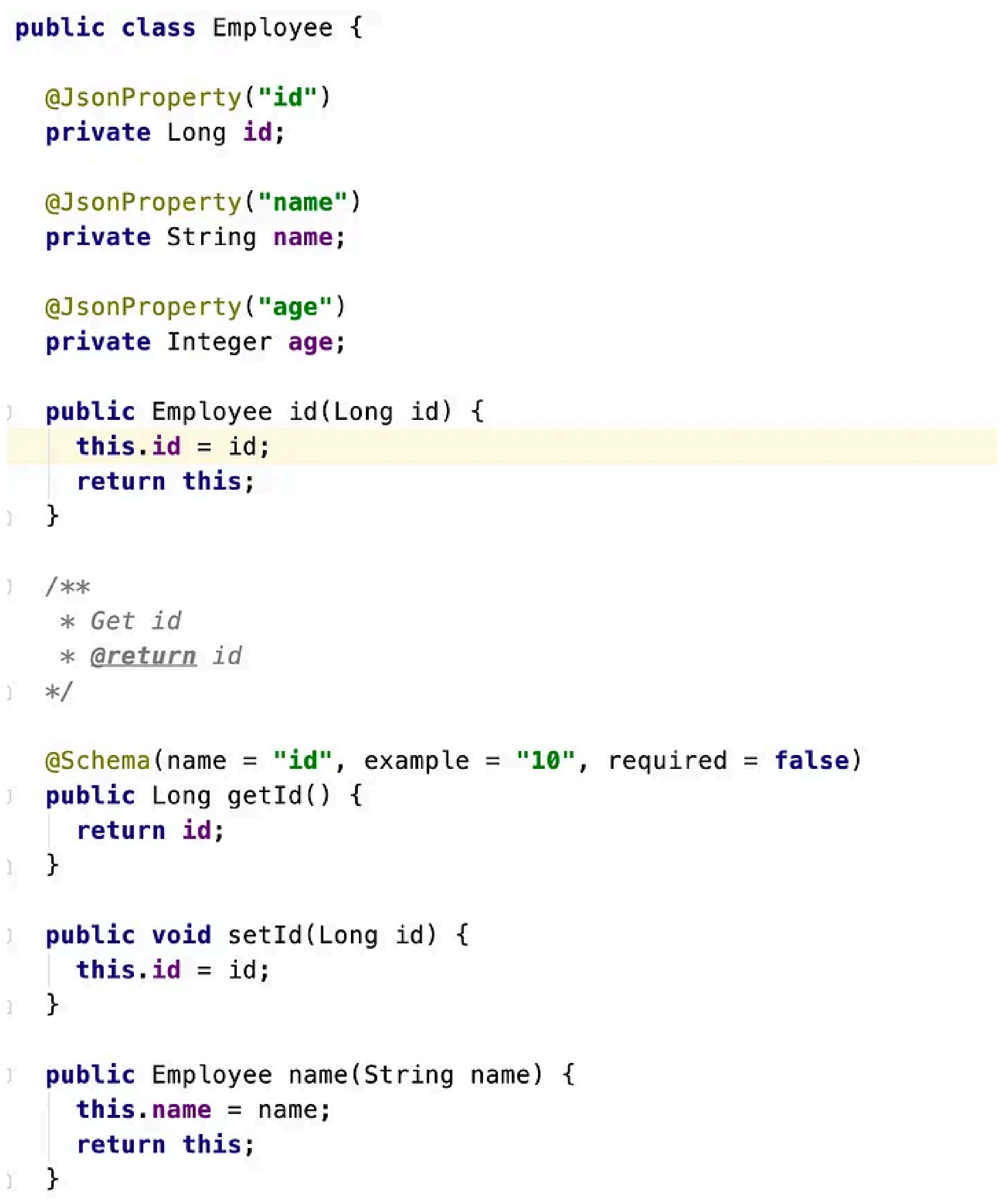
```
mvn clean compile
```

After running maven goal successfully, we can see the generated API and models in the `/target/generated-sources/openapi` directory.



```java
/**
 * POST /employee : Add a new employee
 * Add a new employee
 *
 * @param employee Create a new employee (required)
 * @return Created (status code 201)
 *         or Invalid request (status code 400)
 */
@Operation(
    operationId = "addEmployee",
    summary = "Add a new employee",
    tags = { "employee" },
    responses = {
        @ApiResponse(responseCode = "201", description = "Created", content = {
            @Content(mediaType = "application/json", schema = @Schema(implementation = String.class))
        }),
        @ApiResponse(responseCode = "400", description = "Invalid request")
    }
)
@RequestMapping(
    method = RequestMethod.POST,
    value = "/employee",
    produces = { "application/json" },
    consumes = { "application/json" }
)
default ResponseEntity<String> addEmployee(
    @Parameter(name = "Employee", description = "Create a new employee", required = true) @Valid @RequestBody Employee employee
) {
    return new ResponseEntity<>(HttpStatus.NOT_IMPLEMENTED);

}
```

Generated AddEmployee API



```java
public class Employee {

  @JsonProperty("id")
  private Long id;

  @JsonProperty("name")
  private String name;

  @JsonProperty("age")
  private Integer age;

  public Employee id(Long id) {
    this.id = id;
    return this;
  }

  /**
   * Get id
   * @return id
  */

  @Schema(name = "id", example = "10", required = false)
  public Long getId() {
    return id;
  }

  public void setId(Long id) {
    this.id = id;
  }

  public Employee name(String name) {
    this.name = name;
    return this;
  }
```

Generated Employee.java model class

As we can see, the generated code contains all the request, response, and error codes. We just need to implement the methods generated from the API

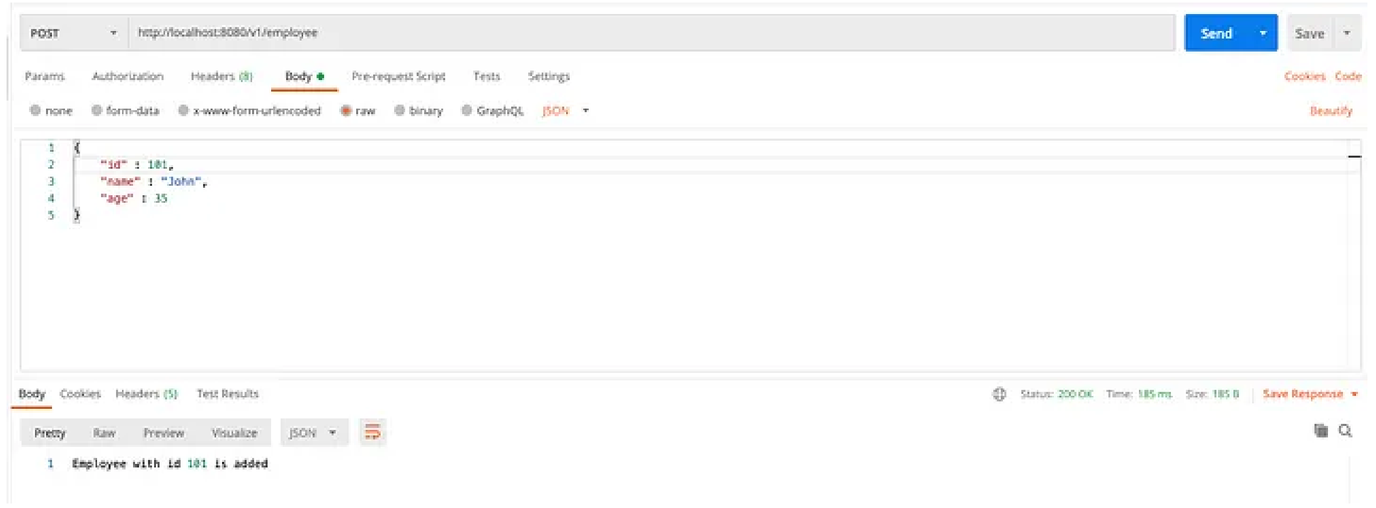
. . .

In this step let's create the EmployeeController.java and implement the EmployeeApi Interface.

```java
@RestController
public class EmployeeController implements EmployeeApi {

    @Autowired
    private EmployeeService employeeService;

    @Override
    public ResponseEntity<String> addEmployee(@Valid Employee employee) {
        employeeService.add(employee);
        return ResponseEntity.ok("Employee with id "+employee.getId()+" is added");
    }

    @Override
    public ResponseEntity<List<Employee>> getEmployees() {
        List<Employee> employeeList = employeeService.getEmployees();
        return ResponseEntity.ok(employeeList);
    }
}
```

EmployeeController.java hosted with ❤ by GitHub view raw

Now run the project and verify whether the APIs are working as expected or not.

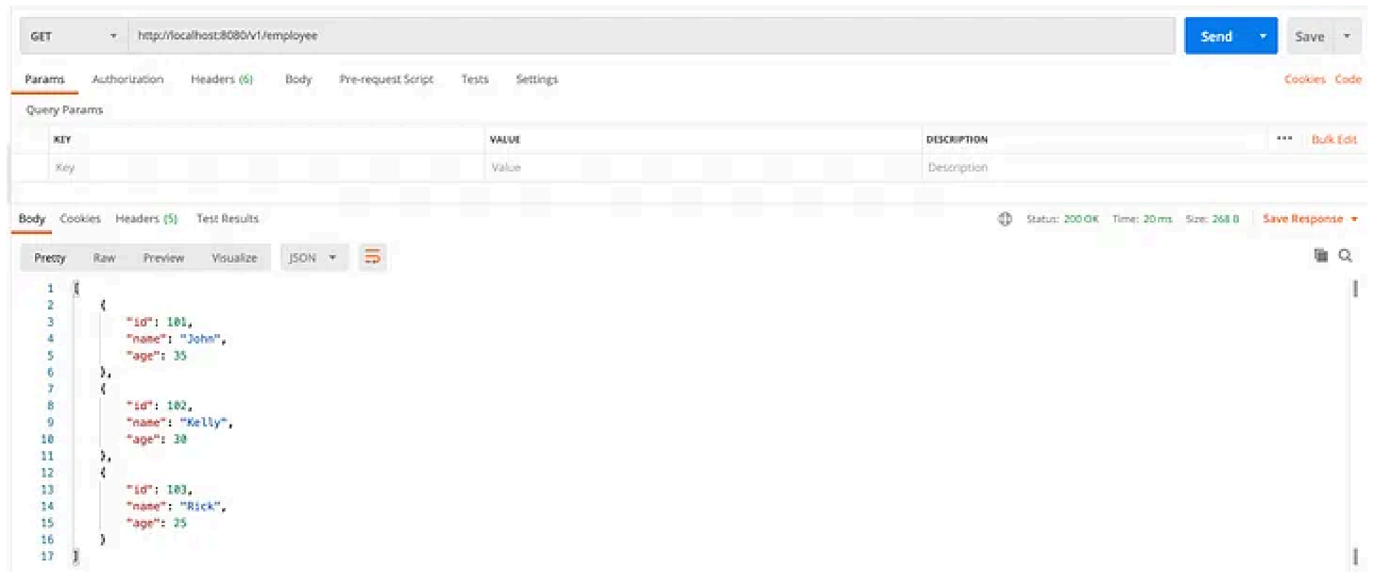

POST http://localhost:8080/v1/employee

```json
{
    "id" : 101,
    "name" : "John",
    "age" : 35
}
```

Status: 200 OK Time: 185 ms Size: 185 B

```
Employee with id 101 is added
```

POST API call to add an employee


GET http://localhost:8080/v1/employee

Status: 200 OK Time: 20 ms Size: 268 B

```json
[
    {
        "id": 101,
        "name": "John",
        "age": 35
    },
    {
        "id": 102,
        "name": "Kelly",
        "age": 30
    },
    {
        "id": 103,
        "name": "Rick",
        "age": 25
    }
]
```

GET API call to fetch all the employees

From the above API calls we verified that both the APIs are working properly.

. . .

Conclusion

In this article we learned how to generate spring boot REST API specs using OpenAPI generator. we can also generate client stubs for various programming languages using the same plugin.

If you liked the article, please offer me a Clap 👏 and share it with your friends and colleagues. If you have any feedback feel free to drop a mail to me. You can also connect with me through linkedIn and twitter.

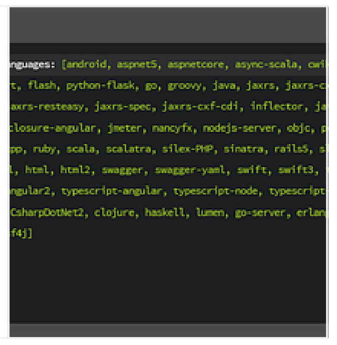
. . .

References

API Documentation & Design Tools for Teams | Swagger

Simplify API development for users, teams, and enterprises with our open source and professional toolset. Find out how...

swagger.io

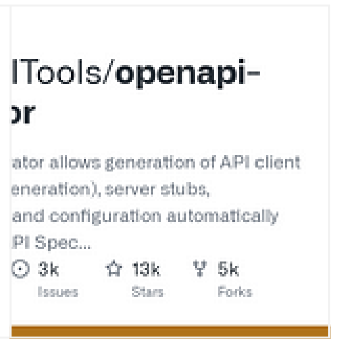


```
nguages: [android, aspnet5, aspnetcore, async-scala, cwi
t, flash, python-flask, go, groovy, java, jaxrs, jaxrs-c
axrs-resteasy, jaxrs-spec, jaxrs-cxf-cdi, inflector, ja
closure-angular, jmeter, nancyfx, nodejs-server, objc, p
pp, ruby, scala, scalatra, silex-PHP, sinatra, rails5, s
l, html, html2, swagger, swagger-yaml, swift, swift3,
ngular2, typescript-angular, typescript-node, typescript
CsharpDotNet2, clojure, haskell, lumen, go-server, erlan
f4j]
```

GitHub - OpenAPITools/openapi-generator: OpenAPI Generator allows generation of API client...

OpenAPI Generator allows generation of API client libraries (SDK generation), server stubs, documentation and...

github.com



ITools/**openapi-**
or

ator allows generation of API client
eneration), server stubs,
and configuration automatically
PI Spec...

3k Issues 13k Stars 5k Forks

Spring Boot Openapi Specification Java Swagger

Written by Tapan Adhikari

Follow

15 Followers

Software Engineer

More from Tapan Adhikari

Tapan Adhikari

Seamless Software Development using Github Actions and Continues Integration

In today's fast-paced software development landscape, it is essential to deliver high-quality code quickly and at the same...

May 18, 2023 5 1

See all from Tapan Adhikari

Recommended from Medium

Tete Kim

Apply Swagger Codegen in Java 21, Spring boot 3.2.x, Gradle 8.5

Using swagger Codegen in gradle project. Java21, spring boot 3.2.x

Mar 30 2

Nine Pages Of My Life

Swagger OpenAPI Tutorial

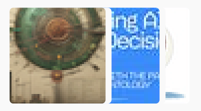
The design and documentation platform for teams and individuals working with the...

Jun 25 9

Lists

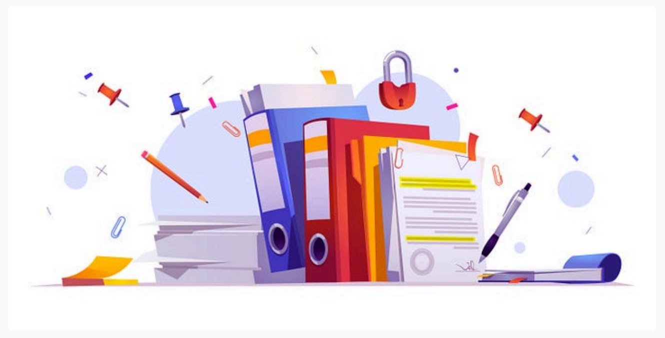
General Coding Knowledge

20 stories · 1559 saves



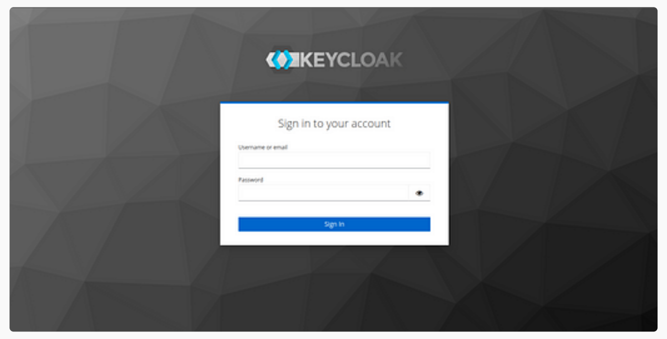
data science and AI

40 stories · 237 saves



Nithidol Vacharotayan

Spring boot 3 Swagger for API documentation



Lejdi Prifti

Secure your application with Spring Security and Keycloak

Spring Boot 3 application using Springdoc, which is a popular choice. The developer ca...

Jul 19 1

Nowadays, writing secure apps is becoming essential, and security is a major componen...

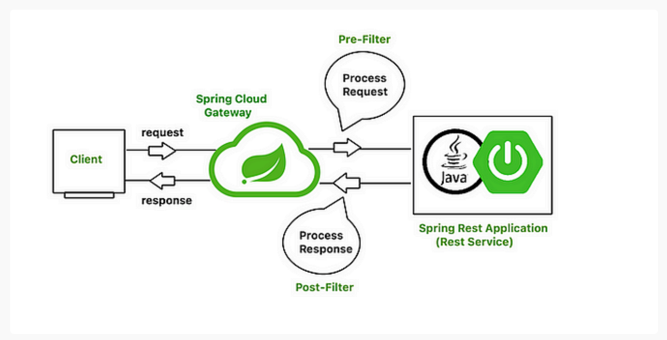
Apr 4 6

Akshay Aryan in Stackademic

Navigating CORS in Spring Boot: A Comprehensive Guide with Sprin...

In the ever-evolving landscape of web development, mastering the intricacies of...

Mar 19 12



Gaurav Raisinghani

Implementing Spring Cloud Gateway: A Comprehensive Guide

Spring Cloud Gateway is a powerful tool designed for managing API traffic efficiently....

Jul 26 12

See more recommendations