EXERCISES — Endpoints

version #1.0

YET ANOTHER KIND OF ASSISTANTS 2024 <assistants@tickets.assistants.epita.fr>

Copyright

Contents

1 Objectives **3**

2 How to start **4**

2.1 Given Files 4

*https://intra.forge.epita.fr

File Tree

```
endpoints/
├─ pom.xml
└─ src/
   └─ main/
      ├─ java/
      │  └─ fr/
      │     └─ epita/
      │        └─ assistants/
      │           └─ presentation/
      │              └─ rest/
      │                 ├─ Endpoints.java   (to submit)
      │                 ├─ request/
      │                 │  └─ ReverseRequest.java   (to submit)
      │                 └─ response/
      │                    ├─ HelloResponse.java   (to submit)
      │                    └─ ReverseResponse.java   (to submit)
      └─ resources/
         ├─ application.properties
         └─ openapi.yaml
```

1 Objectives

The goal of this exercise is to create two simple endpoints: hello and reverse.

hello
Should be a GET endpoint with a path parameter

reverse
Should be a POST endpoint with no path parameter

The specifications of the endpoints can be found in the swagger file *src/main/resources/openapi.yaml*.

For this exercise, you won't need to implement any service and therefore will only create a `presentation` layer. You will need to follow the package architecture shown above.

To complete this exercise, the following annotations will be useful:

- `@GET`: Declares a function as a GET endpoint.

- `@POST`: Declares a function as a POST endpoint.
- `@Path`: Lets you specify the path of the endpoint.
- `@Consumes`: Lets you specify what type of data the endpoint accepts.
- `@Produces`: Lets you specify what type of data the endpoint sends back.

`@Consumes` and `@Produces` can also be put on a class, in which case they apply to all endpoints declared in the class.

Finally, you should take a look at the Response class from the javax.ws.rs.core package for the responses sent by your endpoints.

Tips

- This subject is voluntarily short. You are encouraged to explore the Quarkus documentation and thoroughly read the provided swagger file to guide you on how your endpoints should behave. You may find the guide "Simplified Hibernate ORM with Panache" to be particulary useful.
- This exercise is fundamental to understanding how to start the JWS project. After completing it, make sure you understand everything you have done before moving on.

2 How to start

2.1 Given Files

In order to start the project in good conditions, a tarball is available on the intranet. It contains all the files needed for the proper functioning of the project.

You **should** decompress this tarball and open the directory with IntelliJ before reading the rest of the subject.

When the project is open in your IDE, you can type the following command to start the server.

```
42sh$ mvn quarkus:dev
```

It will launch your API on the port 8082. In order to make a request on your endpoints you should use the following commands:

```
42sh$ curl http://localhost:8082/hello/yaka # replace {name} with an actual value
42sh$ curl http://localhost:8082/reverse -X POST -H "Content-Type: application/json" \
    -d '{"content": "hello world"}'
```

This should produce:

```
{"content":"hello yaka"}
{"original":"hello world","reversed":"dlrow olleh"}
```

You mean it's working? For real this time?