{project-name} Documentation

Systemkern

Version , 2018-06-23 21:49:18 UTC

Table of Contents

Application Start

To start {project-name} open your Unix shell or Windows command prompt, navigate to the {project-name} application directory and type or copy the following command:

```
java -jar {final-name}.jar
```

Chapter 1. Configuration and Startup

The application is configured via two different configuration files. The Technical `application.properties` and the business oriented `application.yml`. The configuration for the so called default profile is called "application.yml". This is most likely the only configuration file that you will need to modify.

Chapter 2. Multiple execution profiles

This application supports multiple profiles which can be activated during startup. They are configured in separate configuration files which are named `application-PROFILENAME.yml`. If you want to use a profile called "myprofile2" you make a copy of application.yml named `application-myprofile2.yml` and start the application as follows:

```
java -jar {final-name}.jar --spring.profiles.active=myprofile2
```

For more information please refer to the Spring Boot Documentation.

API Requests

Chapter 3. Additional API Requests

3.1. Echo

Echo is a sample post request which returns exactly the same object as the one provided. It can be used to test Json Object mapper implementations

operation::echo[snippets='curl-request,http-request,http-response,response-fields']

3.2. Ping

As simple ping signal which returns the current system timestamp

operation::ping[snippets='curl-request,http-request,http-response,response-fields']

3.3. User

User is a module created to answer CRUD(Create,read,update and delete) requests, it does not always return a json, but a Http code that show if was successful. The base url is port/users. Next examples of http requests and responses are exposed.

3.3.1. Create

To create an user is required to send a POST request to the base url. The body of request should contain the following data: "name" and "password". If the request is right then status code of response should be 201 created, the body of response contains the data provided plus 2 links, which can be used to access the user that have been created.

operation::user_create[snippets='curl-request,http-request,http-response']

3.3.2. Login

To login and access system, username and password must be sent to Url + /login. Some data is returned including an access token, which must be added in the header of future read and update requests.

operation::user_login[snippets='curl-request,http-response']

3.3.3. Read

The "id" of user should be added at the end of URL, the Http verb of request is GET but a body is not necessary. If the URL is correct and there is an user identified with the "id" provided, then web service sends 200 ok status code and the body of response contains the data of user. Remember to add the access token in a header called Authorization,the syntax is for example: "Bearer [token]"

operation::user_read[snippets='curl-request,http-request,http-response,response-fields']

3.3.4. Update

To update an user, the url is equal to the Read case. The difference is in the verb which is PUT http, the body of the request should contain the new values of "name" and "password". If the request is right then the response contains; the data provided in the request and 2 links to access the user that have been updated, the status should be 200 ok. Remember to add the access token in a header called Authorization,the syntax is for example: "Bearer [token]"

operation::user_update[snippets='curl-request,http-request,http-response']

3.3.5. Delete

User delete is forbidden

operation::user_delete[snippets='curl-request,http-request,http-response']