Repath

RESTfull Application with Spring Boot

Petros Melachrinidis
[COMPANY NAME] [Company address]

Document Index

General Purpose of this document

This document includes all the important information on using the “Repath RESTfull application”. The document starts with a small description of the assets of the application for the reader to gain a wider view of the project. Following the demonstration, the guide presents a full installation process and how the reader can use the application.

Description of the application

The main goal of the project is to provide the end user an application for basic create, read, update and delete (CRUD) functionality via REST client end points.

We have created a local database using MySQL and Workbench. After the database creation, we created a Spring Boot project built on Maven built automation tool, using Java 8 and NetBeans as IDE. The application was built using SOLID principals the basic structure of MVC project (controllers, service, dao) and a very careful approach to separation of concerns. Additionanally, Postman REST client was used to test the final end points and the basic functionality. For Spring Boot we used the following dependencies: Spring Boot DevTools, Spring Web, Spring Data JPA, MySQL Driver.

Installation

1. To use the application, you need to download the .zip file from the following GitHub Repository.

   ```
   https://github.com/petrosmel/Repath.git
   ```

2. You will also need to have installed the JDK 1.8. Download and install any version of the JDK 1.8. Install JDK in a folder whose path does not contain spaces. Make sure to also set the environmental variable JAVA_HOME in your system. The variable must point to your JDK folder (absolute path). The path of the "\jre\bin" folder under the JDK folder should also be prepended to the PATH system variable.
3. Download and install any version of Maven 3.x . Install Maven in a folder whose path does not contain spaces. Make sure to also set the environmental variables M2_HOME and M2 in your system. The M2_HOME variable must point to the maven folder (absolute path). The M2 variable must point to the "bin" folder under the maven folder (absolute path). The absolute path of the "bin" folder under the maven folder (M2) should also be prepended to the PATH system variable.
4. Install a REST client. We suggest Postman REST client as it is free to use and very capable.
5. Finally you must create a local database to your system. You can use MySQL Workbench to either import the repath.sql file included to the above repository or to run the MySQL code also given. Be careful to use to the local database the following information. Username root, Password 1234.
6. After downloading and unzipping the file start a Command Prompt application and guide yourself inside the application folder. Then type:

   ```
   mvn spring-boot:run
   ```

The application is now running. You can now try the following end points to check the functionality.

The ends points

Show all users from Database

URL	http://localhost:8081/api/users
METHOD	GET
EXAMPLE	http://localhost:8081/api/users
RETURN RESULT	


```
[
    {
        "id": 1,
        "name": "Manolis",
        "email": "manolis@manolis.gr",
        "company": "Microsoft"
    },
    {
        "id": 2,
        "name": "Maria",
        "email": "maria@maria.gr",
        "company": "Amazon"
    }…
]
```


Show all users per company name

URL	http://localhost:8081/api/users/company/{company}
METHOD	GET
EXAMPLE	http://localhost:8081/api/users/company/Amazon
RETURN RESULT	


```
[
    {
        "id": 1,
        "name": "Manolis",
        "email": "manolis@manolis.gr",
        "company": "Amazon"
    },
    {
        "id": 2,
        "name": "Maria",
        "email": "maria@maria.gr",
        "company": "Amazon"
    }…
]
```

Update existing user

URL	http://localhost:8081/api/users/
METHOD	PUT
EXAMPLE	http://localhost:8081/api/users the body of the request should include the user information updated.

```
[{
        "id": 1,
        "name": "Manolis",
        "email": "manolis@manolis.gr",
        "company": "Amazon"
}]
```

RETURN RESULT

```
[{
        "id": 1,
        "name": "Manos",
        "email": "manos@manos.gr",
        "company": "Amazon"
}]
```

Deleting existing user

URL	http://localhost:8081/api/users/{id}
METHOD	DELETE
EXAMPLE	http://localhost:8081/api/users/1 the ID must be the selected users id
RETURN RESULT	User with ID:1 was just deleted!

Add new user to Database

URL	http://localhost:8081/api/users
METHOD	POST
EXAMPLE	http://localhost:8081/api/users the body of the request should include the new user information without the ID because the database creates it automatically.

```
[{
        "name": "Manolis",
        "email": "manolis@manolis.gr",
        "company": "Amazon"
}]
```

RETURN RESULT

```
[{
        "id": 21,
        "name": " Manolis",
        "email": " manolis@manolis.gr",
        "company": "Amazon"
}]
```

User Interface

Finally, there is provided a user interface that is user friendly and implements all the above procedures. A user can see all the users in the database, can update a user by typing and editing user's information, can delete the selected user and also can choose to see users by company using the filter.