# Ebi Person

Contents

[Ebi Person 1](#_Toc64288010)

[Introduction 2](#_Toc64288011)

[Project infrastructure 2](#_Toc64288012)

[APIs 3](#_Toc64288013)

[Save Person 3](#_Toc64288014)

[Update Person 4](#_Toc64288015)

[Retrieve Person 4](#_Toc64288016)

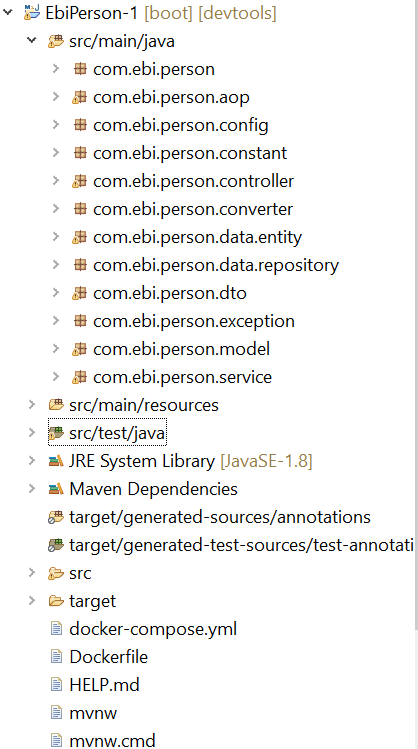
[Delete Person 5](#_Toc64288017)

## Introduction

Ebi person is a simple spring boot application which provide a service for storing ,updating, retrieving and delete person.

## Project infrastructure

Ebi application is using h2 for the database which is very light database and the most important thing that it’s in memory database. In below the project structure and as shown below the project is built by maven building tool and docker



The application has configuration in application.properties which is loaded while application startup

#change the defaulf port to 8000

server.port=8000

#Changing console display log from info to debug

logging.level.org.springframework=debug

#change the userName and password for user authentication

spring.security.user.name=mohsen

spring.security.user.password=P@ssw0rd

#spring.security.user.roles=Create,Delete

#Show sql query that occured in the Database

spring.jpa.show-sql=true

#configuration for h2 database

spring.h2.console.enabled=true

spring.datasource.platform=h2

spring.datasource.url=jdbc:h2:mem:ebi

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

# APIs

As mentioned the Ebi person has provided services for storing, updating, retrieving and deleting person from the database

### Save Person

|  |  |
| --- | --- |
|  |  |
| URL | http://localhost:8000/persons/ |
| Method | POST |
| Username | Mohsen |
| Password | P@ssw0rd |
| Request | {          "firstName": "ahmed",          "lastName": "mohsen",          "age": "28",          "favouriteColor": "red"    } |
| Response | {          "id": 1,          "firstName": "ahmed",          "lastName": "mohsen",          "age": "28",          "favouriteColor": "red"   } |

### Update Person

|  |  |
| --- | --- |
|  |  |
| URL | http://localhost:8000/persons/1 |
| Method | PUT |
| Username | Mohsen |
| Password | P@ssw0rd |
| Request | {          "firstName": "ahmed",          "lastName": "mohsen",          "age": "28",          "favouriteColor": "red"    } |
| Response | {          "id": 1,          "firstName": "ahmed",          "lastName": "mohsen",          "age": "28",          "favouriteColor": "red"   } |

### Retrieve Person

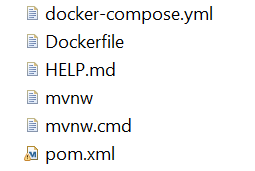
|  |  |
| --- | --- |
|  |  |
| URL | http://localhost:8000/persons/1 |
| Method | GET |
| Username | Mohsen |
| Password | P@ssw0rd |
| Response | {          "id": 1,          "firstName": "ahmed",          "lastName": "mohsen",          "age": "28",          "favouriteColor": "red"   } |

### Delete Person

|  |  |
| --- | --- |
|  |  |
| URL | http://localhost:8000/persons/1 |
| Method | Delete |
| Username | Mohsen |
| Password | P@ssw0rd |
| Response | Response status would be 204 No content as it indicates for successfully deletion |

## Build Project

The application is built by using maven and docker as shown on below.



In above there are couple of files use for building the application and make it work .

Pom.xml is maven configuration file contains all dependencies that we need in the application.

Dockerfile & docker-compse.yaml for running and building the application.

### Commands

There are couple of commands to running the application using docker. First go to the application directory and run the below command.

docker-compose u -d

To stop the application. Please use the below command

docker-compose stop

## Enhancements

Because the application is so simple and provide simple services for CRUD operation may be further we can enhance some parts of the application to make it more robust and secure.

For security we can use Bcrybt for hashing the user password instead of SHA 256 because it’s more powerful.

Instead of configuring the username and password in application.properties we can have a table that contains all users with their password and roles, therefore we can load them while startup the application and authorized the API.

For scalability we can add load balancer for availability , performance and maintain.