Technical Assignment Documentation

14 February 2021

Table of Contents

* Introduction
* Dependencies
* Project Structure
* Build & Running the project
* Introduction

Restful API includes CRUD operations for person with Json format as below

{

    "person":

[{

    "first\_name":"john",

    "last\_name":"Keynes",

    "age":"29",

    "favorite\_coulour":"red"

},

{

    "first\_name":"mohamed",

    "last\_name":"ramadan",

    "age":"30",

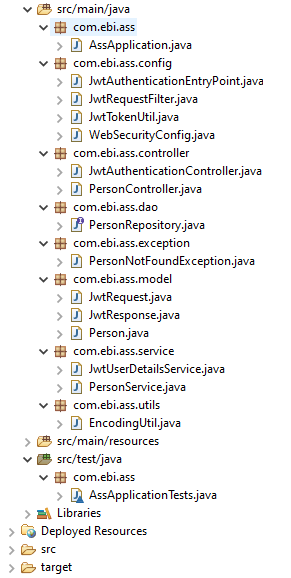
    "favorite\_coulour":"white"

}

]

}

* Dependencies
* JDK 1.8
* Spring boot 2.4.2
  + Spring boot data JPA
  + Spring boot Security – JWT
  + Spring boot test – Junit5
  + Spring boot dev tools
* Maven 3.6.3
* Embeded Tomcat Web server
* In memory database H2
* Lombok
* Postman as A client tool
* Project Structure



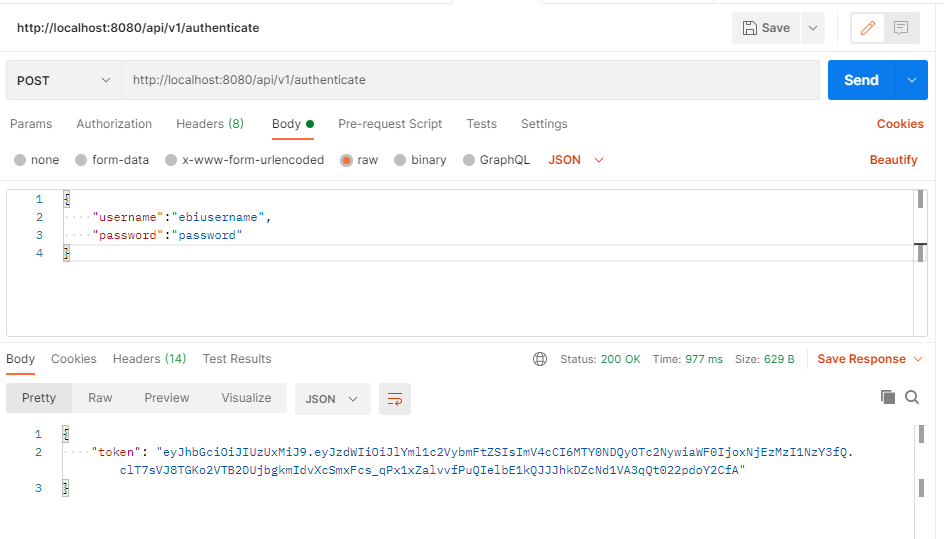
* Model Layer:
  + **Person**: represents person json.
  + **JwtRequest**: represent authentication request
  + **JwtResponse**: represents authentication response
* Controller Layer that represents the Rest API:
  + **PersonController**: represents Rest API with all crud operations.
  + **JwtAuthenticationController**: represents Rest API responsible for creating user authentication token (JWT)
* Service Layer Interact with DAO layers to executes crud operations :
  + **PersonService**: include crud operations logic
* DAO Layer Interact with H2 In memory DB :
  + **PersonRepository**: exchange data with H2 in memory DB
* Config Layer:
  + **JwtAuthenticationEntryPoint**: prevent any unauthorized request calls.
  + **JwtRequestFilter**: Any incoming request pass first through this class to check if the token exists or not and to check it`s validity.
  + **JwtTokenUtil**: responsible for all jwt operations, like creation, generation, payload extraction, validation.
  + **WebSecurityConfig**: responsible for any customization in project security, like permitting specific urls,…
* Utils Layer:
  + EncodingUtil: Encode secret key used as a digital signature in JWT generation.
* Build & Running the project
  + Execute below steps to build and run the project:
    1. mvn clean install -DskipTests
    2. mvn spring-boot:run
  + Execute below steps to run the project test cases:
    1. mvn test
* After Application becomes up and running we should follow below steps to be able to call the REST API:
* Generate Token:
  1. URL: <http://localhost:8080/api/v1/authenticate>
  2. username: ebiusername & password: password
  3. json:

{

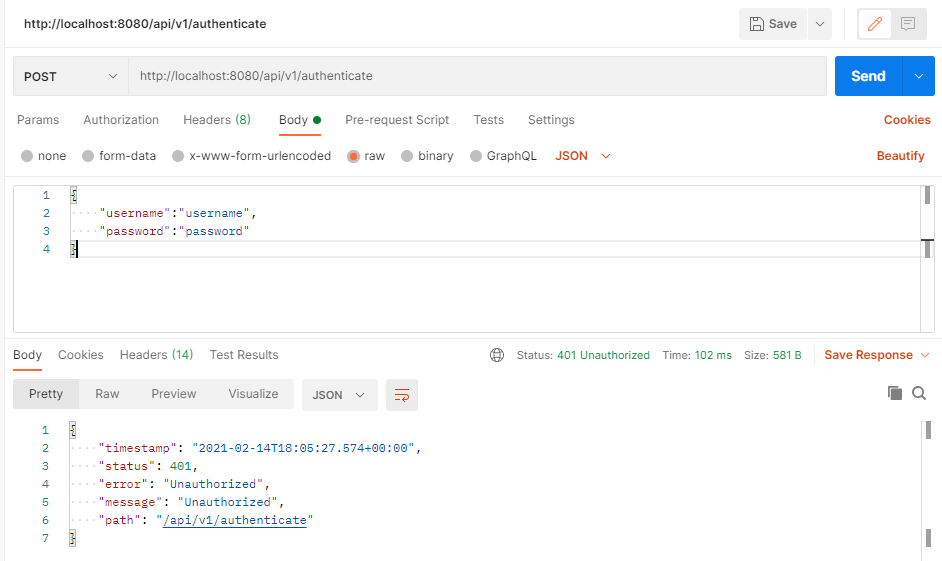
"username":"ebiusername",

"password":"password"

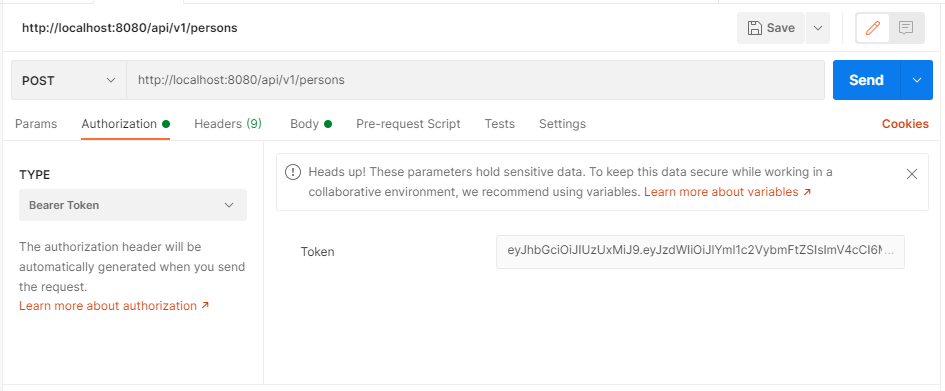
}



We should use the generated token for any call request to the rest API, otherwise *Unauthorized error* will be generated while calling the rest API without the token or using invalid token or invalid credentials while generating the token.



* Execute Create Person:
  1. URL: <http://localhost:8080/api/v1/persons>, Post Mapping
  2. Add Authentication token generated above in postman Authorization tab, and choose the type as Bearer token:



Send request like below:

*Request Json:*

{

    "person":

[{

    "first\_name":"john",

    "last\_name":"Keynes",

    "age":"29",

    "favorite\_coulour":"red"

},

{

    "first\_name":"mohamed",

    "last\_name":"ramadan",

    "age":"30",

    "favorite\_coulour":"white"

}

]

}

*Response Json:*

{

    "Status": "Persons successfully created with uris below...",

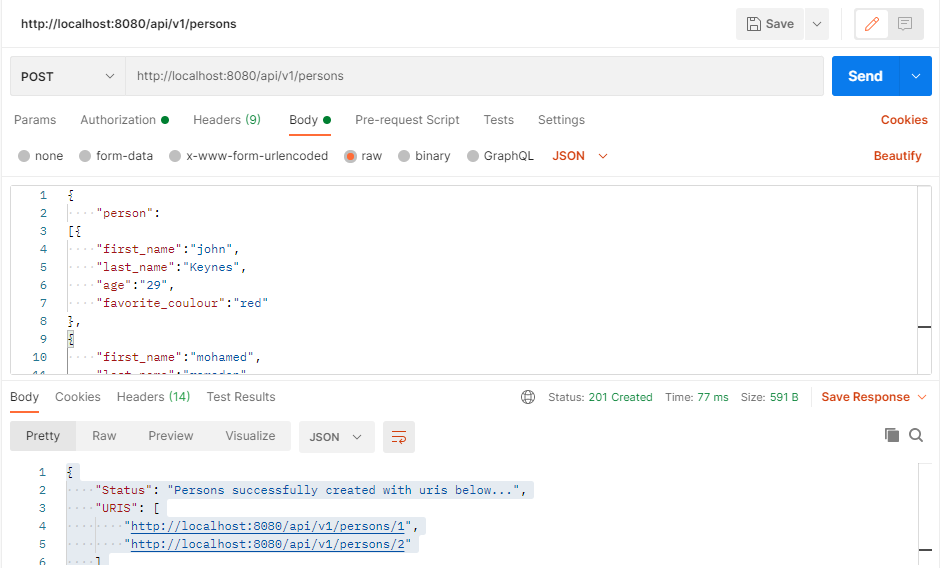
    "URIS": [

        "http://localhost:8080/api/v1/persons/1",

        "http://localhost:8080/api/v1/persons/2"

    ]

}



* Execute Get All Persons:
  1. URL: <http://localhost:8080/api/v1/persons>, Get Mapping
  2. Add Authentication token generated like previous call.

Response json:

{

    "person": [

        {

            "id": 1,

            "first\_name": "john",

            "last\_name": "Keynes",

            "age": 29,

            "favorite\_coulour": "red"

        },

        {

            "id": 2,

            "first\_name": "mohamed",

            "last\_name": "ramadan",

            "age": 30,

            "favorite\_coulour": "white"

        }

    ]

}

* Execute Get Specific Person:
  1. URL: <http://localhost:8080/api/v1/persons/1>, Get Mapping,

As example I used Id: 1

* 1. Add Authentication token generated like previous call.

Response Token:

{

    "person": {

        "id": 1,

        "first\_name": "john",

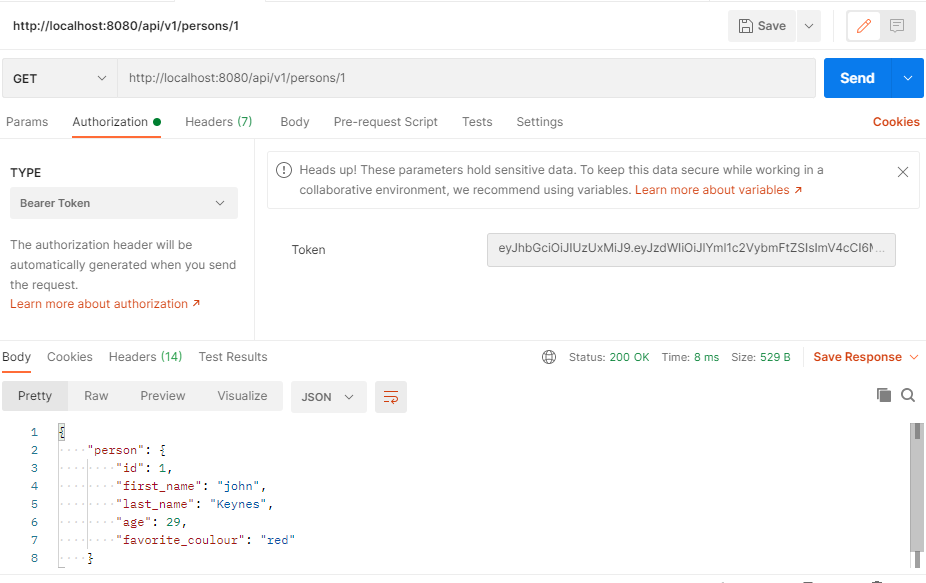
        "last\_name": "Keynes",

        "age": 29,

        "favorite\_coulour": "red"

    }

}



* Execute Update Specific Person:
  1. URL: <http://localhost:8080/api/v1/persons/2>, Put Mapping,

As example I used Id: 2

* 1. Add Authentication token generated like previous call.

Request Json:

{

    "first\_name":"mohamed",

    "last\_name":"ramadan",

    "age":"30",

    "favorite\_coulour":"black"

}

Response Json:

{

    "id": 2,

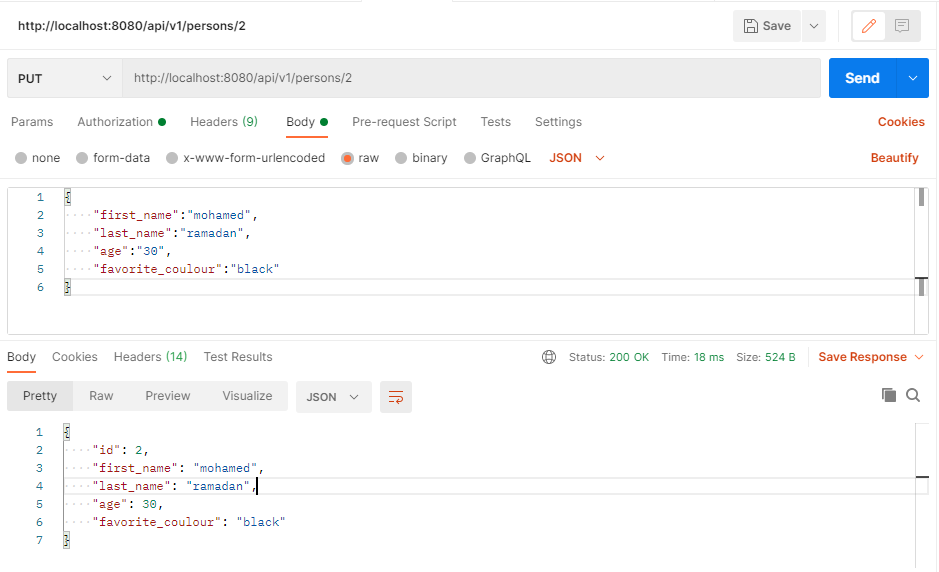
    "first\_name": "mohamed",

    "last\_name": "ramadan",

    "age": 30,

    "favorite\_coulour": "black"

}



* Execute Delete Specific Person:
  1. URL: <http://localhost:8080/api/v1/persons/1>, Delete Mapping,

As example I used Id: 1

* 1. Add Authentication token generated like previous call.

Response Json:

{

    "deleted": **true**

}

