

A Spring Boot Application for Securing a REST API with JSON Web Token (JWT)

To run the application

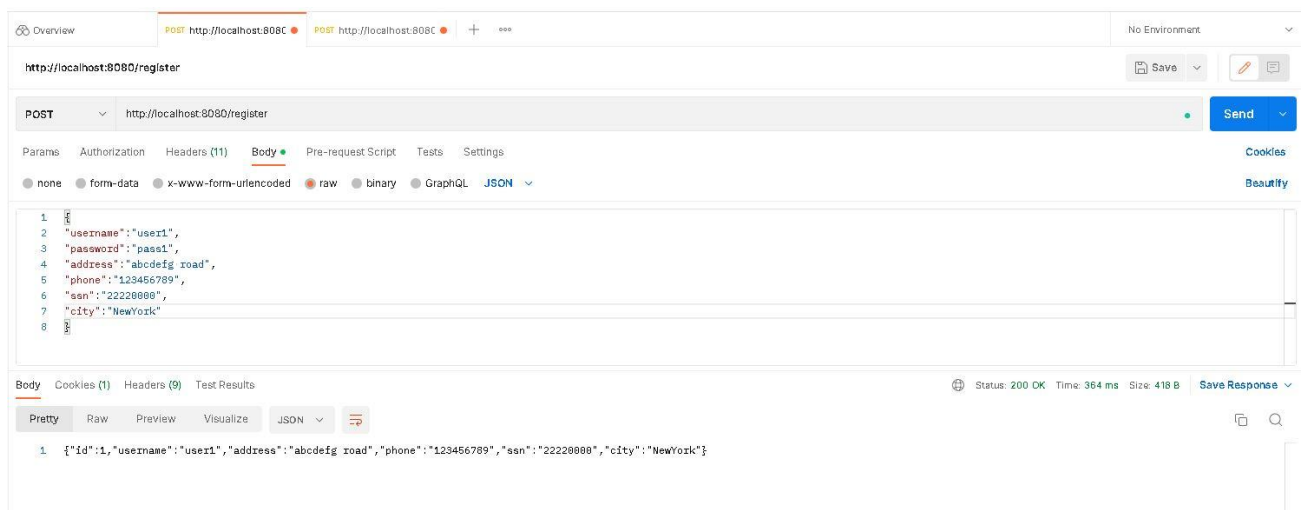
- Make sure you have Maven and Java 1.8 or greater
- Make sure you have created a database in Mysql. You can modify the properties of database in application.properties file.

Postman Demo: (POST, GET, PUT/UPDATE, DELETE)

Register two users with information.

POST: <http://localhost:8080/register>

```
{  
  "username": "user1",  
  "password": "pass1",  
  "address": "abcdefg road",  
  "phone": "123456789",  
  "ssn": "22220000",  
  "city": "NewYork"  
}
```



```
{  
  "username": "user2",  
  "password": "pass2",  
  "address": "ghjkl road",  
  "phone": "100000000",  
  "ssn": "00008888",  
  "city": "San Francisco"  
}
```

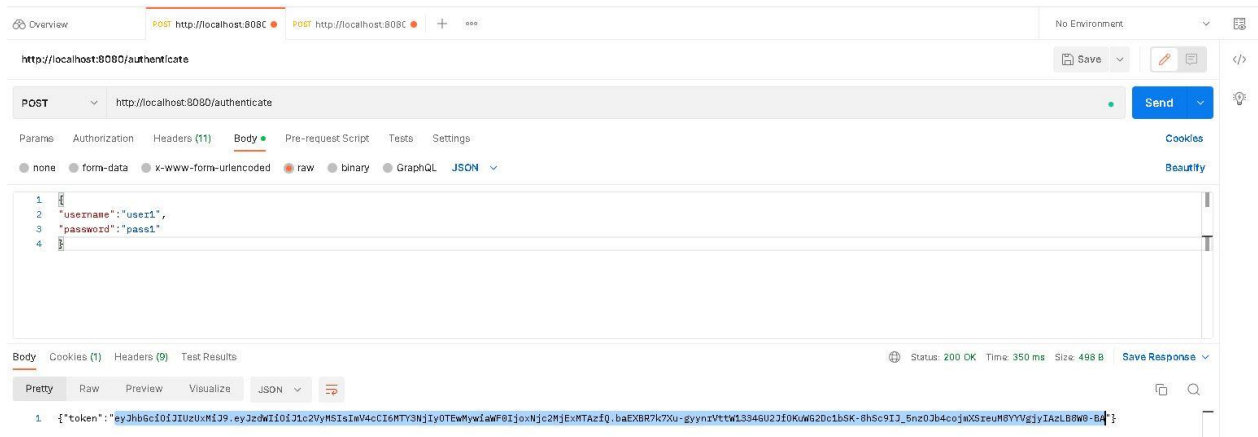
The screenshot displays a REST client interface with the following details:

- Overview Tab:** Shows two recent POST requests to `http://localhost:8080/`.
- Request Configuration:**
 - Method:** POST
 - URL:** `http://localhost:8080/register`
 - Body Type:** raw (selected)
 - Body Content:**

```
1. {  
2.   "username": "user2",  
3.   "password": "pass2",  
4.   "address": "ghjkl road",  
5.   "phone": "100000000",  
6.   "ssn": "00008888",  
7.   "city": "San Francisco"  
8. }
```
- Response Section:**
 - Status:** 200 OK
 - Time:** 91 ms
 - Size:** 422 B
 - Save Response:** (button)
 - Body Tab:** Pretty (selected)
 - Response Content:**

```
1. { "id": 2, "username": "user2", "address": "ghjkl road", "phone": "100000000", "ssn": "00008888", "city": "San Francisco" }
```

}



Copy and enter the token to the header, and get all of user's information.

GET: <http://localhost:8080/get/users>

The screenshot shows the Postman interface for a POST request to `http://localhost:8080/authenticate`. The 'Headers' tab is selected, showing a single header: 'Authorization' with a Bearer token. The token is a long alphanumeric string. The 'Send' button is visible in the top right corner.

KEY	VALUE	DESCRIPTION
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiI1c2VyMSIsImlmIjoiY2VzMTY3NiJyOTUwMywiaWF0IjoxNjc2MjExMTAzfQ.LaEXBR7k7Xu-qvvnVttW1334GU2Jf0KuWG2Dc1bSK-	Description

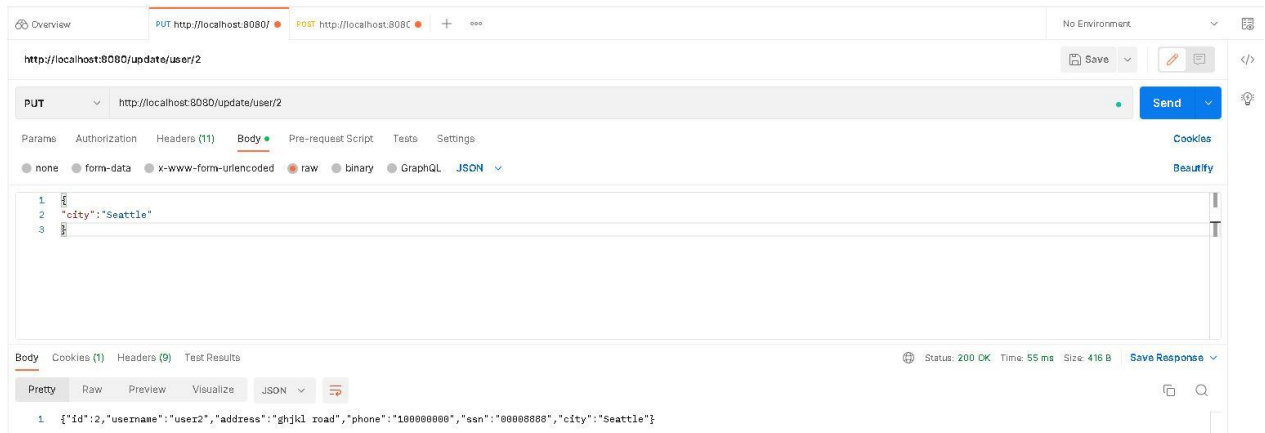
The screenshot shows the Postman interface for a GET request to `http://localhost:8080/get/users`. The 'Body' tab is selected, showing the response body in JSON format. The status is 200 OK, and the response is saved. The response body contains an array of two user objects.

```
1. [{"id":1,"username":"user1","address":"abcdefg road","phone":"123456789","ssn":"22228888","city":"NewYork"}, {"id":2,"username":"user2","address":"ghjkl road","phone":"188888888","ssn":"88888888","city":"San Francisco"}]
```

Enter the token in header, and update the city of user2 to "Seattle".

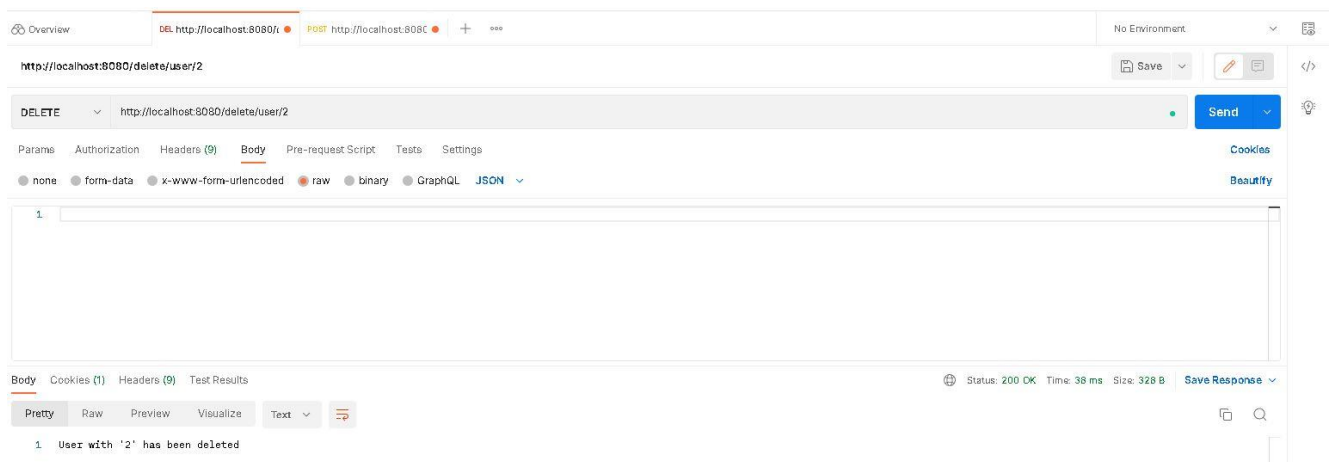
PUT/UPDATE: <http://localhost:8080/update/user/2>

```
{  
  "city": "Seattle"  
}
```



Enter the token in header, and delete user2.

DELETE: <http://localhost:8080/delete/user/2>



Properties of Java class

- JwtTokenUtil (config, class) - JWT operations such as creation and validation are handled by the JwtTokenUtil.
-
- JwtUserDetailsService (service, class) - obtain the user details from the database during the authentication.
-
- UserService (service, class) - contains the methods of save User, get Users, delete users and so on.
-
- HelloWorldController(controller, class) - a Controller class for exposing a GET, POST, UPDATE and DELETE REST API.
-
- JwtAuthenticationController (controller, class) - In the POST API, the client sends the username and password as part of the request body. The Spring Authentication Manager verifies these credentials, and if they are correct, a JWT token is produced by the JWTTokenUtil and returned to the client.
-
- JwtRequest(model, class) - It stores the username and password that recieved from the client.
-
- JwtResponse(model, class) -The JWT is placed into a response, which is then sent back to the user.
-
- JwtRequestFilter(config, class) - The JwtRequestFilter inherits from the Spring Web Filter OncePerRequestFilter class and is executed for every incoming request. It verifies if the request contains a valid JWT token and, if so, sets the authentication context to indicate that the current user has been authenticated.
-
- JwtAuthenticationEntryPoint(config, class) - By extending the Spring AuthenticationEntryPoint class and overriding its commence method, this class denies all unauthenticated requests and sends an error code 401.
-
- WebSecurityConfig(config, class) - This class extends the WebSecurityConfigurerAdapter, which provides a convenient method for customizing both WebSecurity and HttpSecurity.