

Assignment 1 - REST API

Codes:

Create JPA Entity - User.java:

```
src > main > java > net > javaguides > springboot > entity > User.java > Java > User.java
1 package net.javaguides.springboot.entity;
2
3 import jakarta.persistence.*;
4 import lombok.AllArgsConstructorConstructor;
5 import lombok.Getter;
6 import lombok.NoArgsConstructorConstructor;
7 import lombok.Setter;
8
9 @Getter
10 @Setter
11 @NoArgsConstructorConstructor
12 @AllArgsConstructorConstructor
13 @Entity
14 @Table(name = "users")
15 public class User {
16
17     @Id
18     @GeneratedValue(strategy = GenerationType.IDENTITY)
19     private Long id;
20     @Column(nullable = false)
21     private String firstName;
22     @Column(nullable = false)
23     private String lastName;
24     @Column(nullable = false, unique = true)
25     private String email;
26 }
```

Create Spring Data JPA Repository for User JPA Entity

```
1 package net.javaguides.springboot.repository;
2
3 import net.javaguides.springboot.entity.User;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface UserRepository extends JpaRepository<User, Long> {
7 }
```

UserService Interface:

```
package net.javaguides.springboot.service;

import net.javaguides.springboot.entity.User;

import java.util.List;

public interface UserService {
    User createUser(User user);

    User getUserById(Long userId);

    List<User> getAllUsers();

    User updateUser(User user);

    void deleteUser(Long userId);
}
```

UserServiceImpl Class:

```
package net.javaguides.springboot.service.impl;

import lombok.AllArgsConstructor;
import net.javaguides.springboot.entity.User;
import net.javaguides.springboot.repository.UserRepository;
import net.javaguides.springboot.service.UserService;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
@AllArgsConstructor
public class UserServiceImpl implements UserService {

    private UserRepository userRepository;

    @Override
    public User createUser(User user) {
        return userRepository.save(user);
    }

    @Override
    public User getUserById(Long userId) {
        Optional<User> optionalUser = userRepository.findById(userId);
        return optionalUser.get();
    }

    @Override
    public List<User> getAllUsers() {
        return userRepository.findAll();
    }

    @Override
    public User updateUser(User user) {
        User existingUser = userRepository.findById(user.getId()).get();
        existingUser.setFirstName(user.getFirstName());
        existingUser.setLastName(user.getLastName());
        existingUser.setEmail(user.getEmail());
        User updatedUser = userRepository.save(existingUser);
        return updatedUser;
    }
}
```

```
    @Override
    public void deleteUser(Long userId) {
        userRepository.deleteById(userId);
    }
}
```

Creating UserController - Building CRUD Rest APIs:

```
package net.javaguides.springboot.controller;

import lombok.AllArgsConstructor;
import net.javaguides.springboot.entity.User;
import net.javaguides.springboot.service.UserService;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@AllArgsConstructor
@RequestMapping("api/users")
public class UserController {

    private UserService userService;

    // build create User REST API
    @PostMapping
    public ResponseEntity<User> createUser(@RequestBody User user){
        User savedUser = userService.createUser(user);
        return new ResponseEntity<>(savedUser, HttpStatus.CREATED);
    }

    // build get user by id REST API
    // http://localhost:8080/api/users/1
    @GetMapping("{id}")
    public ResponseEntity<User> getUserById(@PathVariable("id") Long userId){
        User user = userService.getUserById(userId);
        return new ResponseEntity<>(user, HttpStatus.OK);
    }
}
```

```
// Build Get All Users REST API
// http://localhost:8080/api/users
@GetMapping
public ResponseEntity<List<User>> getAllUsers(){
    List<User> users = userService.getAllUsers();
    return new ResponseEntity<>(users, HttpStatus.OK);
}

// Build Update User REST API
@PutMapping("{id}")
// http://localhost:8080/api/users/1
public ResponseEntity<User> updateUser(@PathVariable("id") Long userId,
                                         @RequestBody User user){
    user.setId(userId);
    User updatedUser = userService.updateUser(user);
    return new ResponseEntity<>(updatedUser, HttpStatus.OK);
}

// Build Delete User REST API
@DeleteMapping("{id}")
public ResponseEntity<String> deleteUser(@PathVariable("id") Long userId){
    userService.deleteUser(userId);
    return new ResponseEntity<>("User successfully deleted!", HttpStatus.OK);
}
```

Testing Screen Shots:

Create User REST API:

The screenshot shows the Postman application interface. On the left, the sidebar displays 'FREDLUIS OBIDOS's Workspace' with sections for Collections, Environments, Flows, and History. The main area shows a 'New Request' dialog for a 'Spring Boot / New Request' collection. The request method is set to 'POST' and the URL is 'http://localhost:8080/api/users'. The 'Body' tab is selected, showing raw JSON input:

```
1 {
2   "firstName": "Fredluis",
3   "lastName": "Obidos",
4   "email": "fobidos.a12241571@umak.edu.ph"
5 }
```

Below the request, the response is displayed under the 'Body' tab, showing a '201 Created' status with a response time of 35 ms and a response size of 260 B. The response body is identical to the input JSON:

```
1 {
2   "id": 1,
3   "firstName": "Fredluis",
4   "lastName": "Obidos",
5   "email": "fobidos.a12241571@umak.edu.ph"
6 }
```

At the bottom, there are various navigation and utility buttons like 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

Get Single User REST API:

The screenshot shows the Postman application interface. In the left sidebar, under 'Collections', there is a section for 'Spring Boot' which contains a 'GET New Request'. The main workspace shows a 'GET' request to 'http://localhost:8080/api/users/1'. The 'Body' tab displays the JSON response received from the server, which includes fields like id, firstName, lastName, and email.

```
1 {  
2   "id": 1,  
3   "firstName": "Fredluis",  
4   "lastName": "Obidos",  
5   "email": "fobidos.a12241571@umak.edu.ph"  
6 }
```

Update User REST API:

The screenshot shows the Postman application interface. In the left sidebar, under 'Collections', there is a section for 'Spring Boot' which contains a 'PUT New Request'. The main workspace shows a 'PUT' request to 'http://localhost:8080/api/users/1'. The 'Body' tab displays the JSON payload being sent to the server, which updates the user's email address.

```
1 {  
2   "firstName": "Fredluis",  
3   "lastName": "Obidos",  
4   "email": "fredluisobidos@gmail.com"  
5 }
```

Get All Users REST API:

The screenshot shows the Postman interface with a 'Spring Boot / New Request' collection selected. A 'GET' request is being made to `http://localhost:8080/api/users`. The response status is 200 OK, and the body contains two user objects in JSON format:

```
2 |   {
3 |     "id": 1,
4 |     "firstName": "Fredluis",
5 |     "lastName": "Obidos",
6 |     "email": "fredluisobidos@gmail.com"
7 |   },
8 |   {
9 |     "id": 2,
10 |    "firstName": "Obidos",
11 |    "lastName": "Fredluis",
12 |    "email": "obidos@gmail.com"
13 | }
```

Delete User REST API:

The screenshot shows the Postman interface with a 'Spring Boot / New Request' collection selected. A 'DELETE' request is being made to `http://localhost:8080/api/users/1`. The response status is 200 OK, and the body contains the message: 'User successfully deleted!'