

Sicat, Sean Russel

IV-ACSAD

REST API

Source Code:

1. UserController.java

```
1 package net.javaguides.springboottest.controller;
2
3 import lombok.AllArgsConstructor;
4 import net.javaguides.springboottest.entity.User;
5 import net.javaguides.springboottest.service.UserService;
6 import org.springframework.http.HttpStatus;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.web.bind.annotation.*;
9
10 import java.util.List;
11
12 @RestController
13 @AllArgsConstructor
14 @RequestMapping("api/users")
15 public class UserController {
16
17     private UserService userService;
18
19     // build create User REST API
20     @PostMapping
21     public ResponseEntity<User> createUser(@RequestBody User user){
22         User savedUser = userService.createUser(user);
23         return new ResponseEntity<>(savedUser, HttpStatus.CREATED);
24     }
25
26     // build get user by id REST API
27     // http://localhost:8080/api/users/{id}
28     @GetMapping("{id}")
29     public ResponseEntity<User> getUserById(@PathVariable("id") Long userId){
30         User user = userService.getUserById(userId);
31         return new ResponseEntity<>(user, HttpStatus.OK);
32     }
33
34     // Build Get All Users REST API
35     // http://localhost:8080/api/users
36     @GetMapping
37     public ResponseEntity<List<User>> getAllUsers(){
38         List<User> users = userService.getAllUsers();
39         return new ResponseEntity<>(users, HttpStatus.OK);
40     }
41
42     // Build Update User REST API
43     @PutMapping("{id}")
44     // http://localhost:8080/api/users/{id}
45     public ResponseEntity<User> updateUser(@PathVariable("id") Long userId,
46                                             @RequestBody User user){
47         user.setId(userId);
48         User updatedUser = userService.updateUser(user);
49         return new ResponseEntity<>(updatedUser, HttpStatus.OK);
50     }
51
52     // Build Delete User REST API
53     @DeleteMapping("{id}")
54     public ResponseEntity<String> deleteUser(@PathVariable("id") Long userId){
55         userService.deleteUser(userId);
56         return new ResponseEntity<>("User successfully deleted!", HttpStatus.OK);
57     }
58 }
```

2. User.java

```
package net.javaguides.springboot.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String firstName;
    @Column(nullable = false)
    private String lastName;
    @Column(nullable = false, unique = true)
    private String email;
}
```

3. UserRepository.java

```
package net.javaguides.springboot.repository;

import net.javaguides.springboot.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
```

4. UserServiceImpl.java

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

import lombok.AllArgsConstructor;
import net.javaguides.springboottest.entity.User;
import net.javaguides.springboottest.repository.UserRepository;
import net.javaguides.springboottest.service.UserService;
import org.apache.logging.log4j.util.Strings;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;

import java.util.List;
import java.util.Objects;
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);
    }
}
```

5. UserService.java

```
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);
}
```

6. application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/user_management
spring.datasource.username=root
spring.datasource.password=

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update

server.port=8080
```

OUTPUT:

1. Create User:

The screenshot shows a Postman request to `http://localhost:8080/api/users` using the `POST` method. The request body is a JSON object representing a user:

```
1 i
2   "firstName": "andreaspirlo",
3   "lastName": "pirlo",
4   "email": "andreaspirlo@gmail.com"
5 }
```

The response status is `201 Created`, and the response body is identical to the request body, indicating a successful creation of the user.

2. Read Single User:

The screenshot shows a Postman request for a GET operation at `http://localhost:8080/api/users/1`. The request body is set to "none". The response status is `200 OK` with a response time of 5 ms, a size of 246 B, and a status message of "User successfully retrieved!". The response body is a JSON object:

```
{  
  "id": 1,  
  "firstName": "paolo",  
  "lastName": "maldini",  
  "email": "paolomaldini@gmail.com"  
}
```

3. Update User:

The screenshot shows a Postman request for a PUT operation at `http://localhost:8080/api/users/3`. The request body is a JSON object:

```
{  
  "id": 3,  
  "firstName": "andreaspirlo",  
  "lastName": "pirlo",  
  "email": "andreaspirlo213@gmail.com"  
}
```

The response status is `200 OK` with a response time of 87 ms, a size of 252 B, and a status message of "User successfully updated!". The response body is a JSON object:

```
{  
  "id": 3,  
  "firstName": "andreaspirlo",  
  "lastName": "pirlo",  
  "email": "andreaspirlo213@gmail.com"  
}
```

4. Delete User

The screenshot shows a Postman request for a DELETE operation at `http://localhost:8080/api/users/1`. The request body is "none". The response status is `200 OK` with a response time of 58 ms, a size of 190 B, and a status message of "User successfully deleted!". The response body is a JSON object:

```
{  
  "message": "User successfully deleted!"  
}
```