


CABUS, CLEMENT HAROLD MIGUEL

IV-ACSAD

CODE:

Springboot Initializer:



Project
☐ Gradle - Groovy ☐ Gradle - Kotlin
☒ **Maven**

Language
☒ **Java** ☐ Kotlin ☐ Groovy

Spring Boot
☐ 4.0.0 (SNAPSHOT) ☒ **4.0.0 (M2)** ☐ 3.5.6 (SNAPSHOT) ☐ 3.5.5
☐ 3.4.10 (SNAPSHOT) ☐ 3.4.9

Project Metadata

Group

net.javaguides

Artifact

springboot-restful-webservices

Name

springboot-restful-webservices

Description

Demo project for Spring Boot Restful Webservices

Package name

net.javaguides.springboot-restful-webservices

Packaging

☒ **Jar** ☐ War

Java

☒ **24** ☐ 21 ☐ 17

- Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>4.0.0-M2</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>net.javaguides</groupId>
  <artifactId>springboot-restful-webservices</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>springboot-restful-webservices</name>
  <description>Demo project for Spring Boot Restful
webservices</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>24</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>com.mysql</groupId>
      <artifactId>mysql-connector-j</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>
    <dependency>
```

```

        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <annotationProcessorPaths>
                    <path>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                    </path>
                </annotationProcessorPaths>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <excludes>
                    <exclude>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                    </exclude>
                </excludes>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>

```

- UserController.java

```

• package net.javaguides.springboot_restful_webservices.controller;

import lombok.AllArgsConstructor;
import net.javaguides.springboot_restful_webservices.entity.User;
import net.javaguides.springboot_restful_webservices.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);
}
}

```

- User.java

```

package net.javaguides.springboot_restful_webservices.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;
}

```

- UserRepository.java

```

package net.javaguides.springboot_restful_webservices.repository;

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

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

```

- UserServiceImpl.java

```

package net.javaguides.springboot_restful_webservices.service.impl;

import lombok.AllArgsConstructor;
import net.javaguides.springboot_restful_webservices.entity.User;
import net.javaguides.springboot_restful_webservices.repository.UserRepository;
import net.javaguides.springboot_restful_webservices.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);
}
}

```

- UserService.java

```

package net.javaguides.springboot_restful_webservices.service;

import net.javaguides.springboot_restful_webservices.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);
}

```

- SpringbootRestfulWebservicesApplication.java

```
package net.javaguides.springboot_restful_webservices;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringbootRestfulWebservicesApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringbootRestfulWebservicesApplication.class,
args);
    }

}
```

- Application.properties

```
spring.application.name=springboot-restful-webservices
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
```

POSTMAN RESULTS:

Create User REST API:

The image shows the Postman application interface. On the left sidebar, under 'My Workspace', there is a collection named 'http://localhost:8080/api/users' with several endpoints: 'GET Get data', 'POST Post data' (selected), 'PUT Update data', and 'DEL Delete data'. The main panel displays the details of the selected 'POST' request to 'http://localhost:8080/api/users'. The request body is a JSON object:

```
{  "firstName": "umesh",  "lastName": "fadatare",  "email": "umesh@gmail.com"}
```

. Below the request details, the response is shown with a status of '201 Created', a response time of '126 ms', and a body size of '245 B'. The response body is a JSON object:

```
{  "id": 1,  "firstName": "umesh",  "lastName": "fadatare",  "email": "umesh@gmail.com"}
```

. The bottom status bar includes various icons for Postbot, Runner, Start Proxy, Cookies, Vault, and Trash.

Get Single User REST API:

The screenshot shows the Postman application interface. On the left sidebar, under 'My Workspace', there is a collection named 'http://localhost:8080/api/users'. It contains four items: 'GET Get data', 'POST Post data', 'PUT Update data', and 'DEL Delete data'. The 'GET Get data' item is selected. The main panel shows the request details for the endpoint 'http://localhost:8080/api/users/1'. The method is 'GET'. The response is a 200 OK status with a response time of 10 ms and a body size of 240 B. The response body is in JSON format and contains the following data:

```
1 {
2   "id": 1,
3   "firstName": "umesh",
4   "lastName": "fadatare",
5   "email": "umesh@gmail.com"
6 }
```

Update User REST API:

The screenshot shows the Postman application interface. On the left sidebar, under 'My Workspace', there is a collection named 'http://localhost:8080/api/users'. It contains four items: 'GET Get data', 'POST Post data', 'PUT Update data', and 'DEL Delete data'. The 'PUT Update data' item is selected. The main panel shows the request details for the endpoint 'http://localhost:8080/api/users/1'. The method is 'PUT'. The request body is in JSON format and contains the following data:

```
1 {
2   "firstName": "umesh",
3   "lastName": "fadatare",
4   "email": "umesh@yahoo.com"
5 }
```

The response is a 200 OK status with a response time of 39 ms and a body size of 240 B. The response body is in JSON format and contains the following data:

```
1 {
2   "id": 1,
3   "firstName": "umesh",
4   "lastName": "fadatare",
5   "email": "umesh@yahoo.com"
6 }
```

Get All Users REST API:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/users` successfully executed. The response is a 200 OK status with a response time of 9 ms and a body size of 242 B. The response body is displayed in JSON format, showing a single user object.

Request:

- Method: GET
- URL: `http://localhost:8080/api/users`

Response:

200 OK • 9 ms • 242 B

```
1 [
2   {
3     "id": 1,
4     "firstName": "umesh",
5     "lastName": "fadatara",
6     "email": "umesh@yahoo.com"
7   }
8 ]
```

The interface also shows a sidebar with 'My Workspace' containing a collection named 'http://localhost:8080/api/users' with four items: 'GET Get data', 'POST Post data', 'PUT Update data', and 'DEL Delete data'. The bottom status bar includes options like 'Online', 'Find and replace', 'Console', 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', and 'Trash'.

Delete User REST API:

The screenshot displays the Postman interface for a REST API client. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows 'My Workspace' with a collection named 'http://localhost:8080/api/users' containing four methods: GET, POST, PUT, and DELETE. The main panel shows a DELETE request to 'http://localhost:8080/api/users/1'. The 'Send' button is visible. Below the request, the 'Query Params' table is empty. The 'Body' tab is selected, showing a 'Raw' response with a status of '200 OK', a time of '28 ms', and a size of '190 B'. The response body contains the message '1 User successfully deleted!'. The bottom status bar shows 'Online', 'Find and replace', 'Console', 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', and 'Trash'.

My Workspace **New** **Import** DEL http://localhost:8080/api/users/1

Search Postman Ctrl K

Invite Upgrade

My Workspace **New** **Import** DEL http://localhost:8080/api/users/1

Search collections

http://localhost:8080/api/users

GET Get data

POST Post data

PUT Update data

DEL Delete data

Save Share

DELETE http://localhost:8080/api/users/1 Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (5) Test Results 200 OK 28 ms 190 B

Raw Preview Visualize

1 User successfully deleted!

Online Find and replace Console Postbot Runner Start Proxy Cookies Vault Trash