

CABUS, CLEMENT HAROLD MIGUEL

IV-ACSAD

CODE:

Springboot Initializer:

The screenshot shows the Spring Initializr web application interface. At the top, there's a logo consisting of a green leaf icon followed by the text "spring initializr". Below the logo, there are two main sections: "Project" and "Language". Under "Project", "Maven" is selected. Under "Language", "Java" is selected. In the "Spring Boot" section, "4.0.0 (M2)" is selected. The "Project Metadata" section contains the following fields with their current values:

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	<input checked="" type="radio"/> Jar <input type="radio"/> War
Java	<input checked="" type="radio"/> 24 <input type="radio"/> 21 <input type="radio"/> 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 --&gt;
  &lt;/parent&gt;
  &lt;groupId&gt;net.javaguides&lt;/groupId&gt;
  &lt;artifactId&gt;springboot-restful-webservices&lt;/artifactId&gt;
  &lt;version&gt;0.0.1-SNAPSHOT&lt;/version&gt;
  &lt;name&gt;springboot-restful-webservices&lt;/name&gt;
  &lt;description&gt;Demo project for Spring Boot Restful
webservices&lt;/description&gt;
  &lt;url/&gt;
  &lt;licenses&gt;
    &lt;license/&gt;
  &lt;/licenses&gt;
  &lt;developers&gt;
    &lt;developer/&gt;
  &lt;/developers&gt;
  &lt;scm&gt;
    &lt;connection/&gt;
    &lt;developerConnection/&gt;
    &lt;tag/&gt;
    &lt;url/&gt;
  &lt;/scm&gt;
  &lt;properties&gt;
    &lt;java.version&gt;24&lt;/java.version&gt;
  &lt;/properties&gt;
  &lt;dependencies&gt;
    &lt;dependency&gt;
      &lt;groupId&gt;org.springframework.boot&lt;/groupId&gt;
      &lt;artifactId&gt;spring-boot-starter-data-jpa&lt;/artifactId&gt;
    &lt;/dependency&gt;
    &lt;dependency&gt;
      &lt;groupId&gt;org.springframework.boot&lt;/groupId&gt;
      &lt;artifactId&gt;spring-boot-starter-web&lt;/artifactId&gt;
    &lt;/dependency&gt;

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

```

<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.Data;
import lombok.NoArgsConstructor;

```

```
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 screenshot shows the Postman application interface. On the left, the sidebar displays 'My Workspace' with collections, environments, flows, and history. The main workspace shows a POST request to 'http://localhost:8080/api/users'. The request body is set to raw JSON, containing the following data:

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

The response status is 201 Created, with a response time of 126 ms and a response size of 245 B. The response body is identical to the request body.

## Get Single User REST API:

The screenshot shows the Postman application interface. On the left, the sidebar has 'My Workspace' selected, showing collections, environments, flows, and history. A collection named 'http://localhost:8080/api/users' is expanded, revealing four methods: 'GET Get data' (selected), 'POST Post data', 'PUT Update data', and 'DEL Delete data'. The main workspace shows a GET request to 'http://localhost:8080/api/users/1'. The 'Params' tab is active, showing a single parameter 'Key' with 'Value' 'Key'. The 'Body' tab shows a JSON response with the following data:

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

## Update User REST API:

The screenshot shows the Postman application interface. The sidebar and collection structure are identical to the previous screenshot. The main workspace shows a PUT request to 'http://localhost:8080/api/users/1'. The 'Body' tab is active, showing a JSON payload with the following data:

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

The 'Headers' tab shows five headers: 'Content-Type' (application/json), 'Accept' (application/json), 'User-Agent' (Postman/8.0.11), 'Host' (localhost:8080), and 'Connection' (keep-alive). The 'Body' tab also includes options for 'none', 'form-data', 'x-www-form-urlencoded', 'raw' (selected), 'binary', 'GraphQL', and 'JSON'. The response at the bottom is a 200 OK status with a 39 ms duration and 240 B size.

## Get All Users REST API:

The screenshot shows the Postman application interface. On the left, the sidebar includes 'My Workspace' (Collections, Environments, Flows, History), a search bar, and tabs for 'New' and 'Import'. The main area displays a collection named 'http://localhost:8080/api/users' with four methods: 'GET Get data', 'POST Post data', 'PUT Update data', and 'DELETE Delete data'. A specific GET request is selected, showing the URL 'http://localhost:8080/api/users' and method 'GET'. The 'Params' tab is active, showing a single parameter 'Key' with 'Value' and 'Description' columns. Below this is a table for 'Query Params'. The 'Body' tab is selected, showing a JSON response structure. The response body is displayed as:

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

The status bar at the bottom indicates a 200 OK response with 9 ms duration and 242 B size.

## Delete User REST API:

The screenshot shows the Postman application interface. On the left, the sidebar includes 'My Workspace' (selected), 'Collections', 'Environments', 'Flows', and 'History'. The main workspace displays a collection named 'http://localhost:8080/api/users' containing four methods: 'GET Get data', 'POST Post data', 'PUT Update data', and 'DELETE Delete data'. A specific DELETE request is selected, with the URL 'http://localhost:8080/api/users/1' and the method set to 'DELETE'. The 'Params' tab is active, showing a single parameter 'Key' with a value 'Value'. The 'Body' tab is selected, showing the response body: 'User successfully deleted!'. The status bar at the bottom indicates a '200 OK' response with a '28 ms' duration and '190 B' size.

My Workspace

New Import

DEL http://localhost:8080/api/users/1

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

Params Authorization Headers (8) Body Scripts Settings

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (5) Test Results

Raw Preview Visualize

200 OK • 28 ms • 190 B

User successfully deleted!