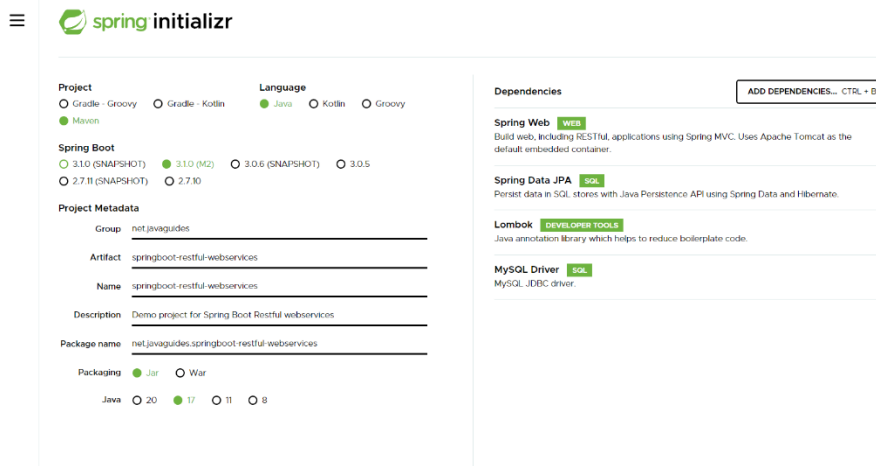


Name: Jose B. Gayares III **Section:** III-ACDS

Create a simple REST API Web Service that can perform CRUD operation, below are the options you can choose to build your own first REST API

1. You can follow this step by step procedure using this tutorial guide. --
> <https://www.javaguides.net/2022/09/spring-boot-rest-api-crud-example-with-mysql-database.html>
2. You can opt to use SoapUI.
3. You can also use Postman as well.

1. Create a Spring Boot Application and Import in IntelliJ IDEA



The Spring Initializr web interface is shown with the following configuration:

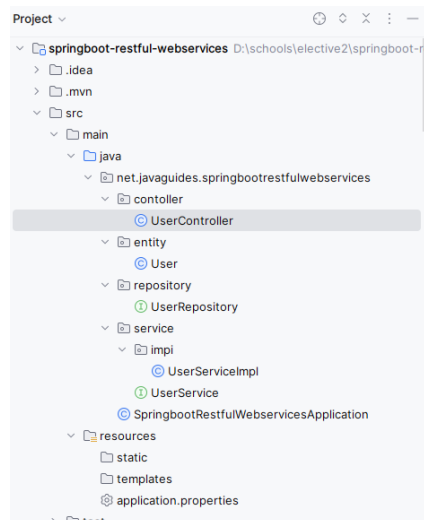
- Project:** ☒ Gradle, ☐ Groovy, ☐ Maven
- Language:** ☒ Java, ☐ Kotlin, ☐ Groovy
- Spring Boot:** ☐ 3.1.0 (SNAPSHOT), ☒ 3.1.0 (M2), ☐ 3.0.6 (SNAPSHOT), ☐ 3.0.5, ☐ 2.7.11 (SNAPSHOT), ☐ 2.7.10
- Project Metadata:**
 - Group: net.javaguides
 - Artifact: springboot-restful-webservices
 - Name: springboot-restful-webservices
 - Description: Demo project for Spring Boot Restful webservices
 - Package name: net.javaguides.springboot-restful-webservices
- Packaging:** ☒ Jar, ☐ War
- Java:** ☐ 20, ☒ 17, ☐ 11, ☐ 8

Dependencies:

- Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
- Spring Data JPA** (SQL): Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
- Lombok** (DEVELOPER TOOLS): Java annotation library which helps to reduce boilerplate code.
- MySQL Driver** (SQL): MySQL JDBC driver.

ADD DEPENDENCIES... CTRL + B

2. Project Structure



3. Configuring MySQL Database

```
application.properties ×  SpringbootRestfulWebservicesApplication.java  UserControlli
1  spring.datasource.url=jdbc:mysql://localhost:3306/user_management
2  spring.datasource.username=root
3  spring.datasource.password=
4
5  spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
6  spring.jpa.hibernate.ddl-auto=update
7  |
```

4. Create JPA Entity - User.java

```
public class User {

    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;|
    no usages
    @Column(nullable = false)
    private String firstName;
    no usages
    @Column(nullable = false)
    private String lastName;
    no usages
    @Column(nullable = false, unique = true)
    private String email;
}
```

5. Create Spring Data JPA Repository for User JPA Entity

```
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> {
}
```

6. Service Layer Implementation

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

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

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

1 usage
@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;
}

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

```
package net.javaguides.springbootrestfulwebservices.service;

import net.javaguides.springbootrestfulwebservices.entity.User;
import java.util.List;

4 usages 1 implementation
public interface UserService {

    1 usage 1 implementation
    User createUser(User user);

    1 usage 1 implementation
    User getUserById(Long userId);

    1 usage 1 implementation
    List<User> getAllUsers();

    1 usage 1 implementation
    User updateUser(User user);

    1 usage 1 implementation
    void deleteUser(Long userId);
}
```

7. Creating UserController - Building CRUD Rest APIs7.

Creating UserController - Building CRUD Rest APIs

```
5 usages
private UserService userService;

// build create User REST API
no usages
@PostMapping
public ResponseEntity<User> createUser(@RequestBody User user){
    User savedUser = userService.createUser(user);
    return new ResponseEntity<>(savedUser, HttpStatus.CREATED);
}

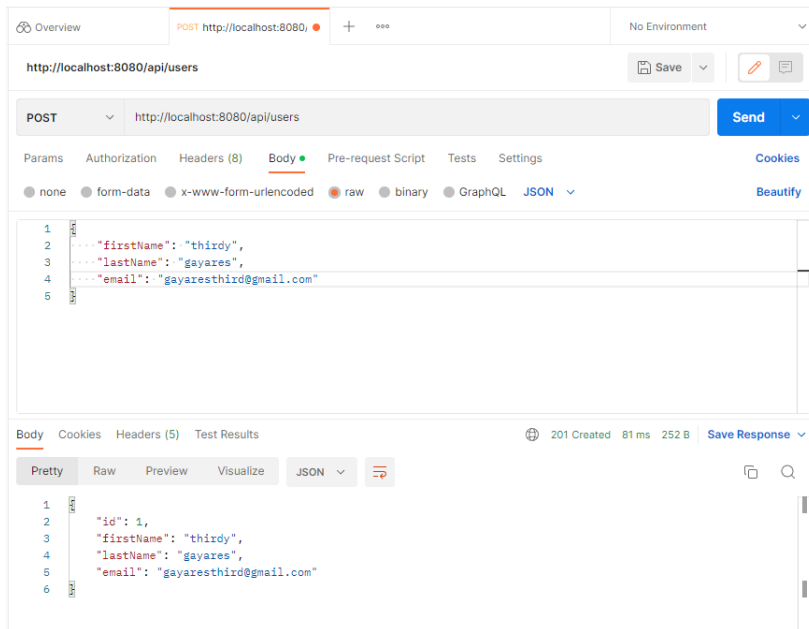
// build get user by id REST API
// http://localhost:8080/api/users/1
no usages
@GetMapping("/{id}")
public ResponseEntity<User> getUserById(@PathVariable("id") Long userId){
    User user = userService.getUserById(userId);
    return new ResponseEntity<>(user, HttpStatus.OK);
}

// Build Get All Users REST API
// http://localhost:8080/api/users
no usages
@GetMapping
public ResponseEntity<List<User>> getAllUsers(){
    List<User> users = userService.getAllUsers();
    return new ResponseEntity<>(users, HttpStatus.OK);
}

// Build Update User REST API
```

9. Test Spring Boot CRUD REST APIs using Postman Client

Create User REST API:



Checking to my server

	id	email	first_name	last_name
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	gayaresthird@gmail.com	thirdy	gayares

Get Single User REST API:

Overview GET http://localhost:8080/api/users/1 No Environment

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

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

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK 37 ms 247 B Save Response

Pretty Raw Preview Visualize JSON

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

Update User REST API:

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

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

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautiful

```
1 {
2   "firstName": "jose",
3   "lastName": "gayares",
4   "email": "jgayares.a12034879@umak.edu.ph"
5 }
```

Body Cookies Headers (5) Test Results 200 OK 20 ms 253 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "firstName": "jose",
4   "lastName": "gayares",
5   "email": "jgayares.a12034879@umak.edu.ph"
6 }
```

Checking to my server

	id	email	first_name	last_name
<input type="checkbox"/> Edit Copy Delete	1	jgayares.a12034879@umak.edu.ph	jose	gayares

Get All Users REST API:

http://localhost:8080/api/users

GET ▼ http://localhost:8080/api/users Send ▼

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
-----	-------	-------------	-----	-----------

body Cookies Headers (5) Test Results 200 OK 15 ms 422 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼

```
1 {
2   "id": 1,
3   "firstName": "jose",
4   "lastName": "gayares",
5   "email": "jgayares.a12034879@umak.edu.ph"
6 },
7 {
8   "id": 2,
9   "firstName": "marielle",
10  "lastName": "zabala",
11  "email": "mariellezbl@gmail.com"
12 },
13 {
14   "id": 3,
15   "firstName": "novem",
16   "lastName": "lanaban",
17   "email": "novemlanaban@gmail.com"
18 }
19 }
20 }
```

Delete User REST API:

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

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

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body Cookies Headers (5) Test Results 200 OK 34 ms 190 B Save Response ▼

Pretty Raw Preview Visualize Text ▼

```
1 User successfully deleted!
```