LAB 4

CREATE RESTFUL WEB SERVICE

➢ CONTENT

- Introduce about web services
- Create RESTful web services with Java Spring Boot

❖ PREREQUISITES

- Install 1 of 3 below tools (or other alternatives) to test API
    - Postman: Desktop app
    - Talend API Tester: Chrome extension
    - Thunder Client: VS Code extension

❖ INTRODUCTION

1. *Formats of APIs:*
    - XML: eXtensible Markup Language
    - JSON: JavaScript Object Notation
2. *Types of web services:*
    - SOAP: stands for Simple Object Access Protocol. SOAP is an XML based industry standard protocol for designing and developing web services
    - REST: stands for Representational State Transfer. REST is an architectural style for developing web services

3. *REST architectural style:*
    - REST is a simple way to organize interactions between independent systems. It is simpler than complex mechanisms such as RPC or SOAP
    - Properties: simplicity, modifiability, visibility, portability, reliability
4. *RESTful web service:*
    - REST architecture-based web services
    - Lightweight, maintainable, scalable
    - Used to create APIs for web-based application
    - The calling client can perform operations using RESTful service
    - The protocol for REST is HTTP

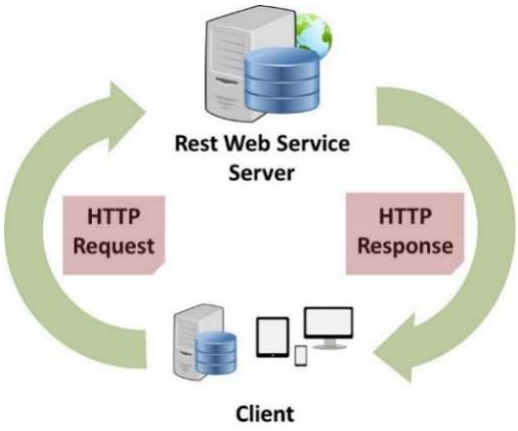

Figure 1 - RESTful Web Service Architecture

- **@GET, @PUT, @POST, @DELETE**: used to specify the HTTP request type for a method

	SQL	HTTP Method	HTTP Status code
CREATE	Insert	POST	201
READ	Select	GET	200
UPDATE	Update	PUT	200
DELETE	Delete/ Drop	DELETE	204

Figure 2 - CRUD (SQL - HTTP)

➢ INSTRUCTION

1. Create new Java Spring Boot project with dependencies

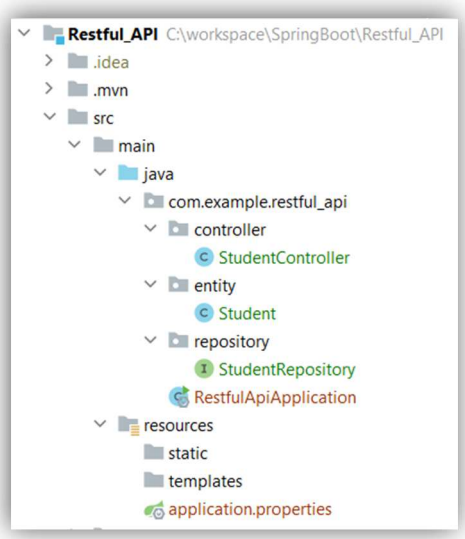

Figure 3 – Sample project structure

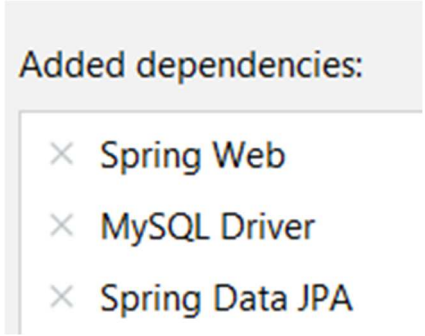

Figure 4 - Project dependencies

2. Config MySQL connection, JPA & Hibernate
3. Create Java class for model (entity) which acts as table in database

```
@Entity
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;
    private String name;
    private int age;

    // auto generated getter & setter
```

*Figure 5 - **Student.java***

4. Create Java interface which extends *JpaRepository*
5. Create Java class for Restful controller to create Restful APIs

```java
@RestController
public class StudentController {
    @Autowired
    StudentRepository studentRepository;

    @GetMapping(value = "/")
    public List<Student> viewStudentList() {
        return studentRepository.findAll();
    }

    @GetMapping(value = "/detail/{id}")
    public Student viewStudentById(
            @PathVariable (value = "id") Long id) {
        return studentRepository.findById(id).get();
    }
```

*Figure 6 - **StudentController.java (1)***

```java
    @PostMapping(value = "/add")
    public Student addStudent(
            @RequestBody Student student) {
        return studentRepository.save(student);
    }

    @PutMapping(value = "/update/{id}")
    public void updateStudent(
            @PathVariable(value = "id") Long id,
            @RequestBody Student student) {
        if (studentRepository.existsById(id)) {
            student.setId(id);
            studentRepository.save(student);
        }
    }

    @DeleteMapping(value = "/delete/{id}")
    public void deleteStudent(
            @PathVariable(value = "id") Long id) {
        if (studentRepository.existsById(id)) {
            Student student = studentRepository.getById(id);
            studentRepository.delete(student);
        }
    }
}
```

*Figure 7 - **StudentController.java (2)***

6. Populate sample data in database using MySQL Workbench or integrated MySQL database in IntelliJ

id	age	name
1	18	Minh
2	20	Hung
3	22	Lien

Figure 8 - Populate sample data in database

7. Run the web application then test Restful APIs with Postman

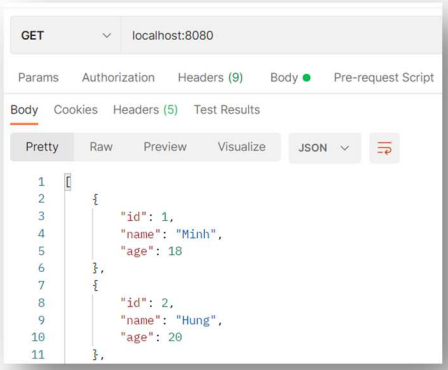

Figure 9 - View all students (GET method)

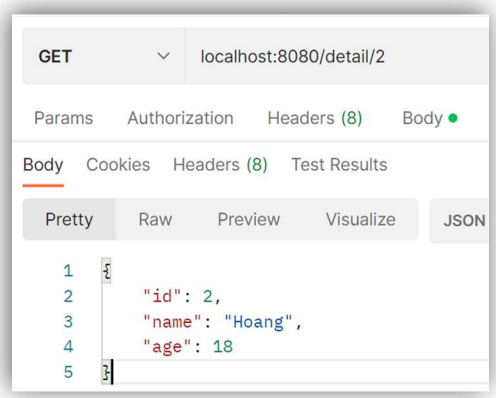

Figure 10 - View student by ID (GET method)

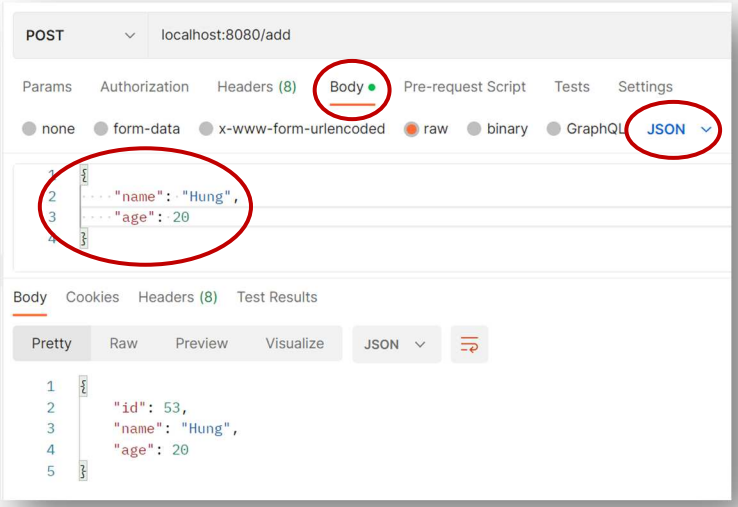

Figure 11 - Add new student (POST method)

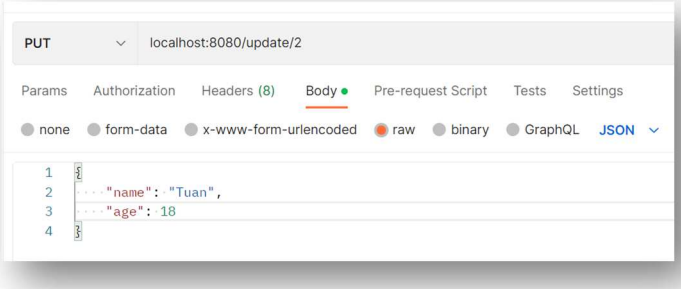

Figure 12 - Update student (PUT method)

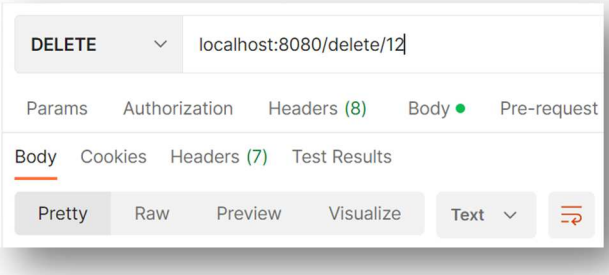

Figure 13 - Delete student (DELETE method)

➢ TO-DO

- Add data validation for Entity & Controller then re-test with Postman
- Create similar Restful APIs for other Spring Boot projects