MỤC TIÊU

- Xây dựng REST Consumer với Java thuần sử dụng HttpURLConnection API
- Xây dựng Spring Boot REST API cơ bản
- Kết hợp Spring Boot REST API với CSDL

BÀI 1: SỬ DỤNG REST API VỚI HTTPURLCONNECTION (2 ĐIỂM)

Hãy viết mã Java thuần tương tác với REST API thực hiện các hoạt động cơ bản:

- ✓ GET ALL: (host+"/students.json")
- ✓ GET BY KEY: (host+"/students/<<key>>.json")
- ✓ POST: (host+"/students.json", {id:?, name:?, mark:?, gender:?})
- ✓ PUT: (host+"/students/<<key>>.json", {id:?, name:?, mark:?, gender:?})
- ✓ DELETE: (host+"/students/<<key>>.json")

Hướng dẫn thực hiện:

1. Xây dựng lớp tiện ích HttpClient tương tác với web server

```
public class HttpClient {
        /**
         * Mở kết nối
         *
         * @param method là web method (GET, POST, PUT hay DELETE)
         * @param url địa chỉ URL của REST API
         */
        public static HttpURLConnection openConnection(String method, String url)
                                                        throws IOException {...}

        /**
         * Đọc dữ liệu phản hồi từ server và đóng kết nối
         */
        public static byte[] readData(HttpURLConnection connection)
                                                        throws IOException {...}

        /**
         * Gửi dữ liệu lên server và đọc dữ liệu phản hồi từ server
         */
        public static byte[] writeData(HttpURLConnection connection, byte[] data)
                                                        throws IOException {...}
}
```

Cài đặt mã nguồn cho HttpClient theo hướng dẫn sau

```
openConnection(method, url)
```

```
var connection = (HttpURLConnection) new URL(url).openConnection();
connection.setRequestProperty("Content-Type", "application/json; charset=utf-8");
connection.setRequestMethod(method);
return connection;
```

readData(connection)

```
if (connection.getResponseCode() == 200) {
        var out = new ByteArrayOutputStream();
        var is = connection.getInputStream();
        var block = new byte[4 * 1024];
        while (true) {
                int n = is.read(block);
                if (n <= 0) {
                        break;
                }
                out.write(block, 0, n);
        }
        connection.disconnect();
        return out.toByteArray();
}
connection.disconnect();
throw new IOException("No response from server!");
```

writeData(connection, data)

```
connection.setDoOutput(true);
connection.getOutputStream().write(data);
return readData(connection);
```

2. Xây dựng lớp RestClient bao gồm các phương thức sau đây để thực hiện các hoạt động tương thác với REST API

```
public class RestClient {
        static String host = "https://fpolyedu.firebaseio.com";

        private static void getAll() {}
        private static void getByKey() {}
        private static void post() {}
        private static void put() {}
        private static void delete() {}
}
```

Cài đặt mã cho RestClient theo hướng dẫn sau

```
getAll(): GET(url)
```

```
var url = host + "/students.json";
try {
        var connection = HttpClient.openConnection("GET", url);
        var response = HttpClient.readData(connection);
        System.out.println(new String(response));
} catch (IOException ex) {
        System.out.println(ex.getMessage());
}
```

getByKey(): GET(url)

```
var url = host + "/students/<<key>>.json";
try {
        var connection = HttpClient.openConnection("GET", url);
        var response = HttpClient.readData(connection);
        System.out.println(new String(response));
} catch (IOException ex) {
        System.out.println(ex.getMessage());
}
```

post(): POST(url, data)

```
var url = host + "/students.json";
var data = "{\"id\": \"SV09\",\"name\": \"Sinh viên 09\",\"mark\": 5.5,\"gender\": true}".getBytes();
try {
        var connection = HttpClient.openConnection("POST", url);
        var response = HttpClient.writeData(connection, data);
        System.out.println(new String(response));
} catch (IOException ex) {
        System.out.println(ex.getMessage());
}
```

put(): PUT(url, data)

```
var url = host + "/students/<<key>>.json";
var data = "{\"id\": \"SV09\",\"name\": \"Sinh viên 09\",\"mark\": 5.5,\"gender\": true}".getBytes();
try {
        var connection = HttpClient.openConnection("PUT", url);
        var response = HttpClient.writeData(connection, data);
        System.out.println(new String(response));
} catch (IOException ex) {
        System.out.println(ex.getMessage());
}
```

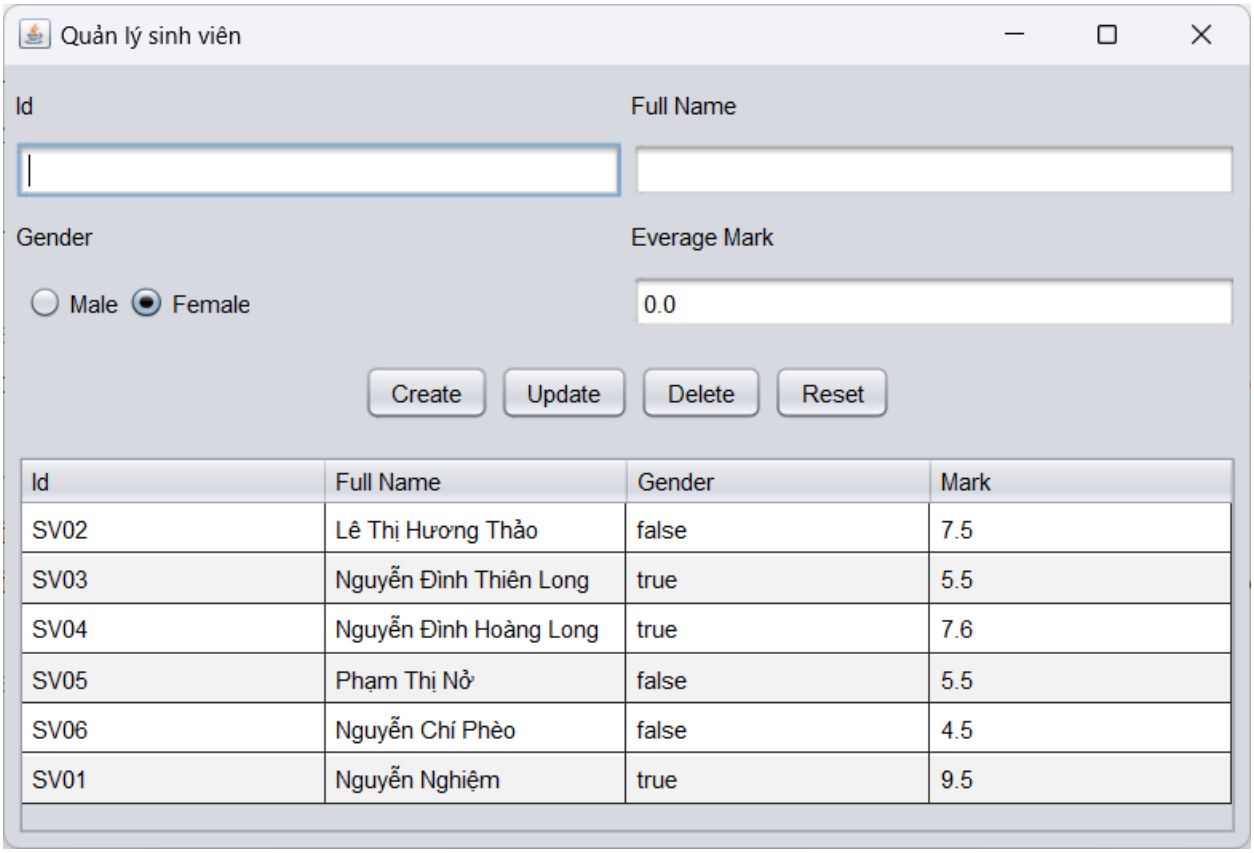
delete(): DELETE(url)

```
var url = host + "/students/<<key>>.json";
try {
        var connection = HttpClient.openConnection("DELETE", url);
        HttpClient.readData(connection);
} catch (IOException ex) {
        System.out.println(ex.getMessage());
}
```

3. Chạy và kiểm thử
   - ✓ Thay thế <<key>> phù hợp
   - ✓ Bổ sung phương thức main() và gọi từng phương thức để kiểm thử các hoạt động cơ bản của REST API

BÀI 2: XÂY DỰNG ỨNG DỤNG CRUD VỚI SWING+REST API (3 ĐIỂM)

Xây dựng ứng dụng Desktop Application quản lý sinh viên có giao diện và hoạt động tương tác được mô tả như sau:



TƯƠNG TÁC	XỬ LÝ
Khởi đầu	• Tải và hiển thị danh sách sinh viên bảng
Edit.DoubleClick	• Tải và hiển thị sinh viên lên form

Create.Click	• Thêm mới sinh viên • Xóa trắng form • Tải và hiển thị danh sách sinh viên bảng
Update.Click	• Cập nhật sinh viên • Tải và hiển thị danh sách sinh viên bảng
Delete.Click	• Xóa sinh viên • Xóa trắng form • Tải và hiển thị danh sách sinh viên bảng
Reset.Click	• Xóa trắng form

Hướng dẫn thực hiện:

- Cần sử dụng thư viên Jackson để chuyển đổi giữa JSON và Java Object. Khai báo thư viện phụ thuộc sau đây vào pom.xml

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.18.3</version>
</dependency>
```

- Cần phải tạo 2 lớp Student và StudentMap để mô tả cấu trúc dữ liệu JSON

Student: mô tả cấu trúc dữ liệu của sinh viên

```
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Data
public class Student {
    private String id;
    private String name;
    private double mark;
    private boolean gender;
}
```

StudentMap: mô tả cấu trúc dữ liệu của {key1:student1, key2:student2,…}

```
public class StudentMap extends HashMap<String, Student>{
}
```

- Sau khi đọc dữ liệu phản hồi từ REST API bạn cần chuyển đổi sang đối tượng Java

```
var response = HttpClient.readData(connection);
var mapper = new ObjectMapper();
// Chuyển JSON thành Student
```

```
var student = mapper.readValue(response, Student.class);
// Chuyển JSON thành StudentMap
var studentMap = mapper.readValue(response, StudentMap.class);
```

- Cần chuyển đổi đối tượng Student sang JSON trước khi gửi đến REST API

```
var student = new Student("SV01", "Tèo", true, 3.5);
var mapper = new ObjectMapper();
var data = mapper.writeValueAsBytes(student);
var response = HttpClient.writeData(connection, data);
```

BÀI 3: XÂY DỰNG REST API VỚI SPRING BOOT CƠ BẢN (2 ĐIỂM)

Hãy sử dụng Spring Boot để xây dựng REST API có các hoạt động tương tác cơ bản tương tự như Firebase REST API quản lý sinh viên. Trên cơ sở đó xây dựng ứng dụng CRUD quản lý sinh viên tương tự Lab4.3.

Chú ý: Firebase REST API quản lý sinh viên

- Đặt tại HOST https://fpolyedu.firebaseio.com
- Cấu trúc dữ liệu sinh viên {id:?, name:?, gender:?, mark:?}
- Các hoạt động cơ bản

REQUEST	RESPONSE
GET(host/students.json)	{key1: student, key2: student,…}
GET(host/students/<<key>>.json)	student
POST(host/students.json, student)	{name: key}
PUT(host/students/<<key>>.json, student)	student
DELETE(host/students/<<key>>.json)	

Hướng dẫn thực hiện:

- Tạo dự án Spring Boot và tạo RestController có tên StudentRestApi được tổ chức như sau:

```
@CrossOrigin("*")
@RestController
public class StudentRestApi {
	@GetMapping("students.json") // => {key1: student, key2: student,...}
	public Map<String, Student> findAll(){}

	@GetMapping("students/{key}.json") // => student
	public Student findByKey(@PathVariable("key") String key){}

	@PostMapping("students.json") // => {name: key}
	public Map<String, String> create(@RequestBody Student student){}

	@PutMapping("students/{key}.json") // => student
	public Student update(@PathVariable("key") String key, @RequestBody Student student){}

	@DeleteMapping("students/{key}.json") // nothing
	public void delete(@PathVariable("key") String key){}
}
```

- Cài đặt mã cho StudentRestApi
  - Tạo lớp tiện ích Database chứa dữ liệu mô phỏng được sử dụng trong StudentRestApi

```
public class Database {
	// Chứa danh sách sinh viên
	public static Map<String, Student> map = new HashMap<>();
	// Bổ sung dữ liệu mẫu vào map
	static {
		map.put(getKey(), new Student("SV01", "Lý Thái Tổ", true, 9.5));
		map.put(getKey(), new Student("SV02", "Lê Trọng Tấn", true, 4.5));
		map.put(getKey(), new Student("SV03", "Nguyễn Thị Minh Khai", false, 8.5));
		map.put(getKey(), new Student("SV04", "Đoàn Trung Trực", true, 6.0));
	}
	// Lấy key ngẫu nhiên
	public static String getKey() {
		return Integer.toHexString(UUID.randomUUID().toString().hashCode());
	}
}
```

  - Cài đặt mã cho các phương thức điều khiển của StudentRestApi

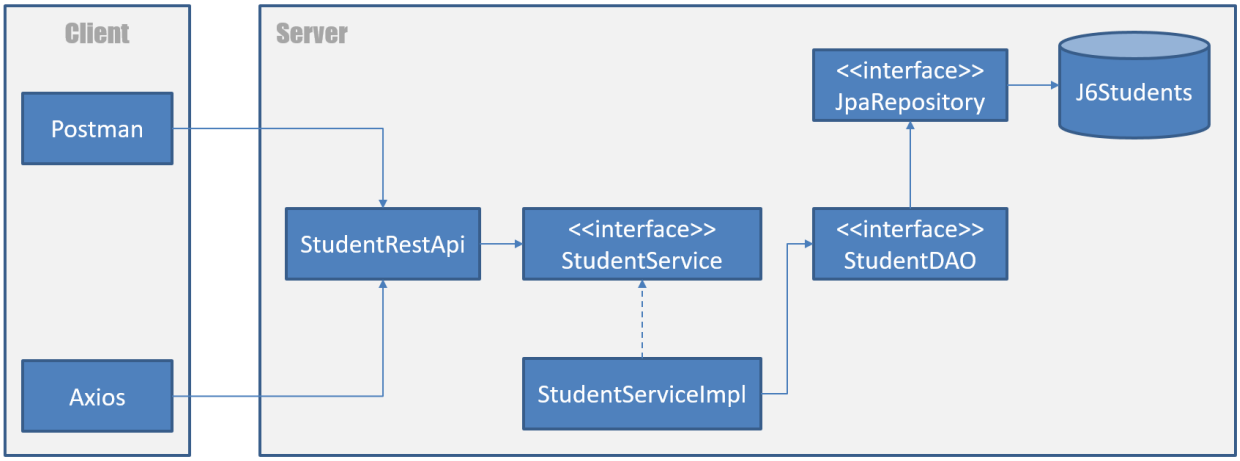
findAll()
return Database.*map*;
findByKey()
return Database.*map*.get(key);
Create()
var key = Database.*getKey*(); Database.*map*.put(key, student);

return Map.*of*("name", key);
Update()
Database.*map*.put(key, student); return Database.*map*.get(key);
Delete()
Database.*map*.remove(key);

- Chạy và kiểm thử các hoạt động cơ bản của REST API đặt tại host http://localhost:8080
- Xây dựng ứng dụng CRUD quản lý sinh viên (Tương tự Lab4.3) với StudentRestApi

BÀI 4: XÂY DỰNG REST API VỚI SPRING BOOT NÂNG CAO (2 ĐIỂM)

Yêu cầu: Xây dựng REST API làm việc với CSDL để quản lý sinh viên theo mô hình:



Hướng dẫn thực hiện:

1. Xây dựng CSDL

```
CREATE TABLE J6Students(
        Id VARCHAR(50) NOT NULL PRIMARY KEY,
        Name NVARCHAR(50) NOT NULL,
        Mark FLOAT NOT NULL,
        Gender BIT NOT NULL
)
INSERT INTO J6Students VALUES('SV001', N'Lý Thái tổ', 9.5, 1);
INSERT INTO J6Students VALUES('SV002', N'Lê Trọng Tấn', 4.5, 1);
INSERT INTO J6Students VALUES('SV003', N'Nguyễn Thị Minh Khai', 9.5, 0);
INSERT INTO J6Students VALUES('SV004', N'Đoàn Trung Trực', 6.0, 1);
```

2. Xây dựng DAO

- Khai báo thư viện cần thiết

```
<dependency>
        <groupId>org.springframework.boot</groupId>
```

```
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
        <groupId>com.microsoft.sqlserver</groupId>
        <artifactId>mssql-jdbc</artifactId>
        <scope>runtime</scope>
</dependency>
```

- Khai báo các thông số kết nối CSDL

```
spring.datasource.url=???
spring.datasource.username=???
spring.datasource.password=???
spring.datasource.driverClassName=???
spring.jpa.hibernate.dialect=???
spring.jpa.hibernate.ddl-auto = none
spring.jpa.show-sql=false
```

- Tạo lớp thực thể Student

```
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Data
@Entity(name = "J6Students")
public class Student {
        @Id
        String id;
        String name;
        boolean gender;
        double mark;
}
```

- Tạo interface StudentDAO

```
public interface StudentDAO extends JpaRepository<Student, String>{}
```

3. Xây dựng StudentService phục vụ cho StudentRestApi

- Tạo interface StudentService

```
public interface StudentService {
        List<Student> findAll();
        Student findById(String id);
```

```
        Student create(Student student);
        Student update(Student student);
        void deleteById(String id);
}
```

- Tạo lớp StudentService và Cài đặt mã cho StudentService

```
@Service
public class StudentServiceImpl implements StudentService{
        @Autowired
        StudentDAO dao;
        @Override
        public List<Student> findAll() {
                return dao.findAll();
        }
        @Override
        public Student findById(String id) {
                return dao.findById(id).get();
        }
        @Override
        public Student create(Student student) {
                return dao.save(student);
        }
        @Override
        public Student update(Student student) {
                return dao.save(student);
        }
        @Override
        public void deleteById(String id) {
                dao.deleteById(id);
        }
}
```

4. Tạo StudentRestApi

```
@CrossOrigin("*")
@RestController
public class StudentRestApi {
        @Autowired
        StudentService studentService;
```

```
        @GetMapping("students") //=> [student1, student2,...]
        public List<Student> findAll(){
                return studentService.findAll();
        }
        @GetMapping("students/{id}") //=> student
        public Student findById(@PathVariable("id") String id){
                return studentService.findById(id);
        }
        @PostMapping("students") //=> student
        public Student create(@RequestBody Student student){
                return studentService.create(student);
        }
        @PutMapping("students/{id}") //=> student
        public Student update(@PathVariable("id") String id, @RequestBody
Student student){
                return studentService.update(student);
        }
        @DeleteMapping("students/{id}") //=> nothing
        public void delete(@PathVariable("id") String id){
                studentService.deleteById(id);
        }
}
```

5. Chạy REST API và kiểm thử với Postman
    - GET(http://localhost:8080/students)
    - GET(http://localhost:8080/students/<<id>>)
    - POST(http://localhost:8080/students, student)
    - PUT(http://localhost:8080/students/<<id>>, student)
    - DELETE(http://localhost:8080/students/<<id>>)
6. Xây dựng ứng dụng CRUD với StudentRestApi

BÀI 5: GIẢNG VIÊN CHO THÊM (2 ĐIỂM)