


SPRAWOZDANIE NR 7			
Nazwa ćwiczenia	Aplikacja RESTful (Serwer)		 <b>POLITECHNIKA BYDGOSKA</b> Wydział Telekomunikacji, Informatyki i Elektrotechniki
Przedmiot	Zaawansowane programowanie obiektowe		
Student Grupa Nr indeksu	Paweł Jońca gr 7 122348		
Data ćwiczeń		01.07	Data oddania sprawozdania

## Spis treści

Treść zadania .....	1
Rozwiązanie problemu .....	1
Kod do StudentController .....	1
kod do Student .....	2
Kod do ResourceNotFoundException.....	3
kod do RestAppApplication .....	3
Wygląd aplikacji.....	7
Wygląd struktury projektu .....	9
Wnioski .....	9

## Treść zadania

Zaimplementować aplikację RESTful, która będzie umożliwiać pobieranie informacji o studentach za pomocą interfejsu REST API. Kontroler powinien zawierać w tym przypadku tylko dwie metody, `getStudent(Long id)` oraz `getStudents()`, które pozwolą odpowiednio na pobieranie informacji o studencie z określonym identyfikatorem `id` oraz listę wszystkich studentów. W przypadku, gdy klient będzie żądał informacji o studencie, który nie istnieje w bazie danych, powinien dostać informację zwrotną z kodem błędu 404 i komunikatem "( Client with id=... not found ".

## Rozwiązanie problemu

### Kod do StudentController

```
package lab7.Controller;

import lab7.repository.StudentRepository;
import lab7.domain.Student;
import lab7.exception.ResourceNotFoundException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
```

```

import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("/students")
public class StudentController {

    @Autowired
    private StudentRepository studentRepository;

    @GetMapping
    public List<Student> getAllStudents() {
        return studentRepository.findAll();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Student> getStudentById(@PathVariable(value =
"\"id\") Long studentId) {
        Student student = studentRepository.findById(studentId)
            .orElseThrow(() -> new ResourceNotFoundException(": Client
with id=\"" + studentId + "\" not found"));
        return ResponseEntity.ok().body(student);
    }
}

```

### kod do Student

```

package lab7.domain;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
@Entity
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String firstName;
    private String lastName;
    private String studentId;

    public Student() {
    }

    public Student(String firstName, String lastName, String studentId) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.studentId = studentId;
    }

    // Getters and Setters
    public Long getId() {
        return id;
    }
}

```

```

    public void setId(Long id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getStudentId() {
        return studentId;
    }

    public void setStudentId(String studentId) {
        this.studentId = studentId;
    }
}

```

### Kod do ResourceNotFoundException

```

package lab7.exception;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException {

    public ResourceNotFoundException(String message) {
        super(message);
    }
}

```

### kod do RestAppApplication

```

package lab7;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import lab7.domain.Student;
import lab7.repository.StudentRepository;

@SpringBootApplication
public class RestAppApplication {

    public static void main(String[] args) {

```

```

        SpringApplication.run(RestAppApplication.class, args);
    }

    @Bean
    public CommandLineRunner demoData(StudentRepository studentRepository)
    {
        return args -> {
            // Dodawanie przykładowych studentów
            studentRepository.save(new Student("Jan", "Kowalski",
"123456"));
            studentRepository.save(new Student("Anna", "Nowak", "654321"));
            studentRepository.save(new Student("Piotr", "Wójcik",
"987654"));
            studentRepository.save(new Student("Zofia", "Wiśniewska",
"112233"));
        };
    }
}

```

application.properties

```

spring.application.name=RESTapp
spring.datasource.url=jdbc:h2:file:./db/myH2Database
spring.datasource.username=user
spring.datasource.password=1234
spring.datasource.driverClassName=org.h2.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto = update
spring.jpa.show-sql=true
spring.h2.console.enabled=true
spring.h2.console.path=/h2

```

plik 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>3.5.3</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>lab7</groupId>
    <artifactId>RESTapp</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>RESTapp</name>
    <description>RESTapp</description>
    <url/>
    <licenses>
        <license/>
    </licenses>

```

```

</licenses>
<developers>
  <developer/>
</developers>
<scm>
  <connection/>
  <developerConnection/>
  <tag/>
  <url/>
</scm>
<properties>
  <java.version>17</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-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-rest-hal-explorer</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>

```



## Wygląd aplikacji

## Logowanie

English Preferences Tools Help

### Login

Saved Settings: Generic H2 (Embedded) ▼

Setting Name: Generic H2 (Embedded) Save Remove

---

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:file:./db/myH2Database

User Name: user

Password: ....

Connect Test Connection

H2 Console

localhost:8080/h2/login.do?sessionId=2f820a73faade5c0b73c9a9b3e60f7a3

Max rows: 1000 Auto complete Off Auto select On

jdbc:h2:file:./db/myH2Database

STUDENT

INFORMATION\_SCHEMA

Users

H2 2.3.232 (2024-08-11)

Run Run Selected Auto complete Clear SQL statement:

#### Important Commands

?	Displays this Help Page
⌂	Shows the Command History
Ctrl+Enter	Executes the current SQL statement
Shift+Enter	Executes the SQL statement defined by the text selection
Ctrl+Space	Auto complete
⌂	Disconnects from the database

#### Sample SQL Script

Delete the table if it exists	DROP TABLE IF EXISTS TEST;
Create a new table with ID and NAME columns	CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
Add a new row	INSERT INTO TEST VALUES(1, 'Hello');
Add another row	INSERT INTO TEST VALUES(2, 'World');
Query the table	SELECT * FROM TEST ORDER BY ID;
Change data in a row	UPDATE TEST SET NAME='Hi' WHERE ID=1;
Remove a row	DELETE FROM TEST WHERE ID=2;
Help	HELP ...

#### Adding Database Drivers

Additional database drivers can be registered by adding the Jar file location of the driver to the environment variables H2DRIVERS or CLASSPATH. Example (Windows): to add the database driver library C:/Programs/hsqldb/lib/hsqldb.jar.

HAL Explorer Theme Settings About

Edit Headers / Go!

**Links**

Relation	Name	Title	HTTP Request	Doc
students			< > + - > > x	
profile			< > + - > > x	

**Response Status**  
200 (OK)

**Response Headers**

connection	keep-alive
content-type	application/hal+json
date	Mon, 30 Jun 2025 23:37:54 GMT
keep-alive	timeout=60
transfer-encoding	chunked
vary	Origin, Access-Control-Request-Method, Access-Control-Request-Headers

**Response Body**

```
{
  "_links": {
    "students": {
      "href": "http://localhost:8080/students?page,size,sort*",
      "templated": true
    },
    "profile": {
      "href": "http://localhost:8080/profile"
    }
  }
}
```

Pobieram listę studentów

localhost:8080/students

Pretty print

```
[{"id":1,"firstName":"Jan","lastName":"Kowalski","studentId":"123456"}, {"id":2,"firstName":"Anna","lastName":"Nowak","studentId":"654321"}, {"id":3,"firstName":"Piotr","lastName":"Wójcik","studentId":"987654"}, {"id":4,"firstName":"Zofia","lastName":"Wiśniewska","studentId":"112233"}, {"id":5,"firstName":"Jan","lastName":"Kowalski","studentId":"123456"}, {"id":6,"firstName":"Anna","lastName":"Nowak","studentId":"654321"}, {"id":7,"firstName":"Piotr","lastName":"Wójcik","studentId":"987654"}, {"id":8,"firstName":"Zofia","lastName":"Wiśniewska","studentId":"112233"}, {"id":9,"firstName":"Jan","lastName":"Kowalski","studentId":"123456"}, {"id":10,"firstName":"Anna","lastName":"Nowak","studentId":"654321"}, {"id":11,"firstName":"Piotr","lastName":"Wójcik","studentId":"987654"}, {"id":12,"firstName":"Zofia","lastName":"Wiśniewska","studentId":"112233"}]
```

Pobieram dane studenta po id

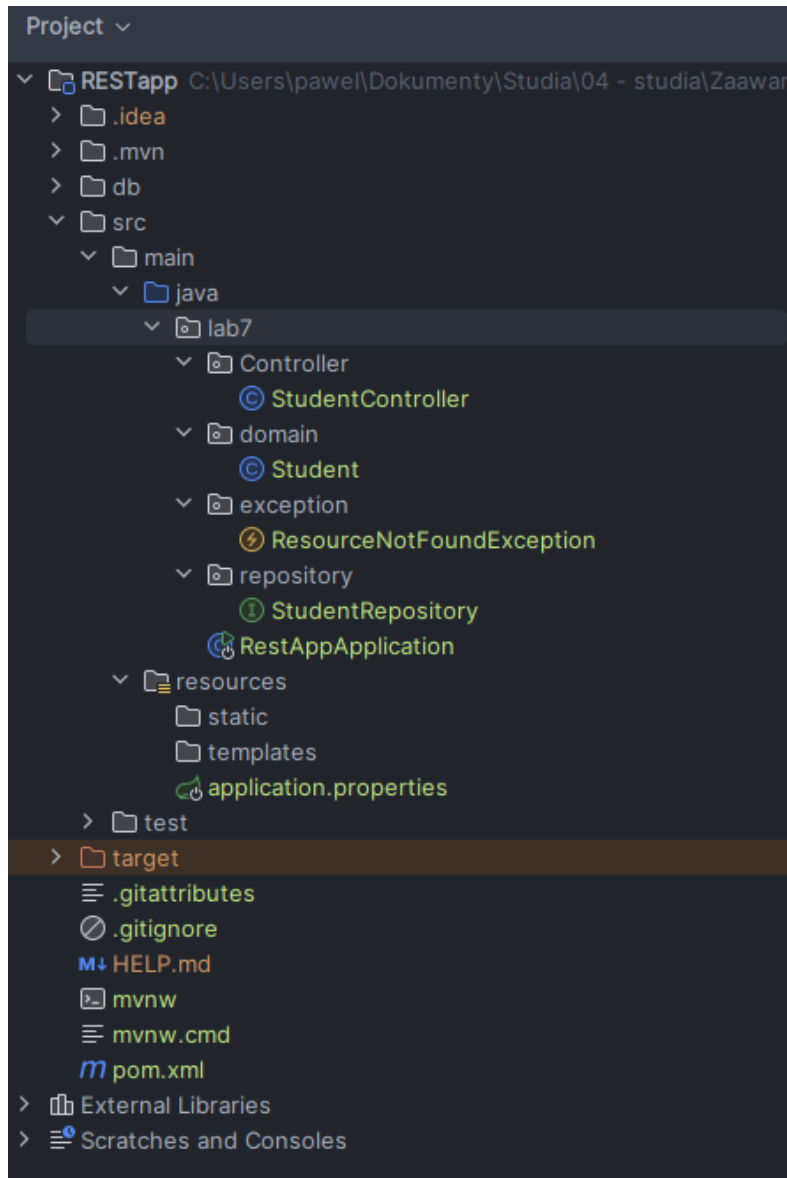
localhost:8080/students/2

Pretty print

```
{"id":2,"firstName":"Anna","lastName":"Nowak","studentId":"654321"}
```



## Wygląd struktury projektu



## Wnioski

Na podstawie wykonanego zadania, nauczyłem się, jak budować prostą aplikację RESTful w Spring Boot, która pozwala na pobieranie danych o studentach. Kluczowe było zrozumienie, jak działają encje JPA z adnotacjami Jakarta Persistence w Spring Boot oraz jak skonfigurować bazę danych H2 do przechowywania tych danych. Dzięki HAL Browserowi i H2 Console mogłem skutecznie testować API i sprawdzać stan bazy danych. Zadanie czasochłonne to i jak również lab6 dlatego małe opóźnienie w przesłaniu. Nie wykonywałem zadania dodatkowego.