Student Management using Spring Boot with H2

Submitted by: Tummala Hiranmai

Table of Contents

1. Introduction

This project is a hands-on implementation of a Student Management System using Spring Boot with an H2 in-memory database. It demonstrates full CRUD functionality through REST APIs and integrates Spring Data JPA for ORM.

2. Technologies Used

- Java 17
- Spring Boot
- Spring Data JPA
- H2 Database
- Maven
- Postman

3. application.properties Configuration

```
# H2 Database Configuration
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa spring.datasource.password=

# JPA + Hibernate Configuration spring.jpa.database-
platform=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=update spring.jpa.show-sql=true

# H2 Console Access spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
```

4. Student Entity Class

```
@Entity
@Table(name = "students") public
class Student {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

   @Column(nullable = false)
private String name;
```

```
    private String department;

    // Getters and Setters
}
```

5. Student Repository

```
public interface StudentRepository extends JpaRepository<Student, Long> {
    List<Student> findByDepartment(String department);
}
```

6. Student Service Layer

```
@Service public class
StudentService {

    @Autowired     private
StudentRepository repository;

    public Student save(Student student) { return repository.save(student); }

    public Optional<Student> get(Long id) { return repository.findById(id); }

    public void delete(Long id) { repository.deleteById(id); }

    public List<Student> getByDepartment(String dept) {
return repository.findByDepartment(dept);
    }
}
```

7. Student Controller

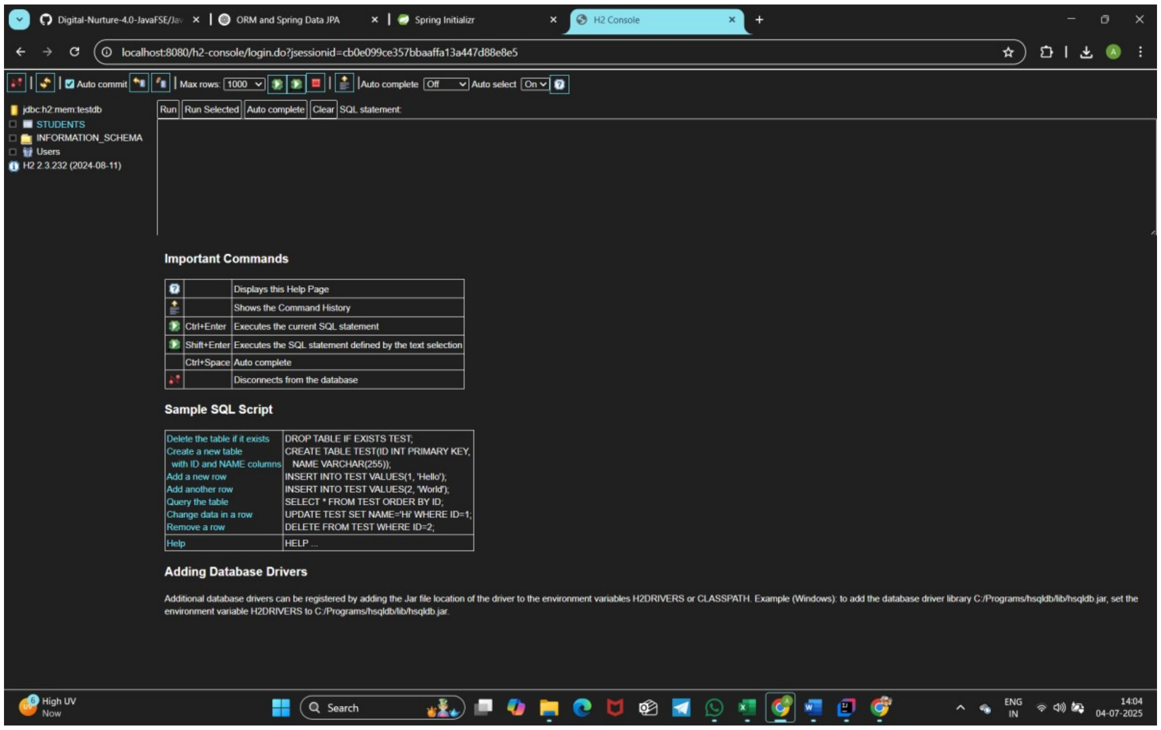
```
@RestController
@RequestMapping("/students") public
class StudentController {
    @Autowired
    private StudentService service;

    @PostMapping
    public Student create(@RequestBody Student student) { return service.save(student); }

    @GetMapping("/{id}")     public ResponseEntity<Student>
get(@PathVariable Long id) {         return
service.get(id).map(ResponseEntity::ok).orElse(ResponseEntity.notFound().build());
    }

    @DeleteMapping("/{id}")
    public void delete(@PathVariable Long id) { service.delete(id); }

    @GetMapping("/department/{dept}")
    public List<Student> byDepartment(@PathVariable String dept) {
return service.getByDepartment(dept);
    }
}
```

8. Testing Endpoints

Use Postman or Swagger to test the following endpoints:

- POST /students
- GET /students/{id}
- GET /students/department/{dept}
- DELETE /students/{id}

9. H2 Console Screenshot



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

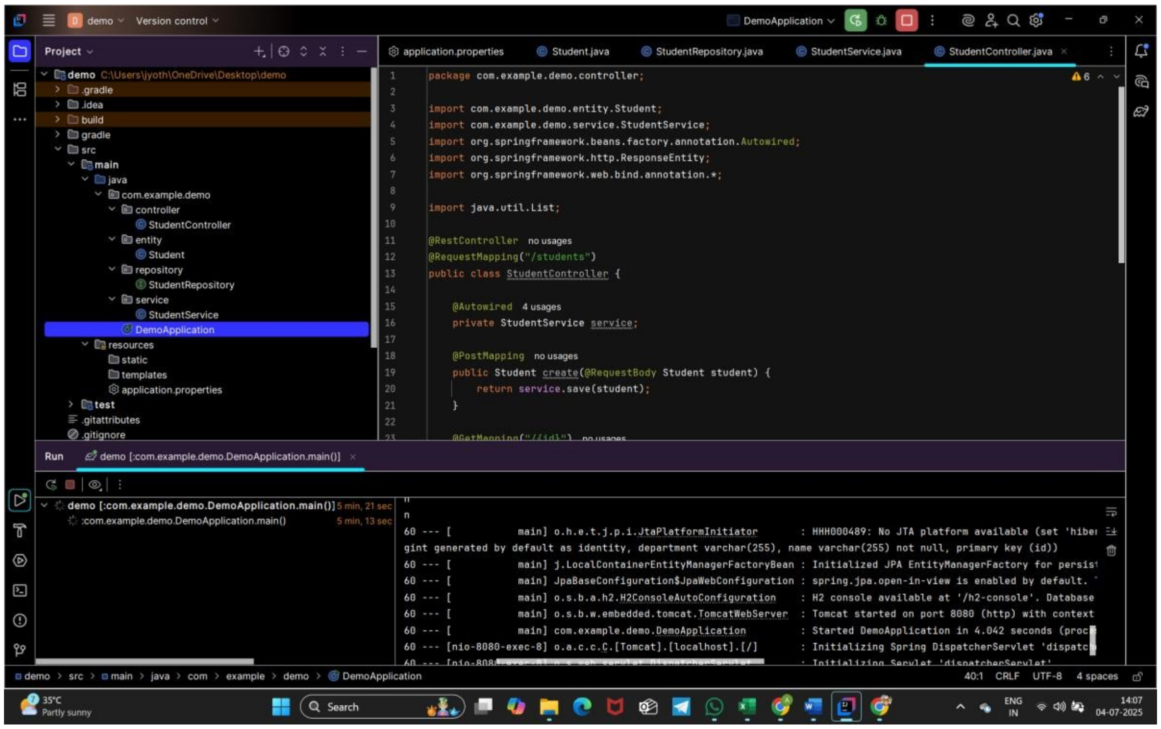
Sample SQL Script

Delete the table if it exists
Create a new table
with ID and NAME columns
Add a new row
Add another row
Query the table
Change data in a row
Remove a row
	DROP TABLE IF EXISTS TEST;
	CREATE TABLE TEST(ID INT PRIMARY KEY,
	NAME VARCHAR(255));
	INSERT INTO TEST VALUES(1, 'Hello');
	INSERT INTO TEST VALUES(2, 'World');
	SELECT * FROM TEST ORDER BY ID;
	UPDATE TEST SET NAME='Hi' WHERE ID=1;
	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, set the environment variable H2DRIVERS to C:/Programs/hsqldb/lib/hsqldb.jar.



```
package com.example.demo.controller;

import com.example.demo.entity.Student;
import com.example.demo.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController  no usages
@RequestMapping("/students")
public class StudentController {

    @Autowired  4 usages
    private StudentService service;

    @PostMapping  no usages
    public Student create(@RequestBody Student student) {
        return service.save(student);
    }

```

```
n
60 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hiber
gint generated by default as identity, department varchar(255), name varchar(255) not null, primary key (id))
60 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persis
60 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default.
60 --- [           main] o.s.b.a.h2.H2ConsoleAutoConfiguration    : H2 console available at '/h2-console'. Database
60 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context
60 --- [           main] com.example.demo.DemoApplication         : Started DemoApplication in 4.042 seconds (proc
60 --- [nio-8080-exec-8] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatch
```

10. Key Concepts Explained

- ORM (Object Relational Mapping): Maps Java classes to database tables automatically.
- JPA (Java Persistence API): A specification that simplifies data persistence.
- Hibernate: A JPA implementation used under Spring Boot for handling database interactions.
- Spring Data JPA: Simplifies the implementation of JPA-based repositories by eliminating boilerplate code.