

LAB 1

DEVELOP JAVA SPRING BOOT WEB APP (P1)

❖ CONTENT

- Create Java Spring Boot project in IntelliJ with autoconfiguration
- Create table with Hibernate
- Implement CRUD features with JPA
- Create view for web with Thymeleaf

❖ INTRODUCTION

- Spring framework: a Java platform that provides comprehensive infrastructure support for developing Java application
- Spring Boot: a tool that makes developing web application and microservices with Spring framework faster and easier with autoconfiguration
- Hibernate: an object-relational mapping (ORM) tool for Java programming language that simplifies the interaction with the database
- JPA (Java Persistence API): a collection of classes and methods to persistently store that vast amounts of data into a database
- Thymeleaf: a modern server-side Java template engine for both web and standalone environments

❖ INSTRUCTION

1. Create new Java Spring Boot project in IntelliJ using Spring Initializr

- **New Project**
- **Select Spring Initializr**
- **Input project parameters:**
 - Project name
 - Project location
 - Language: **Java**
 - Type: **Maven**
 - JDK: **19**
 - Java: **19**
 - Packaging: **Jar**

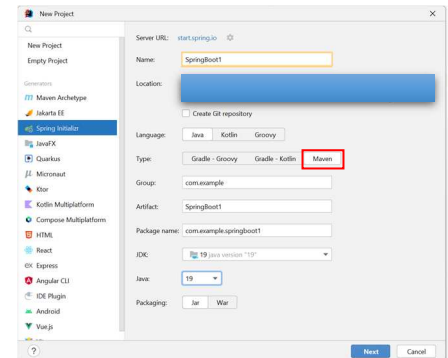


Figure 1 - Create new Spring Boot project (1)

- **Click Next**
- **Spring Boot version: 3.0.2**
- **Select dependencies:**
 - Spring Web
 - Thymeleaf
 - Spring Data JPA
 - MySQL Driver
- **Click Finish**

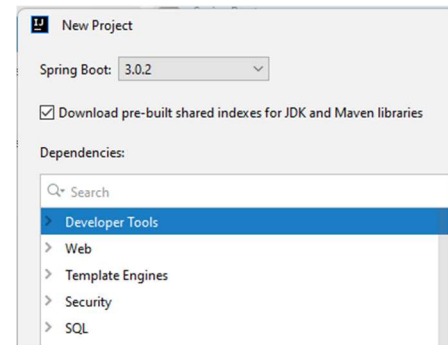


Figure 2 - Create new Spring Boot project (2)



Figure 3 - Create new Spring Boot project (3)

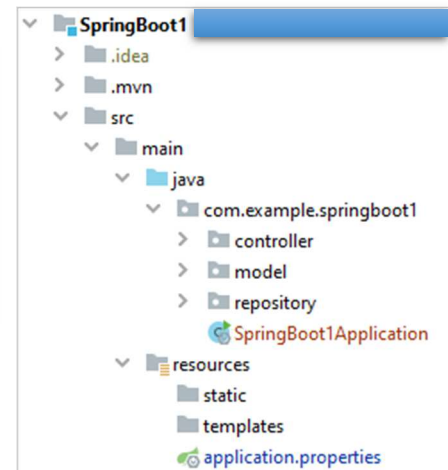


Figure 4 - Sample project structure

2. Config parameters for MySQL connection, JPA & Hibernate (located at *src/main/resources* folder)

```
# MYSQL
spring.datasource.url=jdbc:mysql://localhost:3306/springbootdb?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=root

# JPA / HIBERNATE
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=update

# THYMELEAF
spring.thymeleaf.cache = false
```

Figure 5 - *application.properties*

3. Create Java class for model (entity) which acts as table in database (located at sub-package **model** in *src/main/java* folder)

```

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

    //auto-generated getters & setters

```

Figure 6 - *Employee.java*

4. Create Java interface which extends *JpaRepository* for CRUD features (*located at sub-package **repository** in **src/main/java** folder*)

```

public interface EmployeeRepository extends JpaRepository<Employee, Long> {
}

```

Figure 7 - *EmployeeRepository.java*

5. Create Java class for controller which gets data from database and renders view (*located at sub-package **controller** in **src/main/java** folder*)

```

@Controller
public class EmployeeController {
    @Autowired
    EmployeeRepository employeeRepository;

    @RequestMapping(value = "{/}")
    public String getAllEmployee(Model model) {
        List<Employee> employees = employeeRepository.findAll();
        model.addAttribute( attributeName: "employees", employees);
        return "employeeList";
    }

    @RequestMapping(value = "{/}/{id}")
    public String getEmployeeById(
        @PathVariable(value = "id") Long id, Model model) {
        Employee employee = employeeRepository.findById(id);
        model.addAttribute( attributeName: "employee", employee);
        return "employeeDetail";
    }
}

```

Figure 8 - *EmployeeController.java*

6. Create HTML pages as view (located at *src/main/resources/templates* folder)

```
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Employee List</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
        rel="stylesheet" integrity="sha384-18mE4kWBq781YhF1dvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
        crossorigin="anonymous">
</head>
<body>
  <div class="container col-md-4 text-center mt-4">
    <h2 class="text text-primary">EMPLOYEE LIST</h2>
    <table class="table table-info mt-3">
      <thead>
        <tr>
          <th>ID</th>
          <th>Name</th>
          <th>Image</th>
        </tr>
      </thead>
      <tbody>
        <tr th:each="employee : ${employees}">
          <td th:text="${employee.id}" />
          <td <a th:text="${employee.name}" /> </td>
          <td>
            <a th:href="/" + ${employee.id}" > 
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</body>
</html>
```

Figure 9 - *employeeList.html*

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Employee Detail</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
        rel="stylesheet" integrity="sha384-6Lh1TQ8iRABdZL1603oVMWMSktQOp6b7In1ZL3/Jr59b6EG6oI1aFkw7cmDA6j6gD"
        crossorigin="anonymous">
</head>
<body>
  <div class="container col-md-5 text-center mt-4">
    <h2 class="text text-primary mb-4">EMPLOYEE DETAIL</h2>
    <div class="row bg-light">
      <div class="col">
        
      </div>
      <div class="col">
        <h1 class="text-success" th:text="${employee.name}" />
        <h3 th:text="'Age: ' + ${employee.age}" />
        <h3 th:text="'Address: ' + ${employee.address}" />
      </div>
    </div>
  </div>
</body>
```

Figure 10 - *employeeDetail.html*

7. Add sample data (records) to that table in database with MySQL Workbench or integrated MySQL database in IntelliJ

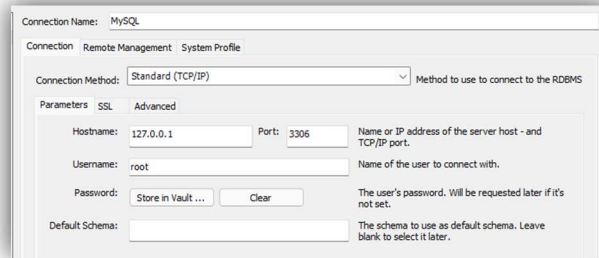


Figure 11 – Setup connection to MySQL with MySQL Workbench

```
/* Create new database */
CREATE DATABASE bookdb;
/* Use this database */
USE bookdb;
/* Create new table */
CREATE TABLE book (
  id INT PRIMARY KEY AUTO_INCREMENT,
  title VARCHAR(50) NOT NULL,
  author VARCHAR(30) NOT NULL,
  price FLOAT NOT NULL);
/* Insert data to this table */
INSERT INTO book (title, author, price)
VALUES ("Java Web", "John", 100), ("Spring Boot", "David", 120);
```

Figure 12 – Create database, table and populate data with SQL script

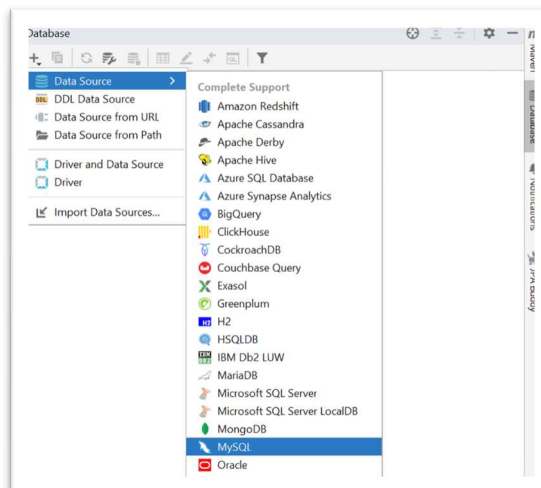


Figure 13 – Setup connection to integrated MySQL database in IntelliJ (1)

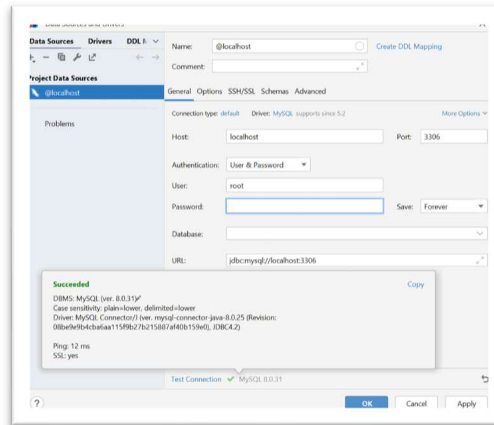


Figure 14 – Setup connection to integrated MySQL database in IntelliJ (2)

WHERE			ORDER BY address		
	id	address	age	image	name
1	1	Hà Nội	30	https://encr...	Nguyễn Tiến Hùng
2	3	Nam Định	40	https://encr...	Hoàng Quốc Tuấn
3	2	Nghệ An	25	https://imag...	Trần Thị Quỳnh Phương

Figure 15 - Add sample data to table

8. Run the web application (CTRL + SHIFT + F10)



Figure 16 – SpringBoot1Application.java

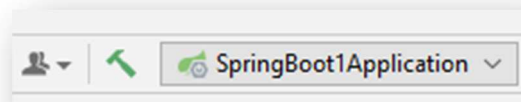


Figure 17 - Run web application

9. Type this address <http://localhost:8080/> in a web browser such as Google Chrome to access the web app


EMPLOYEE LIST		
ID	Name	Image
1	Nguyễn Tiến Hùng	
2	Trần Thị Quỳnh Phương	
3	Hoàng Quốc Tuấn	

Figure 18 - Employee List page


EMPLOYEE DETAIL	
	Nguyễn Tiến Hùng Age: 30 Address: Hà Nội

Figure 19 - Employee Detail page

❖ TO-DO

- Complete remained CRUD features including CREATE, UPDATE and DELETE
- *Note:* Create new methods in Controller then create corresponding HTML files (such as *employeeAdd.html*)