LAB 2

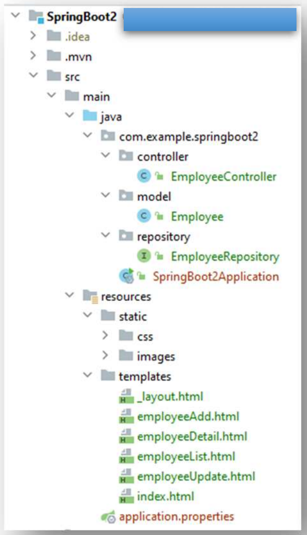
DEVELOP JAVA SPRING BOOT WEB APP (P2)

❖ CONTENT

- Setup system authentication (login/logout) using *Spring Security*
- Make data validation using *Hibernate Validator* and display form input error using *Thymeleaf*
- Establish web template using *Thymeleaf layout dialect*

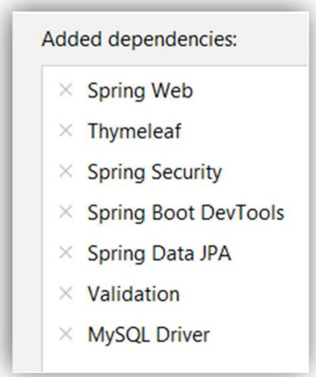
❖ INSTRUCTION

1. Create new Java Spring Boot project with dependencies



```
SpringBoot2
  .idea
  .mvn
  src
    main
      java
        com.example.springboot2
          controller
            EmployeeController
          model
            Employee
          repository
            EmployeeRepository
          SpringBoot2Application
      resources
        static
          css
          images
        templates
          _layout.html
          employeeAdd.html
          employeeDetail.html
          employeeList.html
          employeeUpdate.html
          index.html
        application.properties
```

Figure 1 – Sample project structure



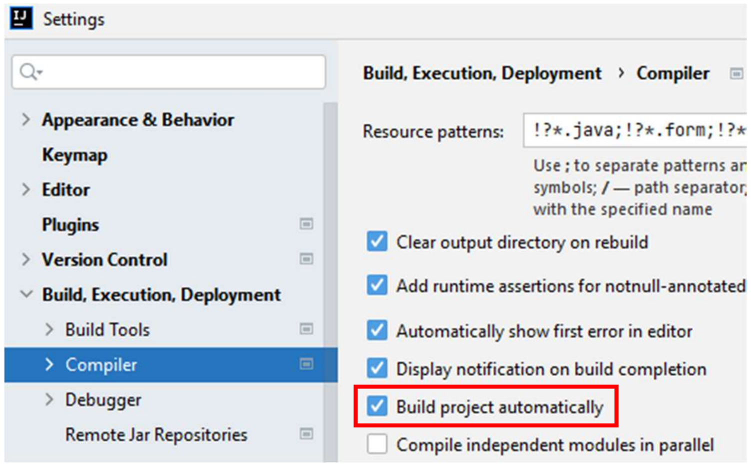
Added dependencies:

- Spring Web
- Thymeleaf
- Spring Security
- Spring Boot DevTools
- Spring Data JPA
- Validation
- MySQL Driver

Figure 2 - Project dependencies

2. Setup automatic reload static web page (HTML + CSS files)

- **File** ⇨ **Settings** (*Ctrl + Alt + S*)

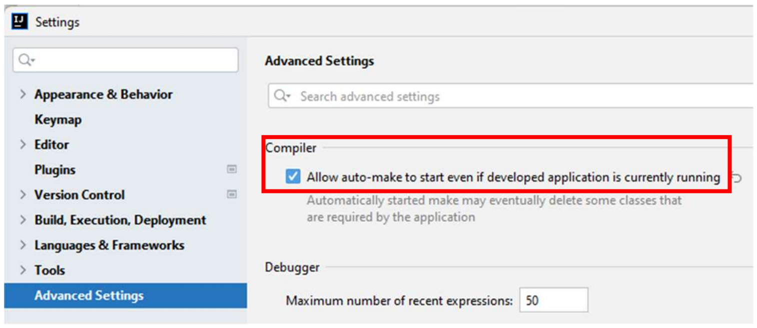


Resource patterns: !?*.java;!?*.form;!?*

Use ; to separate patterns ar
symbols; / — path separator;
with the specified name

- [✓] Clear output directory on rebuild
- [✓] Add runtime assertions for notnull-annotated
- [✓] Automatically show first error in editor
- [✓] Display notification on build completion
- [✓] Build project automatically
- [ ] Compile independent modules in parallel

Figure 3 – Setup automatic reload web page (1)



Compiler

- [✓] Allow auto-make to start even if developed application is currently running

Automatically started make may eventually delete some classes that are required by the application

Debugger

Maximum number of recent expressions: 50

Figure 4 - Setup automatic reload web page (2)

3. Config MySQL connection, JPA & Hibernate, Thymeleaf

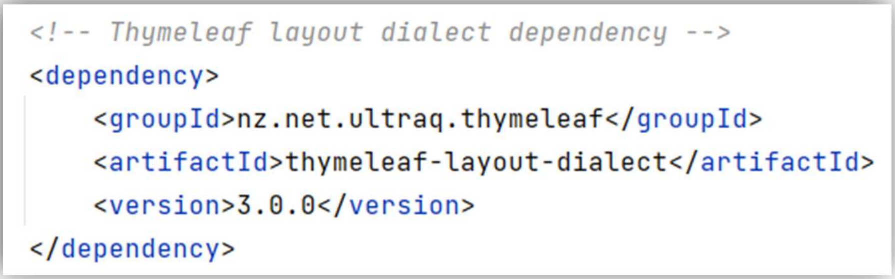
- Config default login information (authentication) & server port *(optional)*

```
# SPRING SECURITY
spring.security.user.name=admin
spring.security.user.password=123456

# SERVER PORT (Optional)
server.port=8081
```

*Figure 5 - **application.properties***

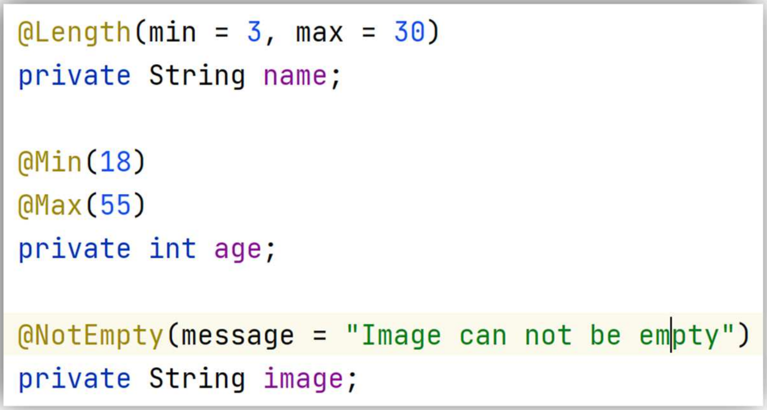
4. Add Thymeleaf layout dependency manually in file *pom.xml*



```xml
<!-- Thymeleaf layout dialect dependency -->
<dependency>
    <groupId>nz.net.ultraq.thymeleaf</groupId>
    <artifactId>thymeleaf-layout-dialect</artifactId>
    <version>3.0.0</version>
</dependency>
```

Figure 6 - ***pom.xml***

5. Create Java class for model (entity) which acts as table in database
   - Update code for data validation



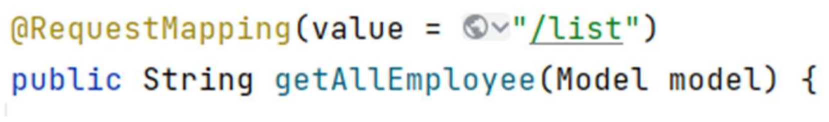
```java
@Length(min = 3, max = 30)
private String name;

@Min(18)
@Max(55)
private int age;

@NotEmpty(message = "Image can not be empty")
private String image;
```

Figure 7 - ***Employee.java***

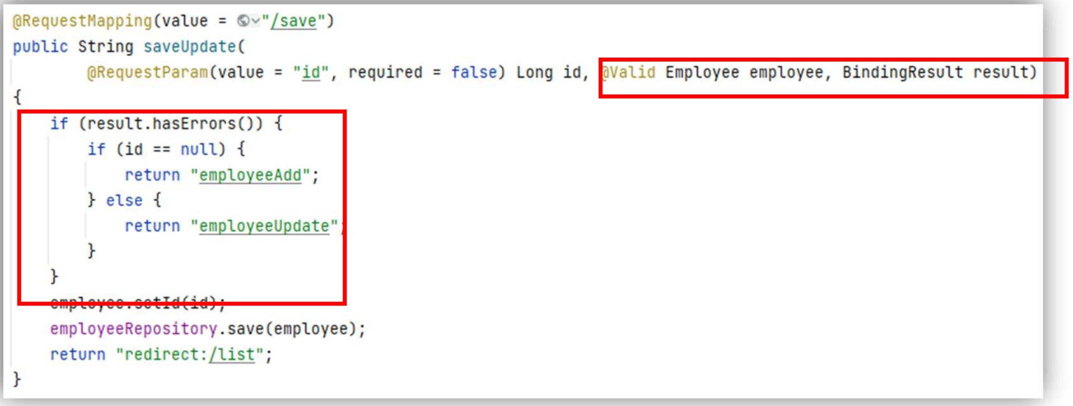
6. Create Java interface which extends *JpaRepository*
7. Create Java class for controller which gets data from database and renders view
   - Update value for *@RequestMapping* annotation



```java
@RequestMapping(value = "/list")
public String getAllEmployee(Model model) {
```

Figure 8 - @RequestMapping

- Update code for *saveUpdate()* method to show the form input error

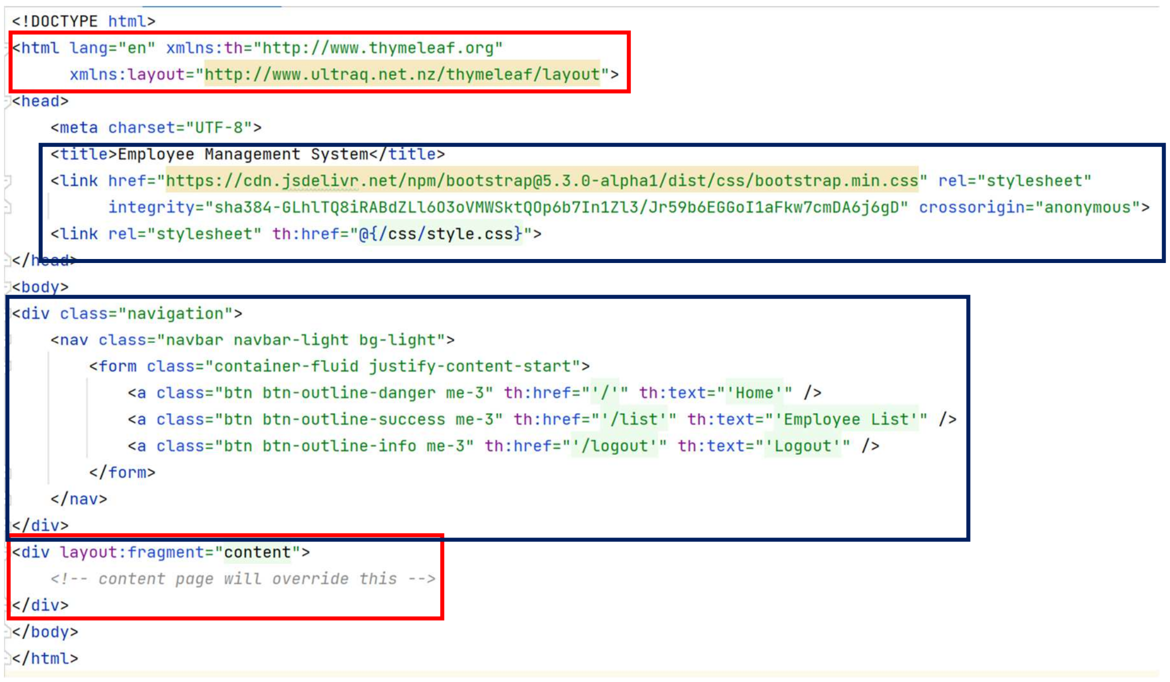


```java
@RequestMapping(value = "/save")
public String saveUpdate(
        @RequestParam(value = "id", required = false) Long id, @Valid Employee employee, BindingResult result)
{
    if (result.hasErrors()) {
        if (id == null) {
            return "employeeAdd";
        } else {
            return "employeeUpdate";
        }
    }
    employee.setId(id);
    employeeRepository.save(employee);
    return "redirect:/list";
}
```

Figure 9 - saveUpdate() method

8. Create HTML pages with Thymeleaf as view *(Refers to Tutorial 2)*

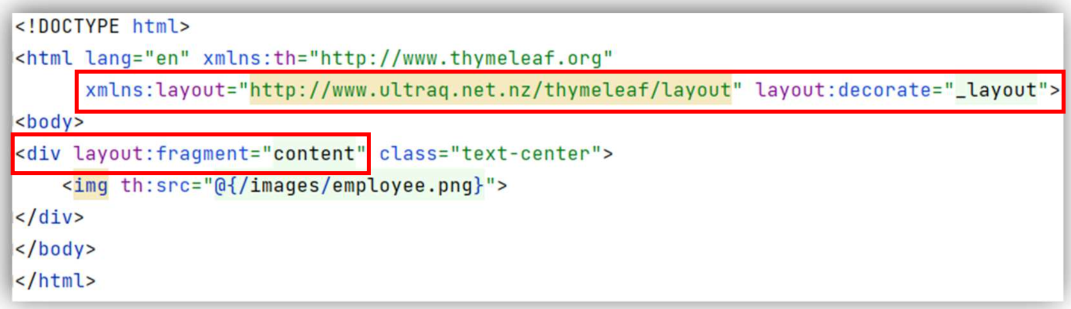
- Add a web template *(_layout.html)*



```html
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
    <meta charset="UTF-8">
    <title>Employee Management System</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet"
          integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">
    <link rel="stylesheet" th:href="@{/css/style.css}">
</head>
<body>
<div class="navigation">
    <nav class="navbar navbar-light bg-light">
        <form class="container-fluid justify-content-start">
            <a class="btn btn-outline-danger me-3" th:href="'/'" th:text="'Home'" />
            <a class="btn btn-outline-success me-3" th:href="'/list'" th:text="'Employee List'" />
            <a class="btn btn-outline-info me-3" th:href="'/logout'" th:text="'Logout'" />
        </form>
    </nav>
</div>
<div layout:fragment="content">
    <!-- content page will override this -->
</div>
</body>
</html>
```

*Figure 10 - **_layout.html***

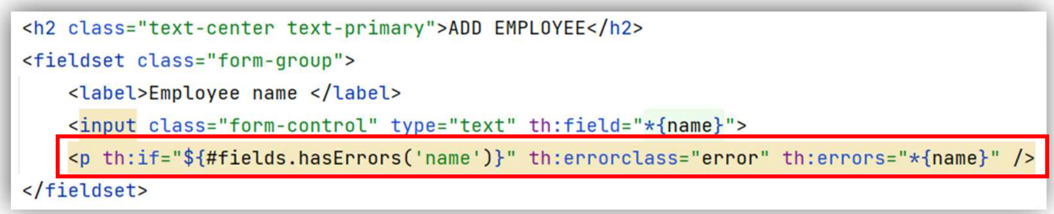
- Add homepage *(index.html)*



```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout" layout:decorate="_layout">
<body>
<div layout:fragment="content" class="text-center">
    <img th:src="@{/images/employee.png}">
</div>
</body>
</html>
```

Figure 11 - index.html

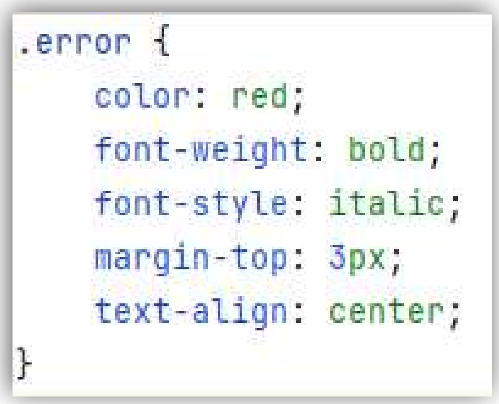
- Add Thymeleaf code to display error on form input



```
<h2 class="text-center text-primary">ADD EMPLOYEE</h2>
<fieldset class="form-group">
    <label>Employee name </label>
    <input class="form-control" type="text" th:field="*{name}">
    <p th:if="${#fields.hasErrors('name')}" th:errorclass="error" th:errors="*{name}" />
</fieldset>
```

Figure 12 - employeeAdd.html

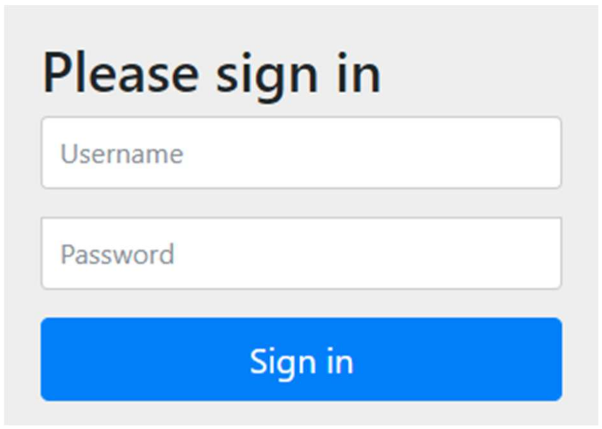
- Create CSS file *(style.css)* to format the input error



```
.error {
    color: red;
    font-weight: bold;
    font-style: italic;
    margin-top: 3px;
    text-align: center;
}
```

Figure 13 - style.css

9. Run the web application with a web browser



Please sign in

Figure 14 – Login page

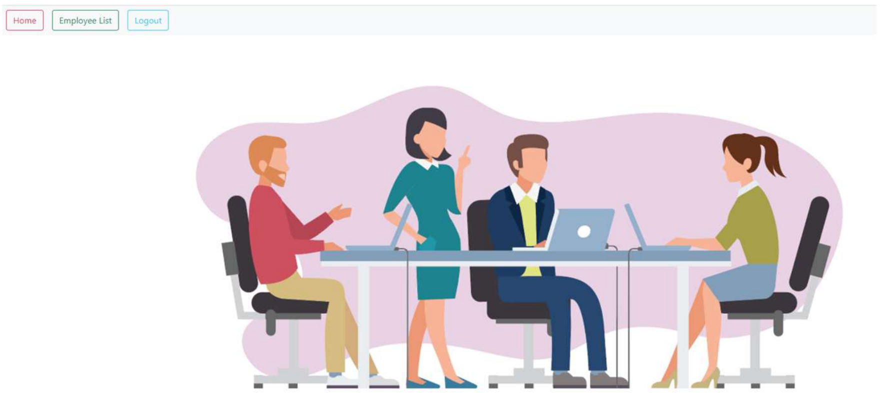
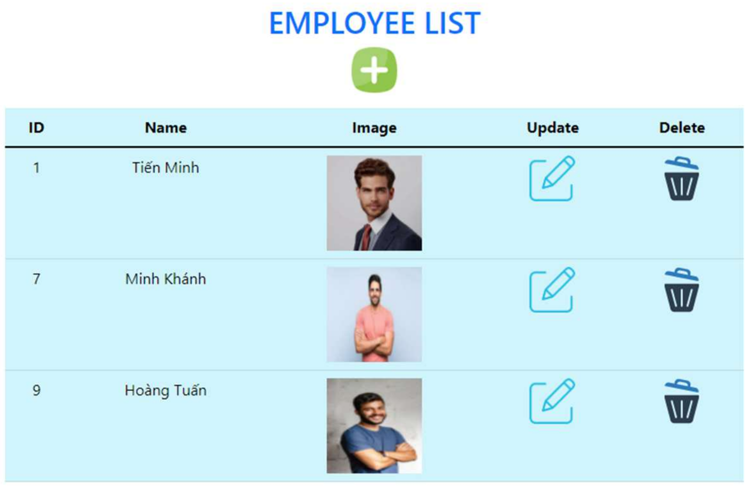


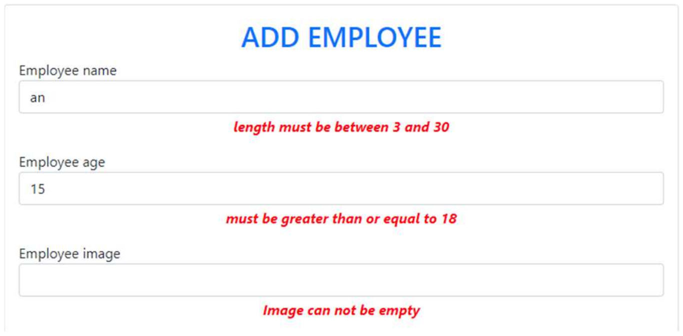
Figure 15 – Homepage with navigation



EMPLOYEE LIST

ID	Name	Image	Update	Delete
1	Tiến Minh			
7	Minh Khánh			
9	Hoàng Tuấn			

Figure 16 - Employee list with Add, Update & Delete features


ADD EMPLOYEE

Employee name

an

length must be between 3 and 30

Employee age

15

must be greater than or equal to 18

Employee image

Image can not be empty

Figure 17 – Add employee with input validation

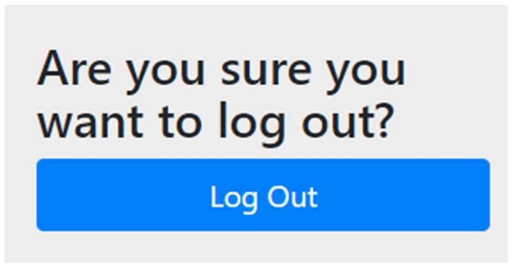

Are you sure you want to log out?

Figure 18 – Logout page

❖ TO-DO

- Complete the remained codes to run web application
- Add more entity attributes with corresponding validation then update codes in Add & Edit forms to show errors
- *Extra:* Create user registration page, change login form interface, create other accounts with different roles then implement authorization feature (role-based access), upload image by file instead of using web url