LAB 3
DEVELOP JAVA SPRING BOOT WEB APP (P3)

❖ **CONTENT**

- Setup relationship between entities (tables)

- Implement extra features: Filter, Search, Sort

❖ **INSTRUCTION**

1. Import the previous Spring Boot project to continue coding

   File ⇨ **Open** ⇨ Select the project folder

2. Add new entity (*Company*) then add a proper attribute as relationship between entities (*Employee – Company : ManyToOne*)

```java
@Entity
public class Company {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;
    @Size(min = 3, max = 30)
    private String name;
    @NotEmpty
    private String image;
    @Length(min = 5, max = 50)
    private String address;
```

*Figure 1 - **Company.java***

```java
@ManyToOne
private Company company;
```

*Figure 2 - **Employee.java***

3. Add the *Repository* for *Company*

4. Add the *Controller* for *Company*

*Figure 3 - **CompanyController.java** (1)*

*Figure 4 - **CompanyController.java** (2)*

5. Update *Controller* for *Employee* (update links & add *CompanyRepository*)

*Figure 5 - **EmployeeController.java** (1)*

```java
@RequestMapping(value = ©∨"/add")
public String addEmployee (Model model) {
    Employee employee = new Employee();
    List<Company> companies = companyRepository.findAll();
    model.addAttribute( attributeName: "companies", companies);
    model.addAttribute( attributeName: "employee", employee);
    return "employeeAdd";
}
```

*Figure 6 - **EmployeeController.java** (2)*

6. Create *View* for *Company* which extends web layout

```html
<h4 th:each="employee: ${employees}">
    <a class="text-decoration-none" th:href="'/employee/' + ${employee.id}"
        th:if="${employee.company?.getId() == company.id}" th:text="${employee.name}" />
</h4>
```

*Figure 7 - **companyDetail.html***

7. Update *View* for *Employee*

```html
<fieldset class="form-group">
    <label>Company name </label>
    <select class="form-select" th:field="*{company}">
        <option th:each="comp : ${companies}" th:value="${comp.id}" th:text="${comp.name}" />
    </select>
</fieldset>
```

*Figure 8 - **employeeAdd.html***

```html
<h3 th:if="${employee.company != null}"
    th:text="'Company: ' + ${employee.company.name}" />
```

*Figure 9 - **employeeDetail.html***

8. Update navigation path in web layout

```html
<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="'/employee/list'" th:text="'Employee'" />
    <a class="btn btn-outline-success me-3" th:href="'/company/list'" th:text="'Company'" />
    <a class="btn btn-outline-info me-3" th:href="'/logout'" th:text="'Logout'" />
</form>
```

Figure 10 - _layout.html

9. Implement *Search* feature for *Employee*

```java
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
    List<Employee> findByNameContaining(String name);
}
```

Figure 11 – EmployeeRepository.java

```java
@RequestMapping(©~"/search")
public String searchEmployee(
        Model model,
        @RequestParam(value = "name") String name) {
    List<Employee> employees = employeeRepository.findByNameContaining(name);
    model.addAttribute( attributeName: "employees", employees);
    return "employeeList";
}
```

Figure 12 – EmployeeController.java (3)

```html
<div class="mt-3 col-3">
    <form action="/employee/search">
        <input type="search" class="form-control" placeholder="Search by name" name="name" />
    </form>
</div>
```

Figure 13 - employeeList.html

10. Implement *Sort* feature for *Employee*

```java
@RequestMapping(©▾"/sort/asc")
public String sortEmployeeAsc(Model model) {
    List<Employee> employees = employeeRepository.findAll(Sort.by(Sort.Direction.ASC, ...properties: "name"));
    model.addAttribute( attributeName: "employees", employees);
    return "employeeList";
}

@RequestMapping(©▾"/sort/desc")
public String sortEmployeeDesc(Model model) {
    List<Employee> employees = employeeRepository.findAll(Sort.by(Sort.Direction.DESC, ...properties: "name"));
    model.addAttribute( attributeName: "employees", employees);
    return "employeeList";
}
```

*Figure 14 - **EmployeeController.java** (4)*

```html
<th>Name
    <a th:href="'/employee/sort/asc'" class="text-decoration-none">
        <img th:src="@{/images/up.png}">
    </a>
    <a th:href="'/employee/sort/desc'" class="text-decoration-none">
        <img th:src="@{/images/down.png}">
    </a>
</th>
```

*Figure 15 - **employeeList.java** (2)*

11. Test the web application
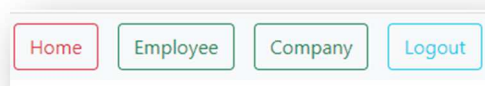
Figure 16 - Navigation bar (updated)

Figure 17 - Company List

COMPANY DETAIL

FPT
Address: Phạm Văn Bạch
Employee List:
Mạnh Linh
Quốc Huy

Figure 18 - Company Detail

ADD EMPLOYEE

Employee name

Employee age

0

Employee image

Employee address

Company name

Viettel ⌄

| FPT |
| Viettel |
| VNG |

Figure 19 - Add Employee (updated)

EMPLOYEE LIST

➕

Search by name

ID	Name ⬆⬇	Image	Company	Update	Delete
13	Mạnh Linh		**FPT**	✎	🗑
14	Tố Uyên		**Viettel**	✎	🗑
15	Quốc Huy		**FPT**	✎	🗑
16	Hà My		**VNG**	✎	🗑

Figure 20 - Employee List (updated)

❖ TO-DO

- Complete the remained codes to run web application

- Implement search & sort features for *Company*

- Add new entity *Job* (*Employee – Job : ManyToMany*) then do similar with entity *Company*

- *Extra:* Implement the pagination feature (such as display 5 records/page)