Tutorial 5 – Web Application With Spring Boot (1d)

❖ Contents:

- Build data filter for web app with Spring Boot, Thymeleaf & JavaScript

❖ Instructions:

1. Import the previous project of **Tutorial 4** to continue coding.

2. In your `getAllEmployee()` method, add a list of all companies into the `model` object:

```
List<Company> companies = companyRepository.findAll();
model.addAttribute("companies", companies);
```

3. Add these drop-downs (`<select>` tags) into your `employeeList.html` template:

```
<div class="filterContainer">
    Company:
    <select id="filterCompany">
        <option value="0">All companies</option>
        <option th:each="comp : ${companies}"
                th:value="${comp.id}"
                th:text="${comp.name}" />
    </select>
    Sort:
    <select id="sortOptions">
        <option value="0">Latest</option>
        <option value="1">Oldest</option>
        <option value="2">By name ASC</option>
        <option value="3">By name DESC</option>
    </select>
</div>
```

4. Add two request parameters to the `getAllEmployee()` method.

```
@RequestMapping(value = "/list") // actual URL: /employee/list
public String getAllEmployee(
        @RequestParam(value = "company", required = false, defaultValue = "0") Long comId,
        @RequestParam(value = "sort", required = false, defaultValue = "0") int sortMode,
        Model model) {
```

You'll be able to receive querystring parameters in this controller method. For example:

`http://localhost:8080/employee/list?company=1&sort=2`

`comId` will get the value `1`, sortMode will receive value `2`.

5. Add these values to the model object so that we can use them later in the Thymeleaf template:

```
model.addAttribute("comId", comId);
model.addAttribute("sortMode", sortMode);
```

6. Use JavaScript to redirect to the correct web URL when a filter changes its value. Also use JavaScript to make sure the selected value on the drop-down menus reflect the current filter choices:

```
<script>
    let comId = [[${ comId }]];
    let sortMode = [[${ sortMode }]];
    function filterRedirect() {
        let url = "/employee/list?company=" + comId + "&sort=" + sortMode;
        window.location.href = url; // redirect
    }
    window.addEventListener("load", function () {
        const comFilter = document.getElementById("filterCompany");
        comFilter.value = comId;
        comFilter.addEventListener("change", function (e) {
            comId = e.target.value;
            filterRedirect();
        });
        const sortMenu = document.getElementById("sortOptions");
        sortMenu.value = sortMode;
        sortMenu.addEventListener("change", function (e) {
            sortMode = e.target.value;
            filterRedirect();
        });
    });
</script>
```

(*) Teacher should explain the above JavaScript code for you in case you're not proficient in JS programming.

7. Add a query method in `EmployeeRepository` to get employees by company and also apply sorting to the result:

```
List<Employee> findByCompany(Company company, Sort sort);
```

8. In your controller method (`getAllEmployee()`), decide on the sorting direction and sorting column based on the received query parameter named `sort` which is stored in the `sortMode` parameter:

```
if (sortMode == 1 || sortMode == 2) {
    sortOrder = Sort.Direction.ASC;
}
if (sortMode == 2 || sortMode == 3) {
    sortColumn = "name";
}
```

9. Try to get the list of employees from a specific company (don't forget to check if that company exists first):

```
List<Employee> employees = null;
if (comId != 0) {
    Optional<Company> comp = companyRepository.findById(comId);
    if (comp.isPresent()) {
        employees = employeeRepository.findByCompany(
                comp.get(),
                Sort.by(sortOrder, sortColumn)
        );
    }
}
```

Please note that, if the provided company is not found, this will results in the `employees` object still being `null`.

10. If the list of employees from a company was not retrieved, we will get a list of employees from all companies:

```
if (employees == null) {
    // failed to filter by Company
    employees = employeeRepository.findAll(
            Sort.by(sortOrder, sortColumn)
    );
}
```

❖ Run, debug & submit your project

Once finished, compress your project into a `.zip` file and submit the file to the tutorial's submission box.