**Overview of React-Spring-Pagination**

**Introduction**

The purpose of this project is to learn some details about a simple application based on React, Spring Boot, REST API, and MariaDB database. React is used to create the UI frontend, while Spring implements the backend. This is not a tutorial. Is is assumed you already know how to install the software.

**Configuration**

See the pom.xml and package.json files for the versions of each component in the application. Netbeans or another IDE may be used to build the jar file. For Windows, a batch file, myReactBoot.bat, may be used to launch the application on port 8081. Open a browser and enter the URL, localhost:8081/front/home to open the home page.

**Frontend Details**

The application consists of just 3 screens: Home, Add Employee, and Employee List.

Graphical user interface, text

Description automatically generated

The Home screen contains a horizontal navigation bar which is defined in Navbar.js, using an HTML list. To make the list horizontal, the CSS “nav > ul > li” in App.css includes “display: inline;”.

Also, note the “Link to” references in Navbar.js, which are defined within the Routes tag in App.js. Each Route tag associates a link path to a React component. In this case the components are Home, AddEmployee, EmployeeList, and NotFound.

The Add Employee form contains input fields for Name, Department, and Location. Examine AddEmployee.js to see how the form is designed. Note how the form is centered horizontally with CSS “center App w400”. Text labels are aligned left with CSS “txleft”. Note the bootstrap CSS “col-lg-6” on the label tag and input field to create equal sized layout space.

Graphical user interface, text, application

Description automatically generated

The buttons are center aligned with “d-flex justify-content-center”. Reset is used to clear all form fields. Submit will send the form fields to the database via employeeService, which uses axios to convert the employee object into json for the Spring Boot REST service, explained later. If the user omits a required input, the submit halts, and an error message appears below the missing field. Look at the arguments to the register function on each form input field. If submit is successful, navigate returns to the Home screen.

The Employee List screen displays a table of employees, with additional columns to update or delete the data. Note this screen is created by two components: EmployeeList which prepares the column configuration, and PageTable which produces the actual HTML table. In EmployeeList note the function “useMemo” to prevent unnecessary calls on renders of the screen. Also, note the “cellProps” argument passed to the update/delete buttons. This represents the employee data in each row.

Graphical user interface, application

Description automatically generated

When the user presses the Update button, handleEdit is called with the employee argument, including the ID. The employee object is converted to json, and saved in sessionStorage for use in AddEmployee. The ID distinguishes a new employee from an updated employee. Therefore, the same form can be used to process both cases. AddEmployee checks sessionStorage for the update, and calls setValue to populate the form fields. Submit will call saveEmployee which invokes employeeService.update to send the updated employee to the backend.

**Backend Details**

The REST service is implemented with Spring Boot, which connects with PagingAndSortingRepository to perform CRUD operations on a database. EmployeeController maps REST paths to database calls. Axios and Spring Boot hide the conversion of EmployeeEntity to or from json.

Add employee sequence:

AddEmployee.save -> employeeService.create -> httpClient.post -> axios -> EmployeeController. saveEmployeeDetails -> EmployeeRepository.save -> database.

In the httpClient functions, note the url paths are enclosed in backticks, not quotes, to allow substitution variables.

Employee List sequence:

EmployeeList. init -> fetchPage -> employeeService.getAll -> httpClient.get -> axios -> EmployeeController. getAllEmployees -> EmployeeRepository. findAll -> database -> PageData<EmployeeEntity> -> EmployeeList.fetchPage response -> setData, setPageCount -> PageTable -> cell.getCellProps.

**Conclusion**

We have discussed a horizontal navigation bar, React form, React pagination table, axios http service, Spring Boot REST API, and PagingAndSortingRepository. This small application has demonstrated many implementation features. Hopefully, you can use some of these details in building your own React, Spring Boot, REST applications.