SPAs with React Router

Overview

In this lab you'll add real "library" functionality back in to your React application. The application will feature:

- React routing
- A menu
- A component to display books, either in tabular or list format
- A component to display films, either in tabular or list format
- Various other components 👍

Source folders

- ReactDev\Student\09-SinglePageApps
- ReactDev\Solutions\09-SinglePageApps

Roadmap

Here's a brief summary of the exercises in this lab. More detailed instructions follow later in this lab document:

1. Familiarization with the 'solution' web app
2. Getting started with the 'student' web app
3. Preparing to use React Router
4. Implementing the Menu component
5. Defining a route table
6. Implementing the PageNotFound component
7. Implementing the MoreStuff component

Exercise 1: Familiarization with the 'solution' web app

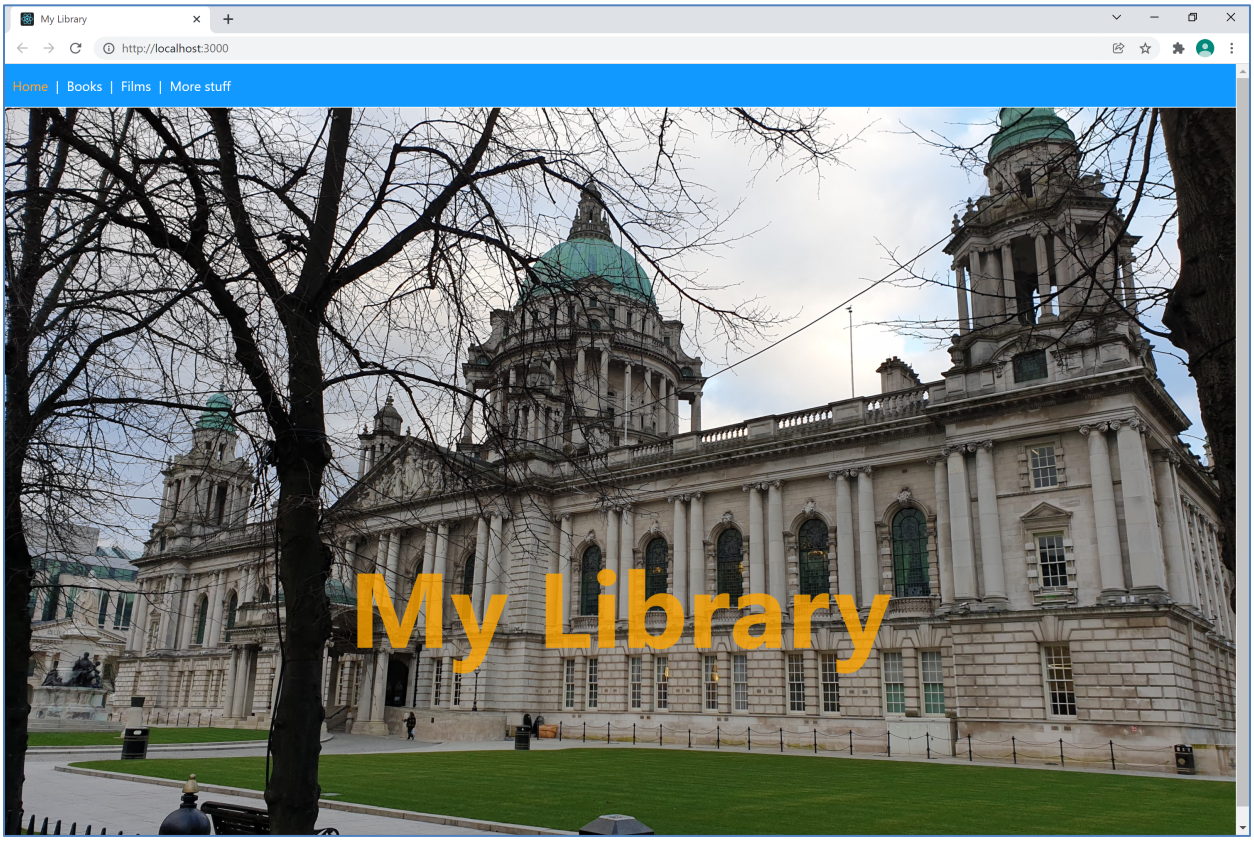
Open a Terminal window and go to the following folder:

```
ReactDev\Solutions\09-SinglePageApps\library-app
```

In this folder, run the following commands to install NPM libraries and to run the application:

```
npm install
npm start
```

The web application appears as follows initially in the browser:



Note the following points:

- There's a nice menu. This is implemented by a Menu component, as you'll see shortly.
- The home page displays a splash screen. This is implemented by a Home component.
- Click *Books* to see books. This is implemented by a Books component.
- Click *Films* to see films. This is implemented by a Films component.
- Click *More stuff* to see more stuff 😁. This is implemented by a MoreStuff component.

When you feel like you're ready, hit Ctrl+C in the Termina window to stop the web app.

Exercise 2: Getting started with the 'student' web app

Go to the ***student*** folder:

```
ReactDev\Student\09-SinglePageApps\library-app
```

We've implemented a lot of the core functionality, to get you started. Let's take a look…

- Book.ts
  Defines a Book class, taking advantage of TypeScript language features.

- Film.ts
  Defines a Film class, taking advantage of TypeScript language features.

- DataProvider.ts
  Defines a simple class that returns books and films. (Later in the course, you'll replace this with a proper call to a remote REST service.)

- Home.tsx
  Defines the Home component, which displays the splash screen for the home page.

- Books.tsx
  Defines the Books component, which renders books. There's nothing new here in terms of React theory, but it looks different to before. The component receives 2 properties:
  - books - an array of Book objects
  - format - a String ("TABLE", "ORDERED_LIST", or "UNORDERED_LIST")

  The component renders a <Table> or <ItemsList>, based on the format property.

- Films.tsx
  Defines the Films component, which renders all the films. This component is similar to the Books component above.

- Table.tsx
  Defines the Table component, plus related sub-components, to render a collection of objects as an HTML table. This is similar to the Table component you saw earlier in the labs, but we've tweaked it now to make use of TypeScript language features.

- ItemsList.tsx
  Defines the ItemsList component, to render a collection of objects as an HTML list. This is similar to the ItemsList component from before, but refactored for TypeScript.

- LikePanel.tsx
  Defines the LikePanel component, to enable the user to "like" the website. This is similar to the LikePanel component from before.

Exercise 3: Preparing to use React Router

Now it's time to start adding functionality to the application. The application doesn't currently support routing. To support routing, add the following dependencies to package.json:

```
{
  "dependencies": {
    "react-router-dom": "^5.2.0",
    "@types/react-router-dom": "^5.1.8",
    …
  },
  …
}
```

Once you've made this change, you'll need to install these packages. Open a Terminal window and go to the following folder:

```
ReactDev\Student\09-SinglePageApps\library-app
```

Then run the following command:

```
npm install
```

To enable routing in your application, modify the code in index.tsx so that the App component in wrapped in <BrowserRouter> (or <HashRouter>).

Run the application and verify that it works without any errors 😊.

Exercise 4: Implementing the Menu component

Open Menu.tsx in a text editor, and complete the Menu component so that it returns a menu with the following links:

/

/books

/films

/moreStuff

Exercise 5: Defining a route table

Open App.tsx in a text editor. Enhance the <App> component where indicated, to define a route table with the following routes:

Path	Component	Properties
/	Home	
/books	Books	books={books} format="TABLE"
/films	Films	films={films} format="TABLE"
/moreStuff	MoreStuff	books={books} films={films}
*****	PageNotFound	

Exercise 6: Implementing the PageNotFound component

Open PageNotFound.tsx in a text editor. Implement the PageNotFound component to display an error such as the following if the user navigates to an invalid route:

Exercise 7: Implementing the MoreStuff component

Open MoreStuff.tsx in a text editor. The purpose of the MoreStuff component is to display the following information:

Implement the MoreStuff component to achieve this effect. Here are some hints and suggestions:

- To display the number of books and the number of films, make use of the books and films properties passed in to the MoreStuff component.
- To display the "Like My Library" section, just render the LikePanel component.