Single-Page Application

Mise en place avec « React-Router »

Single-Page Application

Le package « React-Router »

Pour mettre en place une application SPA (Single-Page Application), il est conseillé d'utiliser une librairie pour faciliter sa mise en place en React.

Une des librairies les plus utilisée pour le créer une SPA est « React-Router »

Celle-ci permet:

- Définir le routing de l'application.
- Naviguer entre les différentes pages.
- Récupérer des informations depuis l'url.

Mise en place du routing

Pour ajouter le routing dans l'application, il faut installer le package « react-router » .

" **npm install —save react-router-dom** "

Ensuite, il est nécessaire de :

- Connecter l'application à l'URL.
- Créer des composants React qui symbolisent les pages de l'application.
- Permettre à l'utilisateur de naviguer entre les pages.
- Configurer les différentes routes :
  - En JSX via les composants « Routes » et « Route ».
  - En JS à l'aide du hook « useRoutes »

Connecter l'application à l'URL

En premier lieu, il est nécessaire de connecter l'application à l'URL.

Pour cela, dans le fichier « index.js », il faut encapsuler l'App avec « BrowserRouter ».

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { BrowserRouter } from 'react-router-dom';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

Ajouter des pages à l'application

Création des composants qui seront utilisés pour définir le contenu des pages

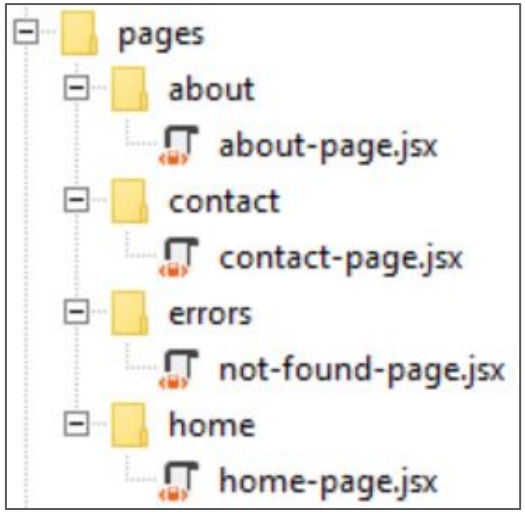
Exemple d'un composant « Page » :

```jsx
const HomePage = () => {

    return (
        <>
            <h1>Home</h1>
            <p>Lorem ipsum dolor sit ...</p>
        </>
    );
};

export default HomePage;
```

Structure de fichier :



Remarque : La structure utilisée dans ce support est « Par types d'éléments ».

Permettre à l'utilisateur de naviguer entre les pages

Le composant « Link » de React-Router permet de mettre en place de la navigation entre les pages de l'app.

Il est également possible d'utiliser le composant « NavLink ». Celui-ci à le même fonctionnement que le « Link ».

Sa particularité est qu'il détecte la route active de l'application. Cela permet d'interagir facilement le visuel.

Exemple de NavBar

```jsx
import { Link } from 'react-router-dom';
import style from './nav-bar.module.css';

const NavBar = () => (
    <nav className={style.nav}>
        <ul>
            <li>
                <Link to=''>Demo React-Router</Link>
            </li>
            <li>
                <Link to='/about'>About</Link>
            </li>
            <li>
                <Link to='/contact'>Contact</Link>
            </li>
        </ul>
    </nav>
);

export default NavBar;
```

Configuration des routes : En JSX

Exemple de création des routes en JSX via les composants « Routes » et « Route »

```jsx
const App = () => {
  return (
    <div className="App">
      <NavBar />
      <Routes>
        <Route path='' element={<HomePage />} />
        <Route path='contact' element={<ContactPage />} />
        <Route path='about' element={<AboutPage />} />
        <Route path='*' element={<NotFoundPage />} />
      </Routes>
    </div>
  );
};

export default App;
```

Configuration des routes : En JavaScript

Exemple de configuration des routes à l'aide du hook « useRoutes »

Fichier JS de configuration des routes

```
const appRoutes = [
    {
        path: '',
        element: <HomePage />
    },
    {
        path: 'contact',
        element: <ContactPage />
    },
    {
        path: 'about',
        element: <AboutPage />
    },
    {
        path: '*',
        element: <NotFoundPage />
    }
];

export default appRoutes;
```

Injection de routes à l'aide du hook

```
import { useRoutes } from 'react-router-dom';
import appRoutes from './routes';
import NavBar from './containers/nav-bar/nav-bar';

const App = () => {
  const routes = useRoutes(appRoutes);

  return (
    <div className="App">
      <NavBar />
      {routes}
    </div>
  );
};

export default App;
```

Les Routes Enfants

Single-Page Application

Objectif

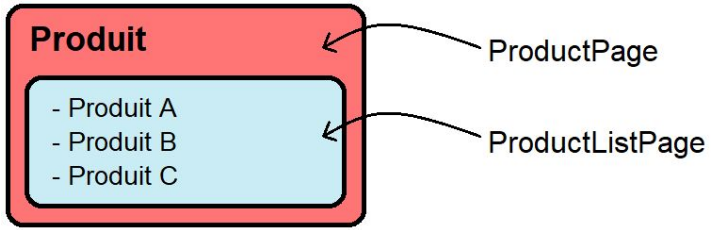
Pour éviter la répétition de code dans les différentes « pages », il est possible d'intégrer un composant dans une page qui dépendra de la route active.

Pour cela, il est nécessaire que :

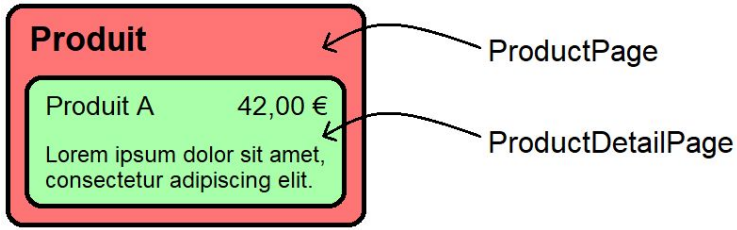
- Le composant page doit contenir un composant « Outlet »
- La configuration de la route contient les différentes routes enfants

Exemple :

➔ /exemple.com/product



➔ /exemple.com/product/42



Page avec un « Outlet »

Pour intégrer des composants dans une page à l'aide de routes enfants, il est nécessaire d'ajouter le composant « Outlet » fourni par « react-router ».

```
const { Outlet } = require('react-router-dom');

const ProductPage = () => {

    return (
        <>
            <h1>Product</h1>
            <Outlet />
        </>
    );
};

export default ProductPage;
```

Configuration de routes enfants

Exemple de configuration d'une route avec des routes enfants

En JSX

```jsx
<Routes path='product' element={<ProductPage />}>
  <Route index element={<ProductListPage />} />
  <Route path=':productId' element={<ProductDetailPage />} />
  <Route path='new' element={<ProductFormPage />} />
</Routes>
```

En Javascript (avec « useRoutes »)

```jsx
{
   path: 'product',
   element: <ProductPage />,
   children: [
      {
         index: true,
         element: <ProductListPage />
      },
      {
         path: ':productId',
         element: <ProductDetailPage />
      },
      {
         path: 'new',
         element: <ProductFormPage />
      }
   ]
},
```

La route configurée en « index » permet d'afficher le composant lorsque l'url correspond à la route parent.

Les Hooks de « React-Router »

Single-Page Application

Quelques Hooks utiles de « React-Router »

- useParams

Permet d'obtenir les données dynamiques contenues dans la route.

```
// exemple.com/product:productId
const { productId } = useParams();
```

- useSearchParams

Permet de lire ou modifier le « Query String » de l'url.

```
// exemple.com/demo?nb=42
const [searchParams, setSearchParams] = useSearchParams();
```

Quelques Hooks utiles de « React-Router »

- useNavigate

Permet d'obtenir une méthode pour naviguer par programmation.

```
const navigate = useNavigate();

const handleDemo = () => {
    // Navigation vers la page "/new-page"
    navigate('/new-page');
};
```