Série 17 – (React + MVC)

Mise en contexte

- Dans cet exercice, nous reprenons la liste des pays de la série 15, mais dans une application React;
- Nous souhaitons appliquer le modèle MVC, donc nous allons utiliser la librairie *signals*;
- Prendre connaissance du fichier *s17.html* qui illustre le résultat final souhaité.

Création de l'application

- Dans un terminal, se déplacer à l'endroit où vous voulez créer votre solution, puis exécuter ces commandes
  - npm create vite@latest s17 [React + JavaScript]
  - cd s17
  - npm i
  - npm i @preact/signals-react@1.3.8
  - npm i bootstrap
  - npm run dev

Configuration du projet

- Préparer l'arborescence de projet suivante
  - /src
    - /Composants
      - ListePays.jsx
    - /Models
      - ModelePays.jsx
    - App.jsx
    - Controller.jsx
    - main.jsx
    - Signals.jsx
  - index.html

Fichier « main.jsx »

Afin de bénéficier du CSS compilé de Bootstrap :

- Remplacer

```
import "./index.css"
```

par

```
import "bootstrap/dist/css/bootstrap.min.css"
```

Fichier « index.html »

- Modifier le titre à votre goût (*exemple* : Série 17);
- Ajouter la balise suivante sous celle qui inclut le main.jsx

```
<script type="module" src="/src/Controller.jsx"></script>
```

Fichier « Signals.jsx »

Ce fichier sert à importer la fonction 'signal' de la librairie @preact/signals-react et y déclarer les variables que l'on souhaite monitorer dans notre application. Ici, notre seule variable sera un

tableau appelé *countries* dans lequel nous allons mettre les objets pays reçus en provenance de l'API. Voici le contenu du fichier :

```
import { signal } from "@preact/signals-react";

export default class Signals {
    static countries = signal([]);
}
```

Fichier « /src/Models/ModelePays.jsx » [modèle]

Pour notre modèle, il s'agit simplement de faire le relais entre l'API des pays et notre contrôlleur. Voici le contenu du fichier, qui contient une méthode (statique) *getCountries*() retournant les données au format JSON :

```
export default class ModelePays{
    static getCountries() {
        return fetch('https://restcountries.com/v3.1/all?fields=capital,flags,ccn3')
            .then(response => response.json())
            .then(data => data);
    }
}
```

Fichier « Controller.jsx » [contrôlleur]

Nous voyons ici qu'à l'intérieur d'un événement DOMLoadedContent, le contrôlleur utilise la méthode *ModelePays.getCountries*() du modèle pour modifier notre variable *countries*. Voici le contenu du fichier :

```
import ModelePays from "./Models/ModelePays"
import Signals from "./Signals";

document.addEventListener('DOMContentLoaded', () => {
    ModelePays.getCountries().then(countriesData => {
        Signals.countries.value = countriesData;
    });
});
```

Fichier « /src/Composants/ListePays.jsx » [vue]
Ce fichier définit le composant <ListePays /> qui affiche la page telle que sur le rendu final. Voici le contenu du fichier :

```jsx
import Signals from "../Signals";

function ListePays() {

    const pays = Signals.countries.value;

    return (
        <>
            <h1 className="h1 p-5">
                Liste des capitales avec le drapeau de leur pays
            </h1>
            <div className="container d-flex flex-wrap justify-content-evenly gap-3">
                {pays.map((country) => {
                    return (
                        <div
                            key={country.ccn3}
                            className="card"
                            style={{ width: "18rem", height: "18rem" }}
                        >
                            <img
                                src={country.flags.png}
                                className="card-img-top"
                                style={{ height: "13rem" }}
                            />
                            <div className="card-body">
                                <p className="card-text">
                                    Voici le drapeau du pays ayant comme capitale :
{country.capital}
                                </p>
                            </div>
                        </div>
                    );
                })}
            </div>
        </>
    );
}

export default ListePays;
```

Fichier « App.jsx »

Il suffit de lui demander d'afficher l'unique composant. Voici son contenu :

```jsx
import ListePays from "./Composants/ListePays";

function App() {
  return (
    <ListePays />
  );
}

export default App;
```