Métaphore : Le Redux Store est comme une pâtisserie

Imagine que tu as une **grande pâtisserie**.
Dans cette pâtisserie, il y a :

- un **rayon gâteaux**
- un **rayon pains**
- un **rayon boissons**
- un **rayon pâtisseries marocaines**
- un **rayon sandwichs**

Chaque rayon est géré par **un responsable spécial (reducer)** :
un pâtissier pour les gâteaux, un boulanger pour le pain, etc.

Slice

Dans Redux Toolkit :

- **un slice = un rayon de la pâtisserie**
- **un reducer = le responsable qui gère ce rayon**
- **le store = la pâtisserie complète**

Pourquoi on ne met pas tout dans un seul reducer ?

Métaphore :

C’est comme si TOUTE la pâtisserie était gérée par **un seul employé** :

Un seul employé = Un seul reducer géant

Cet employé devrait :

- préparer le pain
- décorer les gâteaux
- faire les jus
- servir les clients
- gérer la caisse
- commander les ingrédients
- nettoyer
- réparer la machine à café…

Avec les SLICES, tout devient clair

Chaque slice gère SON domaine :

```
productSlice  → gère les produits
cartSlice     → gère le panier
userSlice     → gère l'utilisateur connecté
todoSlice     → gère les tâches
uiSlice       → gère l'interface
```

Comme chaque employé dans un rayon de la pâtisserie.

Chaque responsable (reducer) connaît **seulement son rayon**, pas les autres → donc :

- Organisation parfaite
- Code propre
- Plus facile à maintenir
- Plus facile à comprendre
- Plus facile à agrandir

Exemple simple visuel

```
--- store ---
    |
 -------------------------------------
 |           |              |
todosSlice   productSlice      cartSlice
 |           |              |
reducer      reducer          reducer
```

Si tu mets tout dans un seul reducer :

```
store
 |
gigantesqueReducerQuiFaitTout
```

Voici la solution complète d'une application Shopping Cart utilisant React + Redux Toolkit, avec tous les fichiers : slices, store, composants**.**
C'est une version simple, propre et professionnelle**,** idéale pour débutants.

1. Structure du projet

```
src/
 ├── app/
 │    └── store.js
 ├── features/
 │    ├── productsSlice.js
 │    └── cartSlice.js
 ├── components/
 │    ├── ProductList.js
 │    └── Cart.js
 └── App.js
```

2. Installation

```
npm install @reduxjs/toolkit react-redux
```

3. store.js

Le magasin global qui centralise tous les slices.

```
import { configureStore } from "@reduxjs/toolkit";
import productsReducer from "../features/productsSlice";
import cartReducer from "../features/cartSlice";

export const store = configureStore({
  reducer: {
    products: productsReducer,
    cart: cartReducer,
  },
});
```

4. productsSlice.js

La liste des produits (simples pour commencer).

```
import { createSlice } from "@reduxjs/toolkit";

const productsSlice = createSlice({
  name: "products",
  initialState: [
    { id: 1, title: "Laptop", price: 8000 },
    { id: 2, title: "Mouse", price: 150 },
    { id: 3, title: "Keyboard", price: 250 },
  ],
  reducers: {}
});

export default productsSlice.reducer;
```

5. cartSlice.js

Toute la logique du panier (add, remove, increase, decrease).

```
import { createSlice } from "@reduxjs/toolkit";

const cartSlice = createSlice({
  name: "cart",
  initialState: [],
  reducers: {
    addToCart: (state, action) => {
      const product = state.find(item => item.id === action.payload.id);
      if (product) {
        product.qty += 1;
```

```
    } else {
      state.push({ ...action.payload, qty: 1 });
    }
   },
   removeFromCart: (state, action) => {
    return state.filter(item => item.id !== action.payload);
   },
   increaseQty: (state, action) => {
    const product = state.find(item => item.id === action.payload);
    if (product) product.qty += 1;
   },
   decreaseQty: (state, action) => {
    const product = state.find(item => item.id === action.payload);
    if (product.qty > 1) product.qty -= 1;
   }
  }
});

export const { addToCart, removeFromCart, increaseQty, decreaseQty } = cartSlice.actions;
export default cartSlice.reducer;
```

6. ProductList.js

Afficher les produits + bouton “Ajouter au panier”.

```
import { useDispatch, useSelector } from "react-redux";
import { addToCart } from "../features/cartSlice";

function ProductList() {
  const products = useSelector(state => state.products);
  const dispatch = useDispatch();

  return (
   <div>
    <h2>Products</h2>

    {products.map(p => (
     <div key={p.id} style={{ border: "1px solid #ccc", margin: 10, padding: 10 }}>
      <h3>{p.title}</h3>
      <p>Price: {p.price} MAD</p>
      <button onClick={() => dispatch(addToCart(p))}>
       Add to Cart
      </button>
     </div>
    ))}
   </div>
  );
}

export default ProductList;
```

7. Cart.js

Manipulation du panier (qty++, qty--, delete).

```
import { useDispatch, useSelector } from "react-redux";
import { increaseQty, decreaseQty, removeFromCart } from "../features/cartSlice";

function Cart() {
  const cart = useSelector(state => state.cart);
  const dispatch = useDispatch();

  const total = cart.reduce((sum, item) => sum + item.price * item.qty, 0);

  return (
    <div>
      <h2>Shopping Cart</h2>

      {cart.map(item => (
        <div
          key={item.id}
          style={{ border: "1px solid #999", margin: 10, padding: 10 }}
        >
          <h3>{item.title}</h3>
          <p>Price: {item.price} MAD</p>

          <button onClick={() => dispatch(decreaseQty(item.id))}>-</button>
          <span style={{ margin: "0 10px" }}>{item.qty}</span>
          <button onClick={() => dispatch(increaseQty(item.id))}>+</button>

          <br />
          <button
            style={{ marginTop: 10 }}
            onClick={() => dispatch(removeFromCart(item.id))}
          >
            Remove
          </button>
        </div>
      ))}

      <h3>Total: {total} MAD</h3>
    </div>
  );
}

export default Cart;
```

8. App.js

Les deux composants : liste des produits + panier.

```
import ProductList from "./components/ProductList";
import Cart from "./components/Cart";

function App() {
  return (
    <div style={{ display: "flex", gap: 40 }}>
      <ProductList />
      <Cart />
    </div>
  );
}

export default App;
```

index.js (ne pas oublier Provider)

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import { Provider } from "react-redux";
import { store } from "./app/store";

const root =
ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <Provider store={store}>
    <App />
  </Provider>
);
```

Ajouter un menu en utilisant en utilisant react-router-dom