|  |  |
| --- | --- |
| **Une image contenant texte, Police, logo, Graphique  Le contenu généré par l’IA peut être incorrect.** | Année universitaire : 2025-2026  **UP-Web**  Mohammed Ali Bellili |
| **Workshop :**  **Angular 17 Avec NgRx** | |

**1. Installation NgRx dans Angular 17**

ng add @ngrx/store@17  
ng add @ngrx/effects@17  
ng add @ngrx/store-devtools@17

**2. Modèle Produit (model/produit.ts)**

export class Produit {  
 id: number;  
 libelle: string;  
 image: string;  
 prix: number;  
 type: string;  
  
 constructor(id: number, libelle: string, image: string, prix: number, type: string) {  
 this.id = id;  
 this.libelle = libelle;  
 this.image = image;  
 this.prix = prix;  
 this.type = type;  
 }  
}

**3. Actions NgRx**

import { createAction, props } from '@ngrx/store';  
import { Produit } from '../model/produit';  
  
export const loadProduits = createAction(  
 '[Produit] Load Page',  
 props<{ start: number, limit: number }>()  
);  
  
export const loadProduitsSuccess = createAction(  
 '[Produit] Load Page Success',  
 props<{ produits: Produit[] }>()  
);  
  
export const loadProduitsFailure = createAction(  
 '[Produit] Load Page Failure',  
 props<{ error: any }>()  
);

**4. Reducer NgRx**

import { createReducer, on } from '@ngrx/store';  
import \* as ProduitActions from './produit.actions';  
import { Produit } from '../model/produit';  
  
export const produitFeatureKey = 'produit';  
  
export interface ProduitState {  
 produits: Produit[];  
 error: any;  
}  
  
export const initialState: ProduitState = {  
 produits: [],  
 error: null  
};  
  
export const produitReducer = createReducer(  
 initialState,  
 on(ProduitActions.loadProduitsSuccess, (state, { produits }) => ({  
 ...state,  
 produits: [...state.produits, ...produits]  
 })),  
 on(ProduitActions.loadProduitsFailure, (state, { error }) => ({  
 ...state,  
 error  
 }))  
);

**5. Effets NgRx**

import { Injectable } from '@angular/core';  
import { Actions, createEffect, ofType } from '@ngrx/effects';  
import { ProduitService } from '../service/produit.service';  
import \* as ProduitActions from './produit.actions';  
import { catchError, map, mergeMap, of } from 'rxjs';  
  
@Injectable()  
export class ProduitEffects {  
 constructor(private actions$: Actions, private produitService: ProduitService) {}  
  
 loadProduits$ = createEffect(() =>  
 this.actions$.pipe(  
 ofType(ProduitActions.loadProduits),  
 mergeMap(({ start, limit }) =>  
 this.produitService.getProduits(start, limit).pipe(  
 map(produits => ProduitActions.loadProduitsSuccess({ produits })),  
 catchError(error => of(ProduitActions.loadProduitsFailure({ error })))  
 )  
 )  
 )  
 );  
}

**6. Sélecteurs NgRx**

import { createFeatureSelector, createSelector } from '@ngrx/store';  
import { ProduitState, produitFeatureKey } from './produit.reducer';  
  
export const selectProduitState = createFeatureSelector<ProduitState>(produitFeatureKey);  
  
export const selectAllProduits = createSelector(  
 selectProduitState,  
 state => state.produits  
);

**7. Service Produit**

@Injectable({ providedIn: 'root' })  
export class ProduitService {  
 private API = 'http://localhost:3000/produits';  
 constructor(private http: HttpClient) {}  
 getProduits(start = 0, limit = 20): Observable<Produit[]> {  
 return this.http.get<Produit[]>(\`\${this.API}?\_start=\${start}&\_limit=\${limit}\`);  
 }  
}

:

**8.produit.module.ts :**

import { NgModule } from '@angular/core';

import { CommonModule } from '@angular/common';

import { ProduitRoutingModule } from './produit-routing.module';

import { HttpClientModule } from '@angular/common/http';

import { ProduitListComponent } from './produit-list/produit-list.component';

import { StoreModule } from '@ngrx/store';

import { produitReducer, produitFeatureKey } from './ngrx/produit.reducer';

@NgModule({

declarations: [],

imports: [

CommonModule,

HttpClientModule,

ProduitRoutingModule,

ProduitListComponent,

]})

export class ProduitModule { }

**9. Composant ProduitListComponent**

import { Component, OnInit, OnDestroy, Inject } from '@angular/core';

import { CommonModule, isPlatformBrowser } from '@angular/common';

import { Store } from '@ngrx/store';

import { Observable } from 'rxjs';

import { ProduitState } from '../ngrx/produit.reducer';

import { loadProduits } from '../ngrx/produit.actions';

import { selectAllProduits } from '../ngrx/produit.selectors';

import { PLATFORM\_ID } from '@angular/core';

import { Produit } from '../models/produit.model';

@Component({

selector: 'app-produit-list',

standalone: true,

imports: [CommonModule],

templateUrl: './produit-list.component.html',

styleUrls: ['./produit-list.component.css']

})

export class ProduitListComponent implements OnInit, OnDestroy {

produits$!: Observable<Produit[]>;

currentStart = 0;

readonly limit = 20;

loading = false;

constructor(

private store: Store<ProduitState>,

@Inject(PLATFORM\_ID) private platformId: Object

) {}

ngOnInit(): void {

this.produits$ = this.store.select(selectAllProduits);

this.loadNextPage();

if (isPlatformBrowser(this.platformId)) {

window.addEventListener('scroll', this.onScroll, true);

}

}

ngOnDestroy(): void {

if (isPlatformBrowser(this.platformId)) {

window.removeEventListener('scroll', this.onScroll, true);

}

}

onScroll = (): void => {

const bottomReached =

window.innerHeight + window.scrollY >= document.body.offsetHeight - 100;

if (bottomReached && !this.loading) {

this.loadNextPage();

}

};

private loadNextPage(): void {

this.loading = true;

this.store.dispatch(loadProduits({ start: this.currentStart, limit: this.limit }));

this.currentStart += this.limit;

// simuler fin de chargement après un court délai

setTimeout(() => (this.loading = false), 300);

}

}

**10. HTML :**

<div class="container mt-4">

<h2>Liste des Produits</h2>

<table class="table table-striped table-bordered">

<thead>

<tr>

<th>Libellé</th>

<th>Image</th>

<th>Prix (€)</th>

<th>Type</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let produit of produits$ | async">

<td>{{ produit.libelle }}</td>

<td><img [src]="produit.image" alt="img" width="50" /></td>

<td>{{ produit.prix }}</td>

<td>{{ produit.type }}</td>

</tr>

</tbody>

</table>

<div class="text-center mt-3" \*ngIf="loading">

<div class="spinner-border text-primary" role="status">

<span class="visually-hidden">Chargement...</span>

</div>

</div>

</div>

**9. main.ts :**

import { provideStore } from '@ngrx/store';

import { provideEffects } from '@ngrx/effects';

import { ProduitEffects } from './app/produit/ngrx/produit.effects';

import { provideStoreDevtools } from '@ngrx/store-devtools';

import { isDevMode } from '@angular/core';

import { bootstrapApplication } from '@angular/platform-browser';

import { AppComponent } from './app/app.component';

import { appConfig } from './app/app.config';

import { produitReducer, produitFeatureKey } from './app/produit/ngrx/produit.reducer';

bootstrapApplication(AppComponent, appConfig);

**9. app.config.ts:**

import { provideStore } from '@ngrx/store';

import { provideEffects } from '@ngrx/effects';

import { ProduitEffects } from './produit/ngrx/produit.effects';

import { produitReducer, produitFeatureKey } from './produit/ngrx/produit.reducer';

import { provideStoreDevtools } from '@ngrx/store-devtools';

import { ApplicationConfig, isDevMode } from '@angular/core';

import { provideRouter } from '@angular/router';

import { provideClientHydration } from '@angular/platform-browser';

import { routes } from './app.routes';

import { provideHttpClient, withFetch } from '@angular/common/http';

export const appConfig: ApplicationConfig = {

providers: [

provideRouter(routes),

provideClientHydration(),

provideHttpClient(withFetch()),

provideStore({ [produitFeatureKey]: produitReducer }),

provideEffects(ProduitEffects),

provideStoreDevtools({ maxAge: 25, logOnly: !isDevMode() })

]

};