import { Actions, Effect } from '@ngrx/effects';

import { Recipe } from '../recipe.model';

import { Ingredient } from '../../shared/ingredient.model';

import \* as **RecipeActions** from './recipe.actions';

import \* as **fromApp** from **'../../store/app.reducers';**

export **interface** **FeatureState** extends **fromApp.AppState** {

recipes: State}

export **interface State** {

recipes: Recipe[];}

const initialState: State = {

recipes: [

new Recipe(

'Tasty Schnitzel',

'A super-tasty Schnitzel - just awesome!',

'https://upload.wikimedia.org/wikipedia/commons/7/72/Schnitzel.JPG',

[

new Ingredient('Meat', 1),

new Ingredient('French Fries', 20)

]),

new Recipe('Big Fat Burger',

'What else you need to say?', 'https://upload.wikimedia.org/wikipedia/commons/b/be/Burger\_King\_Angus\_Bacon\_%26\_Cheese\_Steak\_Burger.jpg',

[ new Ingredient('Buns', 2),

new Ingredient('Meat', 1)])

]

};

export function **recipeReducer**(**state = initialState, action: RecipeActions.RecipeActions**) {

switch (**action.type**) {

case (**RecipeActions.SET\_RECIPES**):

return {

...state,

recipes: **[...action.payload**]};

case (RecipeActions.ADD\_RECIPE):

return {

...state,

recipes: **[...state.recipes, action.payload**]};

case (**RecipeActions.UPDATE\_RECIPE**):

const recipe = state.recipes[action.payload.index];

const updatedRecipe = {

...recipe,

...action.payload.updatedRecipe};

const recipes = [...state.recipes];

recipes[action.payload.index] = updatedRecipe;

return {

...state,

recipes: recipes};

case (**RecipeActions.DELETE\_RECIPE**):

const oldRecipes = [...state.recipes];

oldRecipes.splice(action.payload, 1);

return {

...state,

recipes: oldRecipes

};

default:

return state;}

}

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

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

import { RecipesComponent } from './recipes.component';

import { RecipeStartComponent } from './recipe-start/recipe-start.component';

import { RecipeListComponent } from './recipe-list/recipe-list.component';

import { RecipeEditComponent } from './recipe-edit/recipe-edit.component';

import { RecipeDetailComponent } from './recipe-detail/recipe-detail.component';

import { RecipeItemComponent } from './recipe-list/recipe-item/recipe-item.component';

import { RecipesRoutingModule } from './recipes-routing.module';

import { SharedModule } from '../shared/shared.module';

import { recipeReducer } from './store/recipe.reducers';

import { RecipeEffects } from './store/recipe.effects';

**@NgModule({**

**declarations**: [

RecipesComponent, RecipeStartComponent,

RecipeListComponent, RecipeEditComponent,

RecipeDetailComponent, RecipeItemComponent

],

imports: [

CommonModule,

ReactiveFormsModule,

RecipesRoutingModule,

SharedModule,

StoreModule**.forFeature('recipes', recipeReducer),**

EffectsModule**.forFeature([RecipeEffects])**

]

})

export class RecipesModule {}

**note: we are adding a feature**

this will do **behind** the **scenes** is It tells ngxr to **add this reducer** and it's state to the **global application state** and the map off road users one stays component or once **this lazy loaded module**

here has been added to our application

import { **ActionReducerMap** } from '@ngrx/store';

import \* as **fromShoppingList** from '../shopping-list/store/shopping-list.reducers';

import \* as **fromAuth** from '../auth/store/auth.reducers';

export interface AppState {

shoppingList: fromShoppingList.State,

auth: fromAuth.State

}

export const reducers: **ActionReducerMap**<AppState> = {

shoppingList: **fromShoppingList**.shoppingListReducer,

auth: fromAuth.**authReducer**

};

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

import { ActivatedRoute, Params, Router } from '@angular/router';

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

import { Observable } from 'rxjs/Observable';

import 'rxjs/add/operator/take';

import \* as ShoppingListActions from '../../shopping-list/store/shopping-list.actions';

import \* as **fromRecipe from '../store/recipe.reducers';**

import \* as **RecipeActions** from '../store/**recipe**.actions';

@Component({

selector: 'app-recipe-detail',

templateUrl: './recipe-detail.component.html',

styleUrls: ['./recipe-detail.component.css']

})

export class **RecipeDetailComponent** implements OnInit {

recipeState: Observable<fromRecipe.State>;

id: number;

constructor(private route: ActivatedRoute,

private router: Router,

private store: **Store<fromRecipe.FeatureState>)** {

}

ngOnInit() {

this.route.params

.subscribe(

(params: Params) => {

this.id = +params['id'];

this.recipeState = **this.store.select('recipes');**

}

);

}

onAddToShoppingList() {

this.store.select('recipes')

**.take(1)**

.**subscribe**((recipeState: fromRecipe.State) => {

**this.store.dispatch(new ShoppingListActions.AddIngredients(**

**recipeState.recipes[this.id].ingredients)**

);

});

}

onEditRecipe() {

**this.router.navigate(['edit'], {relativeTo: this.route});**

// this.router.navigate(['../', this.id, 'edit'], {relativeTo: this.route});

}

onDeleteRecipe() {

**this.store.dispatch(new RecipeActions.DeleteRecipe(this.id));**

**this.router.navigate(['/recipes']);**

}

}

<div class="col-xs-12">

<img

[src]="**(recipeState | async).**recipes[id].imagePath"

alt="{{ **(recipeState | async).**recipes[id].name }}"

class="img-responsive" style="max-height: 300px;">

</div>

</div><div class="row">

<div class="col-xs-12">

<h1**>{{ (recipeState | async).recipes[id].name }}**</h1>

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

import { Actions, Effect } from '@ngrx/effects';

import 'rxjs/add/operator/switchMap';

import 'rxjs/add/operator/withLatestFrom';

import { HttpClient, HttpRequest } from '@angular/common/http';

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

import \* as RecipeActions from '../store/recipe.actions';

import { Recipe } from '../recipe.model';

import \* as fromRecipe from '../store/recipe.reducers';

@Injectable()

export class RecipeEffects {

@Effect()

recipeFetch = this.actions$

.ofType(RecipeActions.FETCH\_RECIPES)

.switchMap((action: RecipeActions.FetchRecipes) => {

return **this.httpClient.get<Recipe[]>('https://ng-recipe-book-3adbb.firebaseio.com/recipes.json**', {

observe: 'body',

responseType: 'json'

})

})

.**map**(

(recipes) => {

console.log(recipes);

for (let recipe of recipes) {

if (!recipe['ingredients']) {

recipe['ingredients'] = [];

}

}

**return** {

**type: RecipeActions.SET\_RECIPES,**

**payload: recipes**

};

}

);

**@Effect({dispatch: false})**

recipeStore = this.**actions**$

.**ofType**(RecipeActions.STORE\_RECIPES)

.**withLatestFrom**(this.store.select('recipes'))

.**switchMap**(([action, state]) => {

const req = new HttpRequest('PUT', 'https://ng-recipe-book-3adbb.firebaseio.com/recipes.json', state.recipes, {reportProgress: true});

return this.httpClient.request(req);

});

constructor(private actions$: Actions,

private httpClient: HttpClient,

private store: Store<fromRecipe.FeatureState>) {}

}

**I need to get my array of recipe**

**withLatestFrom**() this now allows me to combine the value I get here from of the type combine the action with the value it's an observable which will be my state