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

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

import 'rxjs/add/operator/map';

import 'rxjs/add/operator/do';

import 'rxjs/add/operator/switchMap';

import 'rxjs/add/operator/mergeMap';

import { fromPromise } from 'rxjs/observable/fromPromise';

import \* as firebase from 'firebase';

import \* as AuthActions from './auth.actions';

@**Injectable**()

export class ***AuthEffects*** {

@***Effect***()

**authSignup = this.actions$**

.**ofType**(AuthActions.TRY\_SIGNUP)

.**map**((action: AuthActions.TrySignup) => {

return action.payload;

}).**switchMap**((authData: {username: string, password: string}) => {

Return fromPromise(firebase.auth().createUserWithEmailAndPassword(authData.username, authData.password));

}).**switchMap**(() => {

return fromPromise(firebase.auth().currentUser.getIdToken());

})

.**mergeMap**((token: string) => {

return [

{ type: AuthActions.**SIGNUP**},

{ type: AuthActions.**SET\_TOKEN**,

payload: **toke** }];

});

**@*Effect*()**

**authSignin = this.actions$**

.**ofType(AuthActions.TRY\_SIGNIN**)

**.map((action: AuthActions.TrySignup**) => {

return action.payload;

})

.**switchMap**((authData: {username: string, password: string}) => {

return fromPromise(firebase.auth().signInWithEmailAndPassword(authData.username, authData.password));

})

.**switchMap**(() => {return fromPromise(firebase.auth().currentUser.getIdToken());})

.**mergeMap**((token: string) => {

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

return [

{ **type: AuthActions.SIGNIN**},

{type: **AuthActions.SET\_TOKEN**,

payload: token}];

});

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

**authLogout = this.actions$**

.**ofType**(AuthActions.**LOGOUT**)

**.do(()** => {

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

});

constructor(private actions$: Actions, private router: Router)

{}}

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

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

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

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

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

import { AppRoutingModule } from './app-routing.module';

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

import { ShoppingListModule } from './shopping-list/shopping-list.module';

import { AuthModule } from './auth/auth.module'

import { CoreModule } from './core/core.module';

import { reducers } from './store/app.reducers';

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

import { **AuthEffects** } from './auth/store/auth.effects';

@**NgModule**({

declarations: [

AppComponent

],

imports: [

BrowserModule,HttpClientModule,

AppRoutingModule, SharedModule,

ShoppingListModule, AuthModule,

CoreModule,

**StoreModule.forRoot(reducers),**

**EffectsModule.forRoot([AuthEffects])**

],

**bootstrap**: [AppComponent]

})

export class AppModule { }--------------------

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

export const TRY\_SIGNUP = 'TRY\_SIGNUP';

export const SIGNUP = 'SIGNUP';

export const TRY\_SIGNIN = 'TRY\_SIGNIN';

export const SIGNIN = 'SIGNIN';

export const LOGOUT = 'LOGOUT';

export const SET\_TOKEN = 'SET\_TOKEN';

export class TrySignup implements Action {

readonly type = TRY\_SIGNUP;

constructor(public payload: {username: string, password: string}) {}

}

export class TrySignin implements Action {

readonly type = TRY\_SIGNIN;

constructor(public payload: {username: string, password: string}) {}

}

export class **Signup** implements **Action** {

readonly type = SIGNUP;}

export class **Signin** implements **Action** {

readonly type = SIGNIN;}

export class **Logout** implements **Action** {

readonly type = LOGOUT;}

export class **SetToken** implements **Action** {

readonly type = SET\_TOKEN;

**constructor(public payload: string) {}}**

export type AuthActions = Signup | Signin | Logout | SetToken | TrySignup | TrySignin;

Asyncronous task or Side Efects

we dont want directly to alter our store though we don't want to change our store or even if

we do which alters our store this library give us a more elegant way of handling these side effects such that we don't have to leave DNG or ex world for that and for that we need to install a new package

npm install --save @ngrx/effects

ng serve

**auth.effects.ts**

because this should handle all the side effects I might have in this.

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

import { NgForm } from '@angular/forms';

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

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

import \* as **AuthActions from '../store/auth.actions';**

@**Component**({

selector: 'app-signup',

templateUrl: './signup.component.html',

styleUrls: ['./signup.component.css']

})

export class **SignupComponent** implements OnInit {

constructor(private store: Store<fromApp.AppState>) { }

ngOnInit() {

}

**onSignup**(form: NgForm) {

const email = form.value.email;

const password = form.value.password;

**this.store.dispatch(new AuthActions.TrySignup({username: ema**il, password: password}));

}

}--------------------------

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

import { NgForm } from '@angular/forms';

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

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

import \* as AuthActions from '**../store/auth.actions'**;

@Component({

selector: **'app-signin'**,

templateUrl: './signin.component.html',

styleUrls: ['./signin.component.css']

})

export class **SigninComponent** implements OnInit {

**constructor(private store: Store<fromApp.AppState>) { }**

ngOnInit() {

}

**onSignin(form: NgForm) {**

const email = form.value.email;

const password = form.value.password;

**this.store.dispatch**(new **AuthActions.TrySignin({username: email, password: password**}));

}

} – AuthService file is deleted is not getting used anymore

import { DataStorageService } from '../../shared/data-storage.service';

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

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

import \* as AuthActions from '../../auth/store/auth.actions';

@Component({

selector: 'app-header',

templateUrl: './header.component.html'

})

export class **HeaderComponent** implements **OnInit** {

authState: **Observable<fromAuth.State>;**

constructor(private **dataStorageService: DataStorageService**,

private **store: Store<fromApp.AppState>**) {

}

ngOnInit() {

this.authState = this.store.select('auth');

}

onSaveData() {

this.**dataStorageService**.storeRecipes()

.**subscribe**(

(**response**) => {

console.log(response);

}

);

}

onFetchData() {

this.dataStorageService.getRecipes();

}

onLogout() {

this.store.dispatch(new AuthActions.Logout());

}

}-----------------------

import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';

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

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

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

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

@**Injectable**()

export class **AuthGuard** implements **CanActivate** {

constructor(private store: Store<**fromApp.AppState**>) {}

**canActivate**(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {

return this.store.select('auth')

**.take(1)**

.map((authState: fromAuth.State) => {

return authState.authenticated;

});

}

}