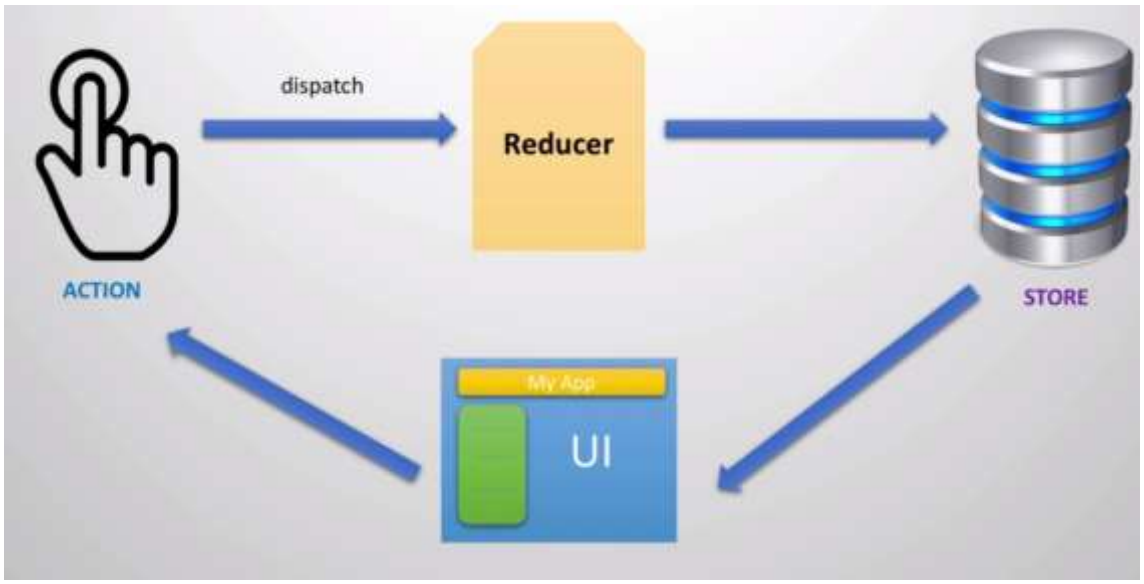
Tryssre777

Arquitectura



Instalando ngrx

npm install @ngrx/store

```
F:\proyectos angular\angular-redux>npm install @ngrx/store
```

Instalando bootstrap

```
F:\proyectos angular\angular-redux>npm install bootstrap --save
```

styles.scss

src > styles.scss

```
/* You can add global styles to this file, and also import other style files */
@import "../node_modules/bootstrap/dist/css/bootstrap.css";
```

Importando store en los módulos y creando el reducer

Store ➔ Una BD del lado del cliente. Habrá una sección, de la “BD”, llamada “mensaje”

```
import { BrowserModule } from "@angular/platform-browser";
import { NgModule } from "@angular/core";

import { AppRoutingModule } from "./app-routing.module";
import { AppComponent } from "./app.component";

// Importaciones
import { StoreModule } from "@ngrx/store";
import { miReducer } from "./app.reducer";

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    AppRoutingModule,
    StoreModule.forRoot({
      // mensaje => Porción del ngrx Store (Como una BD del lado del cliente)
      mensaje: miReducer
    })
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Creando el reducer:

```
import { Action } from "@ngrx/store";

// Modelo al estado de la aplicación
export interface appState {
  texto: String;
}

// Iniciando las propiedades del appState
export const initialState = {
  texto: "Código Mentor"
};

// Creando el miReducer => Crea un nuevo estado sin modificar el anterior (Función pu
ra)
export function miReducer(state: appState = initialState, action: Action) {
  console.log(action);
}
```

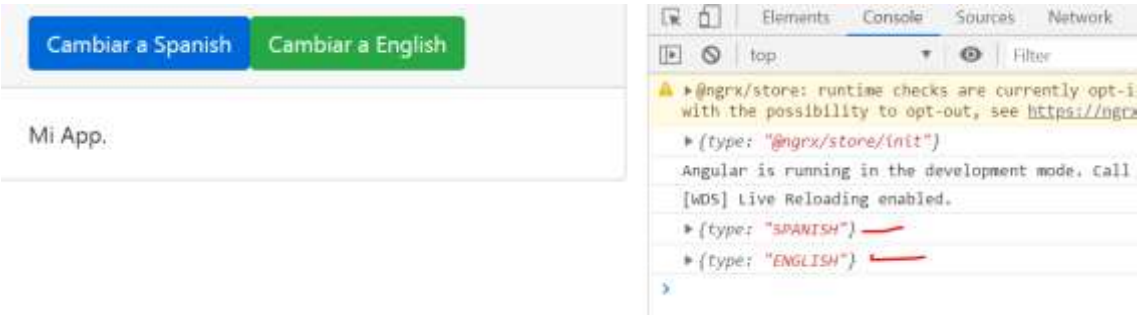
Usando los datos del store creado en el componente “app”:

```
import { Component } from "@angular/core";
import { Observable } from "rxjs";
import { Store } from "@ngrx/store";
import { appState } from "./app.reducer";

@Component({
  selector: "app-root",
  templateUrl: "./app.component.html",
  styleUrls: ["./app.component.scss"]
})
export class AppComponent {
  // Observable
  dato$: Observable<any>;

  // store => Con esto puedo acceder a la BD del lado del cliente
  constructor(private store: Store<appState>) {
    this.dato$ = store.select<any>("mensaje"); // Acceder a la porción de la BD llama
da "mensaje" (app.module)
  }

  // Métodos que despacharán (enviarán una acción)
  spanishMensaje() {
    // Emitiendo una acción => dispatch({type: "", payload: ... })
    this.store.dispatch({type: "SPANISH"});
  }

  englishMensaje() {
    this.store.dispatch({type: "ENGLISH"});
  }
}
```

En el template:

```
<div class="container">
  <div class="card">
    <div class="card-header">
      <button class="btn btn-primary" (click)="spanishMensaje()">
        Cambiar a Spanish
      </button>
      <button class="btn btn-success" (click)="englishMensaje()">
        Cambiar a English
      </button>
    </div>
    <div class="card-body">
      Mi App.
    </div>
  </div>
</div>
```

Resultado:



State - Object.assign - Spread Operator

Hay varias maneras de crear una copia de un objeto en javascript: Una es Object.assign y otra es Spred Operator.

Object.assign

```
> var obj1 = {nombre : 'Luis', hobby: 'cantar'};
< undefined
> var obj2 = Object.assign({}, obj1);
< undefined
> obj2
< ▸{nombre: "Luis", hobby: "cantar"}
> obj2.hobby = 'correr';
< "correr"
> obj2
< ▸{nombre: "Luis", hobby: "correr"}
> obj1
< ▸{nombre: "Luis", hobby: "cantar"}
```

Spred Operator

```
> var obj1 = {nombre : 'Mario', hobby: 'music'};
< undefined
> var obj2 = {...obj1};
< undefined
> obj2
< ▶ {nombre: "Mario", hobby: "music"}
> obj2.nombre = 'Juan';
< "Juan"
> obj2
< ▶ {nombre: "Juan", hobby: "music"}
> obj1
< ▶ {nombre: "Mario", hobby: "music"}
```

copia

En el reducer:

```
import { Action } from "@ngrx/store";

// Modelo al estado de la aplicación
export interface appState {
  texto: String;
}

// Iniciando las propiedades del appState
export const initialState = {
  texto: "Código Mentor"
};

// Creando el miReducer => Crea un nuevo estado sin modificar el anterior (Función pu
ra)
export function miReducer(state: appState = initialState, action: Action) {
  console.log(action);
  switch (action.type) {
    case "SPANISH":
      // Retornará un nuevo objeto: Con el estado anterior y el nuevo
      return {
        ...state, // Copia
        texto: "Holas mundos"
      };

    case "ENGLISH":
      return {
        ...state, // Copia
        texto: "Hellows worlds"
      };
    default:
      // Retorna el estado original por defecto
      return state;
  }
}
```

Luego, los componentes llamarán al store enviando (dispatch) el tipo de acción:

```
// Métodos que despacharán (enviarán una acción)
spanishMensaje() {
  // Emitiendo una acción => dispatch({type: "", payload: ... })
  this.store.dispatch({type: "SPANISH"});
}

englishMensaje() {
  this.store.dispatch({type: "ENGLISH"});
}
```

Use de store-devtools

Visualizar las acciones realizadas en la aplicación

Plugin para redux ➔ redux devtools

Instalando:

npm install @ngrx/store-devtools –save

```
F:\proyectos angular\angular-redux>npm install @ngrx/store-devtools --save
```

Importando al módulo:

```
import { BrowserModule } from "@angular/platform-browser";
import { NgModule } from "@angular/core";

import { AppRoutingModule } from "./app-routing.module";
import { AppComponent } from "./app.component";

// Importaciones
import { StoreModule } from "@ngrx/store";
import { miReducer } from "./app.reducer";
import { StoreDevtoolsModule } from "@ngrx/store-devtools";

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    AppRoutingModule,
    StoreModule.forRoot({
      // mensaje => Porción del ngrx Store (Como una BD del lado del cliente)
      mensaje: miReducer
    }),
    StoreDevtoolsModule.instrument({
      maxAge: 4 // Número de acciones máximas realziadas por la aplicación
    })
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Resultado:

dispatch actions y más

Creando la acción: mensaje.action.ts:

```
import { Action } from "@ngrx/store";

export const SPANISH = "[Mensaje] Spanish";
export const ENGLISH = "[Mensaje] English";

export class SPanishMensaje implements Action {
  readonly type = SPANISH;

  constructor(public payload: string) {}
}

export class EnglishMensaje implements Action {
  readonly type = ENGLISH;

  constructor(public payload: string) {}
}

export type MensajeActions = SPanishMensaje | EnglishMensaje;
```

Modificando el archivo “reducer”:

```
import { Action } from "@ngrx/store";
// Importando mis acciones
import * as fromMensaje from "./mensaje.action";

// Modelo al estado de la aplicación
export interface appState {
  texto: String;
}

// Iniciando las propiedades del appState
export const initialState = {
  texto: "Código Mentor"
};
```

```
// Creando el miReducer => Crea un nuevo estado sin modificar el anterior (Función pu
ra)
export function miReducer(
  state: appState = initialState,
  action: fromMensaje.MensajeActions
) {
  console.log(action);
  switch (action.type) {
    case fromMensaje.SPANISH:
      // Retornará un nuevo objeto: Con el estado anterior y el nuevo
      return {
        ...state, // Copia
        texto: action.payload
      };

    case fromMensaje.ENGLISH:
      return {
        ...state, // Copia
        texto: action.payload
      };
    default:
      // Retorna el estado original por defecto
      return state;
  }
}
```