Angular 6 NgRx Store example – Angular State Management

NgRx Store is a state management solution for Angular apps that helps us build applications by working around our app’s data (state). In this tutorial, we’re gonna look at how to work with NgRx Store, custom Actions, Reducers. Then we will practice to understand all of them in a simple practical Angular 6 example.

### NgRx Store to manage App State

#### Why we need a State container

State container helps JavaScript applications to manage state.  
=> Whenever we wanna read the state, look into only one single place – NgRx Store.  
=> Managing the state could be simplified by dealing with simple objects and pure functions.

#### NgRx Store

Store holds the current state so that we can see it as a single source of truth for our data.  
– access state using store.select(property) (property is defined at **app.module.ts** in StoreModule.forRoot()).  
– update state via store.dispatch(action).

|  |  |
| --- | --- |
|  | export interface AppState {      readonly customer: Customer[];  }    // component  import { Store } from '@ngrx/store';  import { AppState } from '../../app.state';    export class MyComponent {      customers: Observable<Customer[]>;      constructor(private store: Store<AppState>) {      this.customers = store.select('customer');    }      saveCustomer(data) {      this.store.dispatch(new ActionCreateCustomer({data}));    }  } |

#### Action

Action is payload of information that is sent to Store using store.dispatch(action).  
Action must have a type property that should typically be defined as string constants. It indicates the type of action being performed:

|  |  |
| --- | --- |
|  | import { Action } from '@ngrx/store';    export const CREATE\_CUSTOMER = 'Customer\_Create';  export const DELETE\_CUSTOMER = 'Customer\_Delete';    export class CreateCustomer implements Action {      readonly type = CREATE\_CUSTOMER;        constructor(public payload: Customer) { }  }    export class DeleteCustomer implements Action {      readonly type = DELETE\_CUSTOMER;        constructor(public id: string) { }  }    export type Actions = CreateCustomer | DeleteCustomer; |

#### Reducer

Reducer is a pure function that generates a new state based on an Action it receives. These Actions only describe what happened, but don’t describe how state changes.

|  |  |
| --- | --- |
|  | export function reducer(      state: Customer[] = [initialState],      action: Actions) {        switch (action.type) {          case CREATE\_CUSTOMER:              return [...state, action.payload];            case DELETE\_CUSTOMER:              return state.filter(({ id }) => id !== action.id);            default:              return state;      }  } |

**\*Note:** Reducer must be a pure function:  
=> From given arguments, just **calculate** the next state and return it.  
=> No side effects. No API or non-pure function calls. No mutations.

#### Import @ngrx/store and Reducer

app.module.ts

|  |  |
| --- | --- |
|  | import { StoreModule } from '@ngrx/store';  import { reducer } from './reducers/customer.reducer';    @NgModule({    declarations: [ ... ],    imports: [      ...      StoreModule.forRoot({        customer: reducer      })    ],    ...  })  export class AppModule { } |

### Practice

#### Example overview

This is a simple Angular 6 with NgRx Store Application that has:  
– AppState (app.state.ts) as the main state that is stored inside NgRx Store.  
– 2 types of Action: CREATE\_CUSTOMER and DELETE\_CUSTOMER (customer.actions.ts).  
– One Reducer (customer.reducer.ts).

We can save/remove Customer. App will update UI immediately.

>> Click on **Delete** button from any Customer:

#### Step by step

##### Install NgRx Store

Run cmd: npm install @ngrx/store.

##### Create Data Model

app/customers/models/customer.ts

|  |  |
| --- | --- |
|  | export class Customer {      id: string;      name: string;      age: number;      active: boolean;        constructor(id?: string, name?: string, age?: number, active?: boolean) {          this.id = id;          this.name = name;          this.age = age;          this.active = active;      }  } |

##### Create Actions

app/actions/customer.actions.ts

|  |  |
| --- | --- |
|  | import { Injectable } from '@angular/core';  import { Action } from '@ngrx/store';  import { Customer } from '../customers/models/customer';    export const CREATE\_CUSTOMER = 'Customer\_Create';  export const DELETE\_CUSTOMER = 'Customer\_Delete';    export class CreateCustomer implements Action {      readonly type = CREATE\_CUSTOMER;        constructor(public payload: Customer) { }  }    export class DeleteCustomer implements Action {      readonly type = DELETE\_CUSTOMER;        constructor(public id: string) { }  }    export type Actions = CreateCustomer | DeleteCustomer; |

##### Create Reducer

app/reducers/customer.reducer.ts

|  |  |
| --- | --- |
|  | import { Customer } from '../customers/models/customer';  import { Actions, CREATE\_CUSTOMER, DELETE\_CUSTOMER } from '../actions/customer.actions';    const initialState: Customer = {      id: '1',      name: 'Andrien',      age: 27,      active: true  };    export function reducer(      state: Customer[] = [initialState],      action: Actions) {        switch (action.type) {          case CREATE\_CUSTOMER:              return [...state, action.payload];            case DELETE\_CUSTOMER:              return state.filter(({ id }) => id !== action.id);            default:              return state;      }  } |

##### Create App State

app/app.state.ts

|  |  |
| --- | --- |
|  | import { Customer } from './customers/models/customer';    export interface AppState {      readonly customer: Customer[];  } |

##### Import NgRx Store

app/app.module.ts

|  |  |
| --- | --- |
|  | import { BrowserModule } from '@angular/platform-browser';  import { NgModule } from '@angular/core';    import { StoreModule } from '@ngrx/store';  import { reducer } from './reducers/customer.reducer';    import { AppComponent } from './app.component';  import { CreateCustomerComponent } from './customers/create-customer/create-customer.component';  import { CustomersListComponent } from './customers/customers-list/customers-list.component';  import { CustomerDetailsComponent } from './customers/customer-details/customer-details.component';    @NgModule({    declarations: [      AppComponent,      CreateCustomerComponent,      CustomersListComponent,      CustomerDetailsComponent    ],    imports: [      BrowserModule,      StoreModule.forRoot({        customer: reducer      })    ],    providers: [],    bootstrap: [AppComponent]  })  export class AppModule { } |

##### Create Components

###### Create Customer Component

customers/create-customer/create-customer.component.ts

|  |  |
| --- | --- |
|  | import { Component, OnInit } from '@angular/core';  import { Store } from '@ngrx/store';  import { AppState } from '../../app.state';  import { CreateCustomer } from '../../actions/customer.actions';    @Component({    selector: 'app-create-customer',    templateUrl: './create-customer.component.html',    styleUrls: ['./create-customer.component.css']  })  export class CreateCustomerComponent implements OnInit {      constructor(private store: Store<AppState>) { }      ngOnInit() {    }      saveCustomer(id, name, age) {      this.store.dispatch(new CreateCustomer(        {          id: id,          name: name,          age: age,          active: false        }      ));    }  } |

customers/create-customer/create-customer.component.html

|  |  |
| --- | --- |
|  | <div style="max-width:300px;">    <h3>Create Customers</h3>    <div class="form-group">      <input class="form-control" type="text" placeholder="id" #id>    </div>    <div class="form-group">      <input class="form-control" type="text" placeholder="name" #name>    </div>    <div class="form-group">      <input class="form-control" type="number" placeholder="age" #age>    </div>    <button class="btn btn-success" (click)="saveCustomer(id.value,name.value,age.value)">Save Customer</button>  </div> |

###### Customer Details Component

customers/customer-details/customer-details.component.ts

|  |  |
| --- | --- |
|  | import { Component, OnInit, Input } from '@angular/core';  import { Customer } from '../models/customer';  import { Store } from '@ngrx/store';  import { AppState } from '../../app.state';  import { DeleteCustomer } from '../../actions/customer.actions';    @Component({    selector: 'app-customer-details',    templateUrl: './customer-details.component.html',    styleUrls: ['./customer-details.component.css']  })  export class CustomerDetailsComponent implements OnInit {      @Input() customer: Customer;      constructor(private store: Store<AppState>) { }      ngOnInit() {    }      removeCustomer(id) {      this.store.dispatch(new DeleteCustomer(id));    }  } |

customers/customer-details/customer-details.component.html

|  |  |
| --- | --- |
|  | <div \*ngIf="customer">    <div>      <label>Name: </label> {{customer.name}}    </div>    <div>      <label>Age: </label> {{customer.age}}    </div>    <span class="button is-small btn-danger" (click)='removeCustomer(customer.id)'>Delete</span>    <hr/>  </div> |

###### Customers List Component

customers/customers-list/customers-list.component.ts

|  |  |
| --- | --- |
|  | import { Component, OnInit } from '@angular/core';  import { Observable } from 'rxjs';  import { Store } from '@ngrx/store';    import { Customer } from '../models/customer';  import { AppState } from '../../app.state';    @Component({    selector: 'app-customers-list',    templateUrl: './customers-list.component.html',    styleUrls: ['./customers-list.component.css']  })  export class CustomersListComponent implements OnInit {      customers: Observable<Customer[]>;      constructor(private store: Store<AppState>) {      this.customers = store.select('customer');    }      ngOnInit() {    }    } |

customers/customers-list/customers-list.component.html

|  |  |
| --- | --- |
|  | <div \*ngIf="customers">    <h3>Customers</h3>    <div \*ngFor="let customer of customers | async">      <app-customer-details [customer]='customer'></app-customer-details>    </div>  </div> |

##### Import Components to App Component

app/app.component.ts

|  |  |
| --- | --- |
|  | import { Component } from '@angular/core';    @Component({    selector: 'app-root',    templateUrl: './app.component.html',    styleUrls: ['./app.component.css']  })  export class AppComponent {    title = 'grokonez';    description = 'NgRx Example';  } |

app/app.component.html

|  |  |
| --- | --- |
|  | <div class="container">      <div style="color: blue;">          <h1>{{title}}</h1>          <h3>{{description}}</h3>      </div>      <div class="row">          <div class="col-sm-4">              <app-create-customer></app-create-customer>          </div>          <div class="col-sm-4">              <app-customers-list></app-customers-list>          </div>      </div>  </div> |