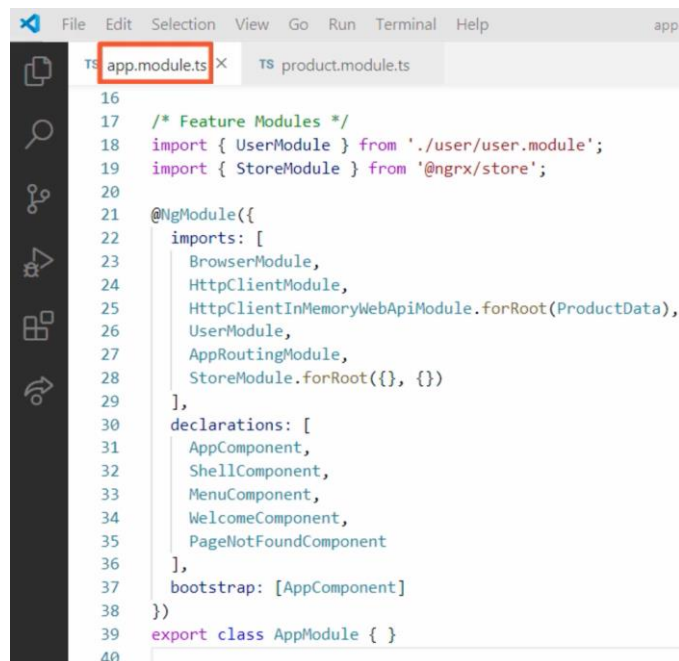


Primer vistazo a NgRx – Inicializando el Store (Demo)

En esta demo, aprovecharemos la técnica *Feature Module State Composition* e inicializaremos el Store en nuestro módulo raíz `app.module` y en nuestro módulo `feature product`.

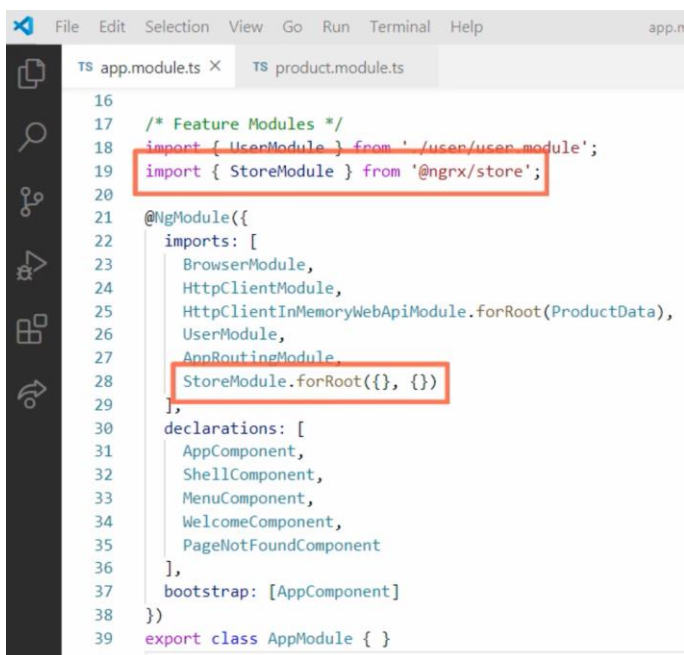
Empezaremos en el `app.module`:



```
16
17 /* Feature Modules */
18 import { UserModule } from './user/user.module';
19 import { StoreModule } from '@ngrx/store';
20
21 @NgModule({
22   imports: [
23     BrowserModule,
24     HttpClientModule,
25     HttpClientModuleInMemoryWebApiModule.forRoot(ProductData),
26     UserModule,
27     AppRoutingModule,
28     StoreModule.forRoot({}, {}),
29   ],
30   declarations: [
31     AppComponent,
32     ShellComponent,
33     MenuComponent,
34     WelcomeComponent,
35     PageNotFoundComponent
36   ],
37   bootstrap: [AppComponent]
38 })
39 export class AppModule { }
40
```

A pesar de que no tenemos ningún estado de aplicación en este punto, aun así, inicializamos el Store en nuestro módulo raíz para registrar el contenedor Store con nuestra aplicación.

Cuando instalamos el Store de `@ngrx/store` usando `ng add`, el CLI de Angular ya ha añadido el *StoreModule* en el array de imports, junto con su import respectivo:



```
16
17 /* Feature Modules */
18 import { UserModule } from './user/user.module';
19 import { StoreModule } from '@ngrx/store';
20
21 @NgModule({
22   imports: [
23     BrowserModule,
24     HttpClientModule,
25     HttpClientModuleInMemoryWebApiModule.forRoot(ProductData),
26     UserModule,
27     AppRoutingModule,
28     StoreModule.forRoot({}, {}),
29   ],
30   declarations: [
31     AppComponent,
32     ShellComponent,
33     MenuComponent,
34     WelcomeComponent,
35     PageNotFoundComponent
36   ],
37   bootstrap: [AppComponent]
38 })
39 export class AppModule { }
40
```

Estamos inicializando el Store en el módulo raíz de nuestra aplicación, por lo que llamamos a su método forRoot, pasándole dos objetos vacíos:

```
21 @NgModule({
22   imports: [
23     BrowserModule,
24     HttpClientModule,
25     HttpClientInMemoryWebApiModule.forRoot(ProductData),
26     UserModule,
27     AppRoutingModule,
28     StoreModule.forRoot({}, {}),
29   ],
```

El método `forRoot` tiene dos parámetros, el Reducer del Store y un objeto de configuración opcional. No tenemos un Reducer para el módulo raíz de nuestra aplicación, así que le pasamos un objeto vacío. Y no necesitamos ninguna configuración, por lo que podemos pasar un objeto vacío allí también, o eliminar el argumento, ya que es opcional.

Ahora nuestra aplicación tiene un Store.

A continuación, inicializamos el `StoreModule` en cada *feature module* que define el *slice* de estado. Actualmente estamos definiendo el estado del producto, por lo que lo añadimos al módulo `product.module`.

Comenzamos con la sentencia `import`:

```
view GO Debug tasks help
p.module.ts TS product.module.ts x
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

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

import { ProductShellComponent } from './product-shell/product-shell.component';
import { ProductListComponent } from './product-list/product-list.component';
import { ProductEditComponent } from './product-edit/product-edit.component';

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

const productRoutes: Routes = [
  { path: '', component: ProductShellComponent }
];

@NgModule({
  imports: [
    SharedModule,
    RouterModule.forChild(productRoutes)
  ],
  declarations: [
    ProductShellComponent,
    ProductListComponent,
    ProductEditComponent
  ]
})
```

Luego añadimos el *StoreModule* al array de imports llamando a su método *forFeature*, ya que estamos añadiendo este código a un *feature module*:

```
TS app.module.ts    TS product.module.ts x
1  import { NgModule } from '@angular/core';
2  import { RouterModule, Routes } from '@angular/router';
3
4  import { SharedModule } from '../shared/shared.module';
5
6  import { ProductShellComponent } from '../product-shell/product-shell.component';
7  import { ProductListComponent } from '../product-list/product-list.component';
8  import { ProductEditComponent } from '../product-edit/product-edit.component';
9
10 /* NgRx */
11 import { StoreModule } from '@ngrx/store';
12
13 const productRoutes: Routes = [
14   { path: '', component: ProductShellComponent }
15 ];
16
17 @NgModule({
18   imports: [
19     SharedModule,
20     RouterModule.forChild(productRoutes),
21     StoreModule.forFeature()
22   ],
```

El primer argumento es el nombre del *slice*. Como se trata de nuestro módulo de productos, lo llamaremos “products”:

```
@NgModule({
  imports: [
    SharedModule,
    RouterModule.forChild(productRoutes),
    StoreModule.forFeature('products')
  ],
```

El segundo argumento es el Reducer, o el conjunto de Reducers, que crean el estado de nuestro producto. No tenemos un Reducer para eso, así que pasaremos un objeto vacío por ahora:

```
@NgModule({
  imports: [
    SharedModule,
    RouterModule.forChild(productRoutes),
    StoreModule.forFeature('products', {})
  ],
```

Eso es todo lo que se necesita para inicializar nuestro Store.