Ole Ersoy · Follow
Feb 15, 2019 · 2 min read

Designing Angular Components For External Configuration

Photo by Federico Vitale on Unsplash

Scenario

We want users of our component to be able to be able to declare it as an NPM dependency and a design time configuration for it.

To demo this we the Stackblitz `AppComponent` as the component we are depending on, but

Inside the `AppComponent` create an exported `InjectionToken` key that will be used by the module to configure the provider.

Side Note

What we are doing here is creating a bridge via a shared key value (The `InjectionToken` instance) that allows the `AppModule` to know how to go about providing the configuration object to our component. The `AppModule.providers` array is used to configure the provision of the configuration via the shared key.

Configure the AppModule

Import the `InjectionKey` export from the `AppComponent` into the `AppModule` and use it to create the provider. Here's a stackblitz demo:

The `AppComponent` will log the injected `config` object.

Summary

declared in the component and uses it to specify the configuration in the **NgModule.providers** array.

Follow Up

If we wanted to do this more formally we could create a module for the component and place a `forRoot` static method on the module that also performs the configuration of the providers. That way we are symmetric with the approach used for the `RouterModule` .

This is how that's done (We are assuming that we also created a configuration interface called `ConfigurationInterface`):

```
static forRoot(config:ConfigurationInterface): ModuleWithProviders {

return {
  ngModule: MyComponentModule
  provider: [{ provider: tokenConfigKey, config}]
  }
}
```