Hooking into the Angular bootstrap process

Max Koretskyi aka Wizard Follow
Apr 25, 2017 · 3 min read

. . .

Angular provides several mechanisms to hook into initialization process. This article explores them and shows how they can be used.

APP_BOOTSTRAP_LISTENER

It is possible to register listeners for the Angular bootstrap process. Here is the code where they are called:

```
private _loadComponent(componentRef: ComponentRef<any>):
void {
  this.attachView(componentRef.hostView);
  this.tick();
  this._rootComponents.push(componentRef);
  // Get the listeners lazily to prevent DI cycles.
  const listeners =
      this._injector.get(APP_BOOTSTRAP_LISTENER,
[]).concat(this._bootstrapListeners);
  listeners.forEach((listener) => listener(componentRef));
}
```

This is the function that Angular calls when instantiating the application. Besides giving insights into how a component is added into the application, it also suggests that for each bootstrapped component Angular calls listeners registered under `APP_BOOTSTRAP_LISTENER` token and passes bootstrapped component to them.

It means that we can use such hooks to subscribe to the application bootstrap process and perform our initialization logic. For example, here is how `Router` hooks into to the process and executes some initialization.

Since Angular passes initialized component into the callback, we can get hold of root ComponentRef of the application like this:

```
import {APP_BOOTSTRAP_LISTENER, ...} from '@angular/core';


@NgModule({
  imports: [BrowserModule, ReactiveFormsModule,
TasksModule],
  declarations: [AppComponent, BComponent, AComponent,
SComponent, LiteralsComponent],
  providers: [{
    provide: APP_BOOTSTRAP_LISTENER, multi: true,
useFactory: () => {
      return (component: ComponentRef<any>) => {
        console.log(component.instance.title);
      }
    }
  }],
  bootstrap: [AppComponent]
})
export class AppModule {
}
```

After running into such functionality in the sources I checked the documentation and it’s described as experimental and the following description is provided:

> *All callbacks provided via this token will be called for every component that is bootstrapped. Signature of the callback:*
>
> `(componentRef: ComponentRef) => void`

APP_INITIALIZER

Angular also provides a mechanism to perform some initialization logic before it declares the application as initialized and continues with change detection and template rendering. This is where this initialization takes place:

```
constructor(@Inject(APP_INITIALIZER) @Optional() appInits:
(() => any)[]) {
  const asyncInitPromises: Promise<any>[] = [];
  if (appInits) {
    for (let i = 0; i < appInits.length; i++) {
      const initResult = appInits[i]();
      if (isPromise(initResult)) {
        asyncInitPromises.push(initResult);
      }
    }
  }
```

So, the same way we did it for the `APP_BOOTSTRAP_LISTENER` token, we just define `APP_INITIALIZER` provider and our function will be called. The following example delays Angular initialization for 5 seconds:

```
{
  provide: APP_INITIALIZER,
  useFactory: () => {
    return () => {
      return new Promise((resolve, reject) => {
        setTimeout(() => {
          resolve();
        }, 5000);
      });
    }
  },
  multi: true
}
```

And you can define multiple `APP_INITIALIZER’` s like this:

```
{
  provide: APP_INITIALIZER,
  useFactory: () => {
    return () => {
      return new Promise((resolve, reject) => {
        setTimeout(() => {
          resolve();
        }, 5000);
      });
    }
  },
  multi: true
},
{
  provide: APP_INITIALIZER,
  useFactory: () => {
    return () => {
      return new Promise.resolve(2);
    }
  },
  multi: true
}
```

BootstrapModule

Another point where you can hook into the application bootstrap process is `bootstrapModule` method:

```
platform.bootstrapModule(AppModule).then((module) => {
  let applicationRef = module.injector.get(ApplicationRef);
  let rootComponentRef = applicationRef.components[0];
});
```

Here you can get NgModuleRef to the bootstrapped module through which you can access ApplicationRef and ComponentRef.

. . .

Thanks for reading! If you liked this article, hit that clap button below 👏. It means a lot to me and it helps other people see the story. For more insights follow me on Twitter and on Medium.