Angular 2

Ángel Villalba Fdez-Paniagua

Table of Contents

1. Componentes

Los componentes son los bloques de construcción de Angular 2 que representan regiones de la pantalla. Las aplicaciones en Angular 2 se desarrollan en base a componentes. En lugar de tener un árbol de etiquetas como tenemos en las páginas web, ahora tendremos un árbol de componentes que cuelgan de un componente padre. De un componente pueden colgar uno o más componentes.

Los componentes a parte de encapsular contenido (como hacen las etiquetas), también encapsulan alguna funcionalidad. Podriamos decir que los componentes son piezas de negocio.

Un componente consta de las siguientes tres partes fundamentales:

- Un template
- Una clase
- Un decorator

A la hora de crear un componente, hay que asegurarse de que se añade en el array de **declarations** del `app.module.ts` y de que se importa correctamente. Esto hace posible que al usar la etiqueta del componente, Angular 2 sea capaz de encontrarlo y mostrarlo.

app.module.ts

```
import { MiComponente } from './mi-componente/mi-componente.component';

@NgModule({
  declarations: [
    AppComponent,
    MiComponente
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

1.1. Template

El template o plantilla representa la vista (capa V del MVC) que se escribe con HTML.

```
<h1>Mi componente</h1>
```

Se puede definir el template en una linea en lugar de definirlo en un archivo externo. Solo hay que indicarselo en el *decorator* del componente. También lo podemos hacer con los estilos.

```
// Con archivos externos para el template y los estilos
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

// Con el template y los estilos en linea
@Component({
  selector: 'app-root',
  template: `
    <h1>Mi componente</h1>
  `,
  styles: [`
    color: black;
  `]
})
```

1.2. Class

La clase de un componente se corresponde con el controlador. Es donde inicializamos y definimos el estado de los componentes. Aquí también se define el comportamiento de los componentes en forma de funciones, las cuales se asignan a los eventos del template.

```
export class AppComponent {
  title = 'app works!';

  constructor() {}

  onClick() {
    // ...
  }
}
```

1.3. Decorator

Un **decorator** es una herramienta que sirve para extender una función con mediante otra función, pero sin tocar la original que se está extendiendo. Angular 2 usa los decoradores para registrar los componentes, añadiendoles información para que sean reconocidos en otras partes de la aplicación.

```
// Con archivos externos para el template y los estilos
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

En el decorator anterior podemos distinguir las tres propiedades que le está agregando al componente. Estas propiedades describen el componente creado. Las propiedades son:

- **selector**: es el nombre de la etiqueta que se crea cuando se procesa el componente. Para mostrar el componente tenemos que llamar a la etiqueta `<app-root></app-root>` en el lugar del HTML donde queremos mostrarlo. El nombre del selector tiene que ser único en la aplicación.
- **templateUrl**: indica donde se encuentra la plantilla o template del componente.
- **styleUrls**: indica los archivos de estilos que se le van a aplicar a este componente.

1.4. Ciclo de vida de los componentes

Los componentes tienen un ciclo de vida que maneja Angular 2 usando los **hooks**. Los hooks del ciclo de vida nos permiten interactuar con los componente en momentos claves del ciclo de vida, como por ejemplo justo cuando se ha inicializado un componente, cuando este sufre cambios en alguna de sus propiedades o cuando va a ser eliminado.

Estos hooks son interfaces, las cuales tienen un método llamado como la propia interface precedido de ng.

A continuación vienen todos los hooks del ciclo de vida de un componente en el orden en que ocurren:

- **ngOnChanges**: se ejecuta cuando hay un cambio en el valor de alguna propiedad.
- **ngOnInit**: se lanza cuando se han inicializado todas las propiedades del componente, por lo que el componente también se ha inicializado.
- **ngDoCheck**: se llama en cada detección de cambios, justo después del `ngOnchanges` y `ngOnInit`.
- **ngAfterContentInit**: se ejecuta cuando se inserta contenido externo al componente en dicho componente (`<ng-content></ng-content>`).
- **ngAfterContentChecked**: cuando hay un cambio en el contenido insertado con `ng-content`.
- **ngAfterViewInit**: se lanza cuando se ha inicializado la vista del componente y las vistas de sus componente hijos.
- **ngAfterViewChecked**: se lanza despues de checkear las vistas del componente y sus hijos, es decir, justo despues del `ngAfterViewInit`.
- **ngOnDestroy**: se ejecuta justo antes de destruir un componente.

```
import { Component, OnInit} from '@angular/core';
import { Item } from '../item';
import { ItemService } from '../item.service';
// ...
export class ItemListComponent implements OnInit {
  items: Item[] = [];

  constructor(private itemService: ItemService) { }

  ngOnInit() {
    this.items = this.itemService.getItems();
  }
}
```