PEC 4
Frameworks: Introducción a Angular

Desarrollo front-end con frameworks Javascript

Máster Universitario en Desarrollo de sitios y aplicaciones web

Semestre 20201

Estudios de Informática, Multimedia y Telecomunicación

Índice

Índice2

TEORÍA3

1. Introducción a los Frameworks4
2. Tooling y código base de los ejemplos5
3. Hello Angular6
4. Directivas7
5. Componentes8

TEORÍA

1. Introducción a los Frameworks

Los frameworks son proyectos software que marcan una manera de crear aplicaciones software. En sí, son una composición de bibliotecas de código con el fin de ayudar a los desarrolladores a crear aplicaciones en un contexto determinado. En nuestro contexto, los frameworks de JavaScript permiten desarrollar la parte frontend de una web.

Los frameworks tienen como principal característica que ofrecen al equipo de desarrollo un esquema o manera de desarrollar. La dificultad de los frameworks proviene en comprender cómo funcionan, aunque la mayoría de los frameworks tienen los mismos conceptos puesto que tratan de resolver el mismo problema (proporcionar una herramienta de desarrollo), aunque suelen variar en la manera de realizar las tareas.

En JavaScript existe una gran cantidad de frameworks**,** puesto que, a diferencia de otras tecnologías**,** no ha existido un claro vencedor en la industria. No obstante, en los últimos años se puede decir que el mercado ha sido copado por dos frameworks y una biblioteca de UI. Los frameworks son Angular y Vue**,** y la biblioteca de UI es React. En el caso de que busque**s** ofertas de trabajo**,** la mayoría oscilan entre estas tres tecnologías que ayudan al desarrollo de aplicaciones frontend.

El siguiente enlace muestra una comparativa entre frameworks de JavaScript: https://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks

En nuestro caso concreto el curso continuará con la elección de Angular como framework. El motivo es que Angular tiene una gran comunidad de desarrollo (soportada y apoyada por Google) y además fuerza a usar muchas buenas prácticas que pueden ser llevadas a cualquier otro framework.

Angular

Angular es un framework que proporciona un conjunto de herramientas y técnicas de desarrollo web que permiten un desarrollo más rápido y más testeable, controlando la arquitectura de las aplicaciones web de gran envergadura. Además, Angular proporciona un patrón de trabajo que, si se conoce, facilita el poder estar en diferentes proyectos web con equipos grandes. Algunas de las características de Angular son:

- **Custom Components.** Angular permite construir tus propios componentes, lo cual va a permitir reutilizar piezas de la web, poder realizar tests y, sobre todo, manejar la complejidad.
- **Data Binding/Redux.** Angular incluye nativamente el *data binding*, lo que permite tener conexión directa entre la vista y los controladores en JavaScript/TypeScript sin tener que manipular el DOM. Del mismo

modo, permite realizar un uso avanzado de esta conexión a través de implementaciones del patrón redux (avanzado).

- **Inyección de dependencias.** Angular permite escribir servicios modulares, y estos pueden ser inyectados a los componentes o a otros servicios. Esto permite tener la lógica concentrada en un único punto que es compartido y puede ser intercambiable en la aplicación.
- **Testing.** Los tests son ciudadanos de primera clase en Angular. Es decir, Angular está construido para que tus aplicaciones puedan tener test de lógica o de UI de un modo fácil.
- Angular incluye todos los elementos necesarios para construir una **SPA**.

2. Tooling y código base de los ejemplos

Angular necesita un conjunto de herramientas (*tooling*) instaladas y configuradas en el entorno de trabajo para comenzar a crear aplicaciones. Muchas de estas herramientas las hemos trabajado en las anteriores PECs de la asignatura. La gran curva de aprendizaje de Angular realmente no es propia de Angular, sino de conocer ECMaScript en profundidad, TypeScript y el desarrollo web como aplicaciones y no como simple script. En el momento que un desarrollador proviene del entorno de crear aplicaciones web a modo de script utilizando jQuery y otras bibliotecas, se encuentra con un muro de tecnologías cuando quiere comenzar a trabajar con Angular. En resumen, Angular requiere de las siguientes herramientas instaladas y configuradas:

- node.js y npm.
- TypeScript.
- Angular CLI (No obligatorio, pero sí recomendable).

Las dos primeras herramientas las hemos trabajado en las anteriores PECs. Ahora comenzaremos a trabajar la herramienta específica de Angular.

Angular CLI

Es una herramienta de línea de comando creada y mantenida por el propio equipo de desarrollo de Angular para facilitar la creación de proyectos en Angular, creación de piezas fundamentales del framework, configuración de empaquetador (Webpack) y configuración de npm scripts.

AngularCLI se ejecuta desde la consola del sistema operativo de la siguiente manera:

```
ng <options>
```

El código fuente de esta guía es parte del libro utilizado en la asignatura. Lo podéis encontrar en el siguiente repositorio GIT:

git clone https://github.com/shyamseshadri/angular-up-and-running.git

3. Hello Angular

En este apartado se va a crear la primera aplicación en Angular usando Angular CLI. Para ello se debe ejecutar el comando de Angular CLI:

```
ng new stock-market
```

A continuación, se creará el esqueleto de la aplicación en un directorio denominado “stock-market”. Ten en cuenta que, al usar el comando anterior, Angular CLI te va a pedir que decidas varias cuestiones relacionadas con la configuración del proyecto, tales como:

- Preprocesador de CSS (SASS/LESS o CSS puro).
- Si se utiliza `Router` (Esto permite movernos entre diferentes páginas Web).

Ahora mismo es indiferente la configuración escogida. No obstante, inicialmente **se recomienda no elegir `Router`** (lo veremos más adelante en una PEC específica).

Una vez tenemos el esqueleto de la aplicación debemos poder levantarla en modo de desarrollo. Esta tarea es similar a lo que hacíamos con `nodemon` en el mundo node.js, pero en este caso concreto estaremos usando `webpack` y un `live-reload` sin que nosotros tengamos que realizar ninguna configuración para conseguir este resultado. Esto se realiza haciendo uso del siguiente comando:

```
ng serve
```

El resultado será similar al siguiente (no tiene que ser exactamente igual puesto que cambian con las versiones de Angular CLI). Se puede ver que todo el trabajo duro de *tooling* lo está haciendo Angular CLI de manera transparente para nosotros.

```
** NG Live Development Server is listening on localhost:4200,
   open your browser on http://localhost:4200/ **
Date: 2018-03-26T10:09:18.869Z
Hash: 0b730a52f97909e2d43a
```

```
Time: 11086ms
chunk {inline} inline.bundle.js (inline) 3.85 kB [entry]
[rendered]
chunk {main} main.bundle.js (main) 17.9 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js (polyfills) 549 kB
[initial] [rendered]
chunk {styles} styles.bundle.js (styles) 41.5 kB [initial]
[rendered]
chunk {vendor} vendor.bundle.js (vendor) 7.42 MB [initial]
[rendered]

webpack: Compiled successfully.
```

El resultado de lo que estamos trabajando se puede visualizar en tiempo real (cada cambio se refresca) en la dirección: `http://localhost:4200`. Ten en cuenta que esta dirección y el modo es de desarrollo ahora mismo. Esta web no puede ser desplegada tal cual, eso se verá más adelante (asignatura avanzada).

Para la superación de esta práctica debes leer y realizar el paso a paso del código relativo al Capítulo 2 (https://learning.oreilly.com/library/view/angular-up-and/9781491999820/ch02.html):

- Basics of an Angular application.
- Creating a Component.

4. Directivas

Una directiva en Angular permite agregar funcionalidades a elementos de tu HTML. Un componente en Angular, como el que se ha construido en relación al `stock-item` en la sección anterior, es una directiva que proporciona tanto funcionalidades como lógica de la interfaz. Un componente se puede resumir como un elemento que encapsula la lógica de comportamiento y renderizado (i.e. la visualización de la interfaz de usuario).

Por otro lado, las directivas que no son componentes permiten modificar los elementos existentes. Estas directivas que no son componentes pueden clasificarse en dos grandes categorías:

- **Atributo**: Las directivas de atributo permiten cambiar el aspecto o el comportamiento de un elemento existente o incluso de un componente. Un ejemplo de estas directivas son `NgClass` o `NgStyle`.

- **Estructurales**: Estas directivas cambian el DOM añadiendo o eliminando elementos a la vista. `NgIf` y `NgFor` son ejemplos de directivas estructurales.

A continuación lee y desarrolla los ejemplos del libro de texto relativos a **Built-In Attributes Directives** y **Built In Structural Directives que encontrarás en la siguiente dirección:** https://learning.oreilly.com/library/view/angular-up-and/9781491999820/ch03.html. De esta manera podrás ver en acción las directivas `NgClass`, `NgStyle`, `NgIf` y `NgFor`.

5. Componentes

En la sección anterior se especificó que Angular dispone de dos tipos de directivas, de atributo y estructurales, las cuales permiten cambiar el comportamiento de un elemento existente o cambiar la estructura de un *template* (i.e. vista/interfaz).

El tercer tipo de directiva que dispone Angular son los componentes, los cuales están siendo utilizados desde los inicios de Angular, puesto que una aplicación de Angular es un árbol de componentes. Cada componente tiene su propio comportamiento y *template* (o vista) asociado para renderizar. Este *template* puede a su vez utilizar otros componentes permitiendo crear una estructura de componentes.

Un componente es una clase que permite encapsular el comportamiento (p.ej. carga de datos, cómo responder a las interacciones de los usuarios, etc.) y el *template* (cómo los datos se van a visualizar). Existen diferentes parámetros de configuración de un componente lo que permite una gran flexibilidad de trabajo.

El capítulo del libro que debes leerte para la comprensión de estos conceptos es el siguiente: https://learning.oreilly.com/library/view/angular-up-and/9781491999820/ch04.html

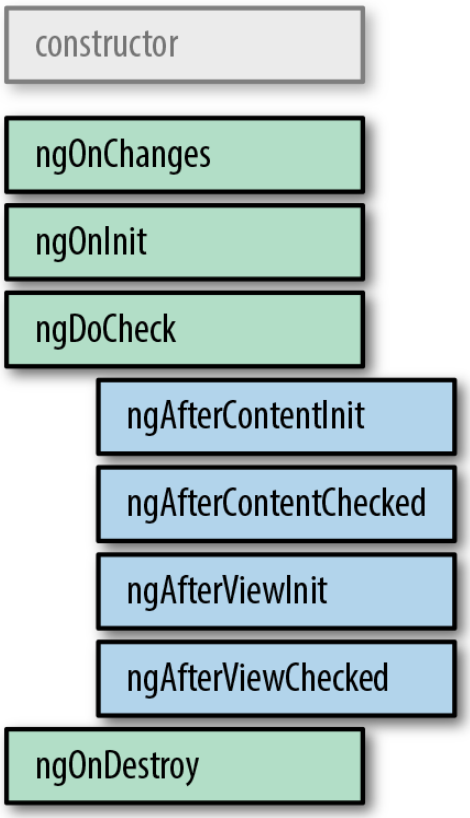
El desarrollo frontend basado en componentes tiende a que se creen muchos componentes de pequeña envergadura. Esto permite que puedan reutilizarse y realizar tests sobre los mismos. La reutilización de componentes se consigue siempre y cuando no tengamos escrito en el código fuente de los componentes los valores que los componen, sino que podamos enviar parámetros de configuración de entrada y poder obtener mensajes o eventos de salida de los componentes hacia arriba en el árbol de componentes que debemos construir.

Angular proporciona dos *hooks* específicos para estas tareas: **`Input`** y **`Output`**. Estos decoradores no se aplican sobre una clase (la que define el componente), sino que se aplican sobre atributos de la clase.

Lee la sección concreta dedicada a **Input y Output** en el capítulo https://learning.oreilly.com/library/view/angular-up-and/9781491999820/ch04.html.

Otro punto muy importante que hay que conocer es el ciclo de vida de los componentes. Si recordamos, los componentes nos están enmascarando muchas de las tareas que teníamos que realizar en el DOM, las cuales se mezclan con lógica de comportamiento. De igual manera que en el desarrollo de dispositivos móviles existe un ciclo de vida de instanciación de los componentes visuales de la pantalla, en Angular disponemos de un ciclo de vida de los componentes. De este ciclo de vida es muy importante entender en qué orden se van disparando los diferentes *hooks* que se dispone. Este ciclo de vida es ejecutado en preorder en el árbol de componentes, es decir, desde arriba hacia abajo. Después de que Angular renderiza un componente, este comienza el ciclo de vida para cada uno de sus hijos, así hasta que la aplicación entera es renderizada.

En la siguiente figura se muestran los *hooks* del ciclo de vida de un componente. No obstante, debes leer la documentación oficial o desde el libro cuándo se dispara cada uno de los *hooks* y así comprender perfectamente el orden y el momento en que se ejecutan. Estos *hooks* nos van a dar una flexibilidad y control en la instanciación de componentes sin tener que bajar al nivel del desarrollo del DOM.



NOTA: Como miembros de la Comunidad UOC, tenéis disponible un gran catálogo de recursos de la editorial O'Reilly. Para acceder debéis seguir los siguientes pasos:

* Acceder a https://learning.oreilly.com

* Cuando te pida el *login* y *password*, escribir en el *campo Email Address or Username* vuestro correo electrónico de la UOC, p.ej. dgarciaso@uoc.edu

* Después pulsáis fuera con el ratón y desaparecerá el campo *password*. Además el botón de *Sign In* cambiará por un botón rojo con el texto *Sign In with Single Sign On*.

* A partir de aquí seguid los pasos que os vaya pidiendo, seguramente os llevará a una página de autenticación con el logo de la UOC. Además os llegará un e-mail a vuestra cuenta de correo electrónico de la UOC avisando de que os habéis suscrito a este catálogo.

* Una vez tengáis acceso al catálogo de O'Reilly, para ver el contenido de los enlaces que os proporcionamos, sólo tenéis que estar autenticados en O'Reilly y abrir una pestaña nueva en vuestro navegador y copiar el enlace que os facilitamos.