PEC 6
Frameworks: Servicios en Angular

Desarrollo front-end con frameworks Javascript

Máster Universitario en Desarrollo de sitios y aplicaciones web

Semestre 20201

Estudios de Informática, Multimedia y Telecomunicación

TEORÍA

1. Introducción a los servicios en Angular

En la anterior PEC se desarrollaron formularios (dirigidos por *template* y reactivos) abordando la capa de UI del usuario. En esta PEC vamos a desplazarnos por debajo de la UI y explorar los servicios en Angular. Angular dispone inherentemente de la inyección de dependencias sin necesidad de realizar ninguna tarea por parte de los desarrolladores. Esto nos permitirá crear código más organizado, reusable y sobre todo limpio (*clean code*).

Hasta el momento se ha desarrollado la aplicación utilizando solamente componentes, pero los componentes tienen como finalidad decidir qué datos mostrar y cómo mostrarlos en la UI. Es decir, en los componentes se relacionan los datos con la UI, se enlazan los eventos a los métodos y se permite la interacción del usuario con nuestra aplicación. Por tanto, los componentes en Angular se encuentran en la capa de presentación, y por consiguiente, sólo deberían centrarse en aspectos relacionados con la presentación de los datos.

Así pues, si los componentes se encuentran en la capa de presentación, ¿qué elemento es el responsable de gestionar la lógica de negocio de nuestra aplicación o la interacción con el servidor? Aquí es donde surge la necesidad de los servicios.

Los servicios en Angular forman la capa que es común a toda la aplicación, es decir, los servicios pueden ser usados en varios componentes. El uso común de los servicios serían los siguientes:

- Necesitas recuperar/enviar datos al servidor. Esta tarea puede incluir el procesamiento o no de los datos mientras estos son transferidos.
- Necesitas encapsular la lógica de la aplicación que no es específica de un componente, o la lógica puede ser reutilizada por varios componentes.
- Necesitas compartir datos entre componentes, especialmente entre componentes que pueden (o no) conocerse entre ellos.

Por defecto, los servicios son únicos en toda la aplicación (son *singleton*), lo que permite almacenar el estado de la aplicación para poder ser accedido entre varios componentes.

De manera genérica se puede resumir que los servicios son la capa que abstrae el “cómo” se realizan las tareas de los componentes (capa visual).

2. Servicios en Angular

Los servicios son una pieza fundamental en el ecosistema de Angular y, por ello, vamos a profundizar en los mismos. Para la superación de esta práctica debes leer y realizar el paso a paso del código relativo al Capítulo 8 (https://learning.oreilly.com/library/view/angular-up-and/9781491999820/ch08.html).

Algunas de las tareas que se pueden realizar utilizando los servicios serían los siguientes:

- Recuperar la lista de vinos para mostrarlas desde el servicio, en lugar de escribirlas en el componente.
- Cuando se crea un vino, esta se enviará al servicio.
- Cuando se crea un vino, queremos actualizar nuestra lista de vinos.

El esqueleto de un servicio es el siguiente:

```
import { Injectable } from '@angular/core';

@Injectable()
export class StockService {

  constructor() { }

}
```

Observa que se ha utilizado el decorador `@Injectable`, el cual nos va a permitir transformar cualquier clase a un servicio, que podrá ser inyectado a través de los constructores de cualquier componente o servicio.

3. RxJS y Observables: Introducción

En JavaScript estamos acostumbrados a manejar el asincronismo haciendo uso de *callbacks* y promesas**,** pero en ambas herramientas lo que se está gestionando es un único dato de manera asíncrona.

En las interfaces de usuario se deben manejar secuencias de datos asíncronos en el tiempo. Un ejemplo claro es la pulsación de varios clicks sobre un elemento de la pantalla, cada vez que se hace click sobre el elemento se emitirá un dato, y asíncronamente (**no sabemos cuánto** tiempo pasará entre cada click) se irán haciendo clicks y, ahí es donde encontramos una secuencia de datos asíncronos. En esta asignatura no vamos a profundizar en **RxJS**, pero es una pieza **fundamental** en el desarrollo

frontend hoy en día (en la asignatura avanzado se profundizará), sino que vamos a solo dar una breve introducción para poder usarlos con Angular.

Los **arrays** y otras estructuras de datos son síncronas, y bloquean el hilo principal de la aplicación. No obstante, no debemos nunca bloquear el hilo, aunque sea para transmitir un dato. De ahí que haya que usar mecanismos asíncronos.

Hoy en día, la biblioteca que se **ha convertido en un estándar** de facto en el desarrollo frontend es RxJS**,** la cual implementa el patrón `Observable` e `Iterator`.

Para la comprensión de este apartado, aparte de leer el capítulo 8 de nuestro libro, es **recomendable que utilices el siguiente recurso**:

https://learning.oreilly.com/library/view/build-reactive-websites/9781680506495/f_0011.xhtml#chp.creatingObservables

En este apartado es muy importante comentar que el desarrollo web avanza a una velocidad que no da tiempo a generar materiales didácticos. La mayoría de los **recursos que se encuentran en la web hacen uso de RxJS 5** (o inferiores). La **versión actual 6.x** ha hecho *changes break* en el uso de sus métodos, no en los conceptos. Por tanto, los operadores se denominan de otra manera y se encuentran de un modo *pipeable* (descompuestos en lugar de en un macro objeto denominado `Observable`). Por tanto, **habrá que ir adaptando el uso de dicha biblioteca**.

En el siguiente enlace está **la documentación oficial de RxJS** desde la cual se puede ver la signatura de los operadores y objetos. Así como una pequeña guía de migración (no **automática** en nuestro caso).

https://rxjs-dev.firebaseapp.com/guide/v6/migration

4. Conectando frontend y backend: Http calls in Angular

Angular incluye un servicio muy popular para **realizar llamadas asíncronas con el servidor: `HttpClient`**. Este servicio es utilizado por la mayoría de los servicios que necesitan **conectarse con un backend** puesto que proporciona todos los verbos de HTTP. Además permite la configuración de cabeceras avanzadas e incluso de interceptores que permiten reutilizar el código cuando se están realizando llamadas **RESTful** API.

Debido a la importancia de este servicio, existe un capítulo completo dedicado a explicar dicho servicio. Es de obligada **lectura y comprensión** el siguiente capítulo para poder realizar la práctica:

https://learning.oreilly.com/library/view/angular-up-and/9781491999820/ch09.html

5. Pipes

Los pipes son una **herramienta de Angular** que nos permite **transformar visualmente la información**, por ejemplo, cambiar un texto a mayúsculas o minúsculas, o darle formato de fecha y hora.

Angular trae una serie de pipes **por defecto** (documentación oficial aquí) pero también nos permite **construir nuestros propios pipes**.

Para ampliar conocimientos sobre los pipes y como crear nuevos pipes para nuestros proyectos podéis consultar este capítulo del video curso Angular – The Complete Guide 2020.

https://learning.oreilly.com/videos/angular-the/9781788998437/9781788998437-video17_1

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.