BEEVA - CURSO DE ANGULAR 2015
TEORÍA (KATA NIVEL AVANZADO)
Fernando Castro / Juan Ferrer

SERVICIOS

DEFINICIÓN

Los servicios son objetos utilizados para organizar y compartir información entre las diferentes partes de la aplicación. Se hace uso de ellos mediante la inyección de dependencias, y hay de diferentes tipos, con propósitos muy definidos.

Algunas de sus características principales son las siguientes:

- Se crea una única instancia en cada dependencia (Singleton).
- El nombre con el que creamos un servicio dentro de un determinado módulo es independiente del mismo, por lo que para utilizarlo nos bastará con inyectar directamente el nombre del servicio.
- La nomenclatura seguirá el mismo estándar que cuando creamos variables en Angular, el **CamelCase**.
- Los servicios nativos de Angular siempre empiezan con el símbolo del $, como por ejemplo, $http, $timeout...

SERVICIOS NATIVOS

Algunos de los servicios que nos proporciona angular, y que nos permiten agilizar determinados procesos:

- **$http**: facilita la comunicación con servidores HTTP remotos
- **$filter**: nos permite formatear y filtrar los datos mostrados por el usuario
- **$interval**: es la conversión para Angular del setInterval de window, permitiendo la ejecución de determinado código (n veces) cada x tiempo
- **$timeout**: es la conversión para Angular del setTimeout de window, permitiendo la ejecución de determinado código (1 vez) pasado un tiempo determinado
- **$location**: parsea una URL en la barra de direcciones, y permite el acceso a la aplicación
- **$animate**: permite utilizar una serie de métodos que nos dan acceso a elementos de animación
- **$anchorScroll**: nos permite hacer scroll a determinados elementos
- **$q**: ayuda a ejecutar funciones de manera asíncrona, utilizando los valores devueltos (éxito o fracaso) cuando estén disponibles (promesas)

TIPOS DE SERVICIOS QUE PODEMOS CREAR

Angular nos permite crear **cinco tipos diferentes de servicios**, cada uno con sus especificaciones y usos más adecuados:

- constant
- value
- service
- factory
- provider

Para poder entender algunas de las diferencias existentes entre estos tipos de servicios, veamos primero dos **diferentes tipos de bloques** (que usaremos a través de sus métodos, de mismo nombre) que podemos definir dentro de un módulo, serán generales para toda la aplicación, y se ejecutarán al inicio del programa. Se pueden crear tantos bloques como sean necesarios.

- **config**
  - nos permite configurar los provider

```
angular.module('myModule', [])

  .config(function($provide, $compileProvider, $filterProvider) {
    $provide.value('a', 123);
    $provide.factory('a', function() { return 123; });
    $compileProvider.directive('directiveName', ...);
    $filterProvider.register('filterName', ...);
  });
```

  - se pueden inyectar constant y provider

- **run**
  - sería una especie de main de la aplicación, donde podremos definir cualquier función
  - se ejecutan posteriormente a los bloques config
  - se suele definir código que resulta complejo testear de forma unitaria
  - se puede acceder al $rootScope
  - se puede inyectar cualquier cosa menos provider

CONSTANT

Nos permite crear valores que podremos inyectar en cualquier parte de la aplicación. Se pueden definir objetos, funciones, cadenas, números…

```
angular.module('myModule', [])

  .constant("miServicioConstante", {

        VARIABLE1: 'dato 1',
        VARIABLE2: 'dato 2'
  });
```

VALUE

Este servicio nos permite definir valores exactamente igual que constant, la diferencia radica en dónde podremos utilizarlo: en cualquier sitio excepto en los bloques config y los servicios provider.

```
angular.module('myModule', [])

  . value("miServicioValue", {
        VARIABLE1: 'dato 1',
        VARIABLE2: 'dato 2'
  });
```

SERVICE

En este caso, definiremos una clase, y Angular será el encargado de instanciarlo, proporcionándolo cuando lo inyectemos en cualquier elemento. Se suele utilizar para exponer funcionalidad utilizada en varios lugares (por ejemplo, para llamadas a servicios de datos externos, creación de clases para los modelos…).

Al tratarse de una clase, podremos definir métodos, variables, constructor….

```
angular.module('myModule', [])

  .value("miValueInit", {
        value: 2
  })

  . service("miServicio", [miValueInit, function(miValueInit) {

        this.value = miValueInit.value;

        this.setValue = function(value) {
                this.value = value;
        };

        this.getValue = function() {
                return this.value;
        }
  }]);
```

FACTORY

Más propicio para crear objetos complejos, se diferencia del tipo ‘service’ en que devolveremos lo creado dentro de la factoría.

```
angular.module('myModule', [])

  .factory("miFactoria", function() {

        var data = {
                name: '',
                postal_code: '',
                address: function() {
                        // code
                }
        }
        return data;
  });
```

PROVIDER

Se trata de un servicio parecido al tipo 'factory', pero permitiendo una configuración inicial antes de crear el valor del servicio.

El servicio provider está formado por dos partes:

- una clase javascript de la que se crea una única instancia, y cuya función constructora (a la que llamamos **provider**) permite la configuración del servicio
  - se pueden inyectar: constant y otros providers (de otros módulos)

- una factoría (**factory-provider**), quien realmente crea el valor del servicio
  - se pueden inyectar: constant, value, service, factory y factory-provider

La configuración inicial se realiza desde el bloque config.

```
app.provider("prueba", function() {

        var _type = '';

        // constructor (provider)
        this.setType = function(value) {
                _type = value;
        }

        // factory provider
        this.$get = function() {

                var op;

                if(_type === 'suma') {
                        op = SUMA;
                } else if(_type === 'resta') {
                        op = RESTA;
                }

                var op_result = function(value1, value2) {
                        return op(value1, value2);
                }

                return op_result;
        }
})

.constant("operation","suma");


app.config(["pruebaProvider", "operation" ,function(pruebaProvider, operation) {

    pruebaProvider.setType(operation);
}]);
```

BIBLIOGRAFÍA

Se ha recurrido al abundante material disponible en la red para explicar de la mejor manera posible los diferentes conceptos que aparecen en esta documentación:

Página principal de Angular: https://angularjs.org
Filtros: http://cursoangularjs.es/doku.php?id=unidades:05_filtros:00_start
Servicios: http://nightdeveloper.net/service-factory-angular/
http://cursoangularjs.es/doku.php?id=unidades:03_servicios:00_start