





# Servicios en Angular

Por [Miguel Angel Alvarez](#)Seguir a [@midesweb](#) 04 de diciembre de 2017  [Frameworks Javascript](#)

**Qué son los servicios en el framework Javascript Angular, cómo crear nuestros primeros services, cómo usarlos desde los componentes.**

Hasta ahora en el [Manual de Angular](#) hemos hablado mucho de componentes, y también de módulos. Pero existen otros tipos de artefactos que podemos usar para organizar el código de nuestras aplicaciones de una manera más lógica y fácil de mantener.

En importancia a componentes y módulos le siguen los servicios. Con este artículo iniciamos su descripción. Los servicios, o "services" si lo prefieres en inglés, son una de las piezas fundamentales en el desarrollo de aplicaciones Angular. Veremos qué es un servicio y cómo dar nuestros primeros pasos en su creación y utilización en un proyecto.



## Concepto de servicio en Angular

Si eres ya viejo conocido del desarrollo con Angular sabrás lo que es un servicio, pues es una parte importante dentro de la arquitectura de las aplicaciones con este framework. Pero si eres nuevo seguro que te vendrá bien tener una pequeña descripción de lo que es un servicio.

Básicamente hemos dicho anteriormente que el protagonista en las aplicaciones de Angular es el componente, que las aplicaciones se desarrollan en base a un árbol de componentes. Sin embargo, a medida que nuestros objetivos sean más y más complejos, lo normal es que el código de los componentes también vaya aumentando, implementando mucha lógica del modelo de negocio.

En principio esto no sería un problema, pero sabemos que la organización del código, manteniendo piezas

pequeñas de responsabilidad reducida, es siempre muy positiva. Además, siempre llegará el momento en el que dos o más componentes tengan que acceder a los mismos datos y hacer operaciones similares con ellos, que podrían obligarnos a repetir código. Para solucionar estas situaciones tenemos a los servicios.

Básicamente un servicio es un proveedor de datos, que mantiene lógica de acceso a ellos y operativa relacionada con el negocio y las cosas que se hacen con los datos dentro de una aplicación. Los servicios serán consumidos por los componentes, que delegarán en ellos la responsabilidad de acceder a la información y la realización de operaciones con los datos.

## Cómo crear un servicio

Tal como viene siendo costumbre en el desarrollo con Angular, nos apoyaremos en [Angular CLI](#) para la creación del esqueleto, o scaffolding, de un servicio.

Para crear un servicio usamos el comando "generate service", indicando a continuación el nombre del servicio que queremos generar.

```
ng generate service clientes
```

Esto nos generaría el servicio llamado "ClientesService". La coletilla "Service", al final del nombre, te la agrega Angular CLI, así como también nombra al archivo generado con la finalización "-service", para dejar bien claro que es un servicio.

Es habitual que quieras colocar el servicio dentro de un módulo en concreto, para lo que puedes indicar el nombre del módulo, una barra "/" y el nombre del servicio. Pero atención: ahora lo detallaremos mejor, pero queremos advertir ya que esto **no agregará el servicio al código de un módulo concreto**, sino que colocará el archivo en el directorio de ese módulo. Enseguida veremos qué tienes que hacer para que este servicio se asigne realmente al módulo que desees.

```
ng generate service facturacion/clientes
```

**Nota:** como los servicios son algo que se suele usar desde varios componentes, muchos desarrolladores optan por crearlos dentro de un módulo compartido, que puede llamarse "común", "shared" o algo parecido.

Ahora, si quieres, puedes echarle un vistazo al código básico de un service, generado por el CLI. Enseguida lo examinamos. No obstante, antes queremos explicar otro paso importante para que el servicio se pueda usar dentro de la aplicación.

## Agregar la declaración del servicio a un módulo

Para poder usar este servicio es necesario que lo agregues a un módulo. Inmediatamente lo podrás usar en cualquiera de los componentes que pertenecen a este módulo. Este paso es importante que lo hagas, puesto que, al contrario de lo que ocurría al crear un componente con el CLI, la creación de un servicio no incluye la modificación del módulo donde lo has creado.

Así pues, vamos a tener que declarar el servicio manualmente en el módulo. Lo haremos gracias al decorador del módulo (@NgModule), en el array de "providers". El decorador de un módulo con el array de providers

```
@NgModule({
  imports: [
    CommonModule
  ],
  declarations: [ListadoClientesComponent],
  providers: [ClientesService]
})
```

Obviamente, tendrás que hacer el correspondiente import al módulo, para que se conozca la clase ClientesService.

```
import { ClientesService } from './clientes.service';
```

Como decimos, ahora este módulo ha declarado el servicio como "provider", por lo que sus componentes podrán usar este servicio.

**Nota:** El provider que acabamos de declarar en el módulo es un array que también podremos declarar en un componente, con lo que podríamos asignar un servicio a un componente en concreto y no a un módulo completo. Más adelante explicaremos dónde puedes declarar el provider, dependiendo de cómo quieres que tu servicio se gestione a nivel de aplicación.

## Código básico de un service en Angular

Ahora podemos examinar el código generado para nuestro servicio y aprender nuevas cosas de Angular. Este sería nuestro recién creado servicio "ClientesService".

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

@Injectable()
export class ClientesService {

  constructor() { }

}
```

Como verás, el servicio no tiene nada todavía, solo su declaración, pero hay cosas interesantes que tenemos que explicar, principalmente un nuevo decorador que no habíamos conocido hasta el momento: "@injectable".

El decorador @injectable indica a Angular que la clase que se decora, en este caso la clase ClientesService, puede necesitar dependencias que puedan ser entregadas por inyección de dependencias. De momento puedes quedarte que los servicios necesitan de este decorador, aunque realmente disponer de él no es condición indispensable.

**Nota:** La inyección de dependencias es un patrón de desarrollo bastante habitual en el mundo de los frameworks. Ya es familiar para los antiguos desarrolladores de Angular, pues estaba ya implementada en las versiones de AngularJS (1.x).

Sin embargo, no es algo propio o específico de Angular, sino de la programación en general. Tenemos un artículo que explica de manera teórica más cosas importantes sobre este patrón de diseño de software: [Inyección de dependencias](#).

El import, arriba del todo, { Injectable } from '@angular/core', lo que hace es que nuestra clase conozca y sea capaz de usar el decorador @injectable.

Por lo demás, el servicio está vacío, pero le podemos poner ya algo de código para que nos sirva de algo. De momento vamos a exponer mediante el servicio una simple propiedad con un dato, que luego vamos a poder consumir desde algún componente. Para ello usamos las propiedades de la clase, tal como nos permite TypeScript.

```
export class ClientesService {  
  accesoFacturacion = 'https://login.example.com';  
  constructor() { }  
}
```

En nuestra clase ClientesService hemos creado una propiedad llamada "accesoFacturacion", en la que hemos asignado un valor que sería común para todos los clientes. El dato es lo de menos, lo interesante es ver que la declaración no dista de otras declaraciones en clases que hayamos visto ya.

## Cómo inyectar dependencias de servicios

Ahora nos toca ver la magia de Angular y su inyección de dependencias, que vamos a usar para poder disponer del servicio en un componente.

Como en otros frameworks, en Angular la inyección de dependencias se realiza por medio del constructor. En el constructor de un componente, que hasta ahora habíamos dejado siempre vacío, podemos declarar cualquiera de los servicios que vamos a usar y el framework se encargará de proporcionarlo, sin que tengamos que realizar nosotros ningún trabajo adicional.

Esto es tan sencillo como declarar como parámetro la dependencia en el constructor del componente.

```
constructor(private clientesService: ClientesService) { }
```

De esta manera estamos indicando a TypeScript y Angular que vamos a usar un objeto "clientesService" que es de la clase "ClientesService". A partir de entonces, dentro del componente existirá ese objeto, proporcionando todos los datos y funcionalidad definida en el servicio.

**Nota:** Es muy importante que al declarar la inyección de dependencias en el constructor no te olvides de la declaración de visibilidad (ya sea public o private), pues si no la colocas no se generará la propiedad en el objeto construido. Si defines la propiedad de la clase como pública, afectará a su visibilidad: constructor(public clientesService: ClientesService) esto es algo que te proporciona TypeScript. Público o privado el efecto será el mismo, se creará la propiedad "clientesService", inyectando como valor un objeto de la clase ClientesService.

## Explicando con detalle la declaración de propiedades implícita en el constructor

Aquí tenemos que detenernos para explicar algo de la magia, o azúcar sintáctico, que te ofrece TypeScript.

Porque el hecho que se declare una propiedad en un objeto solo porque se reciba un parámetro en el

Cuando TypeScript detecta el modificador de visibilidad "public" o "private" en el parámetro enviado al constructor, inmediatamente declara una propiedad en la clase y le asigna el valor recibido en el constructor. Por tanto, esta declaración:

```
export class ListadoClientesComponent {  
  constructor(public clientesService: ClientesService) { }  
}
```

Sería equivalente a escribir todo el código siguiente:

```
export class ListadoClientesComponent {  
  clientesService: ClientesService;  
  constructor(clientesService: ClientesService) {  
    this.clientesService = clientesService;  
  }  
}
```

En resumen, TypeScript entiende que, si defines la visibilidad de un parámetro en el constructor, o que quieres hacer en realidad es crear una propiedad en el objeto recién construido, con el valor recibido por parámetro.

## Usando el servicio en el componente

Ahora, para acabar esta introducción a los servicios en Angular, tenemos que ver cómo usaríamos este servicio en el componente. No nos vamos a detener demasiado en hacer ejemplos elaborados, que podemos abordar más adelante, solo veremos un par de muestras sobre cómo usar la propiedad declarada en el servicio.

### 1.- Usando el servicio dentro de la clase del componente

Dentro de la clase de nuestro componente, tendremos el servicio a partir de la propiedad usada en su declaración. En el constructor dijimos que el servicio se llamaba "clientesService", con la primera en minúscula por ser un objeto.

Pues como cualquier otra propiedad, accederemos a ella mediante la variable "this".

```
export class ListadoClientesComponent implements OnInit {  
  
  constructor(public clientesService: ClientesService) { }  
  
  ngOnInit() {  
    console.log(this.clientesService);  
  }  
  
}
```

El ejemplo no vale para mucho, solo para mostrar que, desde que esté creado el objeto podemos acceder al servicio con "this.clientesService".

También nos sirve para recordar que, el primer sitio donde podríamos usar los servicios declarados es en el método `ngOnInit()`. Dicho de otro modo, si necesitamos de estos servicios para inicializar propiedades en el componente, el lugar donde ponerlos en marcha sería el `ngOnInit()`.

## 2.- Usando el servicio en el template de un componente

Por su parte, en el template de un componente, podrás también acceder al servicio, para mostrar sus propiedades o incluso invocar sus métodos como respuesta a un evento, por ejemplo.

Como el servicio está en una propiedad del componente, podremos acceder a él mediante ese nombre de propiedad.

```
<p>  
  URL De acceso: {{clientesService.accesoFacturacion}}  
</p>
```

## Conclusión a la introducción a los servicios en Angular

Eso es todo lo que tienes que saber para comenzar a experimentar con los servicios en Angular. Es momento que pongas las manos en el código y comiences a experimentar por tu cuenta, creando tus propios servicios y usándolos desde tus componentes.

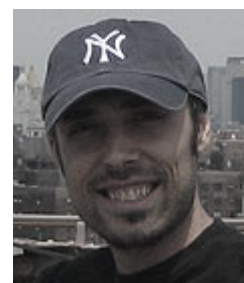
En artículos posteriores veremos servicios un poco más completos, que hagan más cosas que el que hemos visto en la presente entrega.

### Autor

[Miguel Angel Alvarez](#)

Seguir a [@midesweb](#)

Miguel es fundador de [DesarrolloWeb.com](#) y la plataforma de formación online [EscuelaIT](#). Comenzó en el mundo del desarrollo web en el año 1997, transformando su hobby en su trabajo.



Subir ↑

## Manual

[Manual de Angular](#)

← Emisión de eventos personalizados con `@Output` en Angular

Usar clases e Interfaces en los servicios Angular →

## Compartir

<https://desarrolloweb.com/articulos/servicios-angular.html>

## Comentarios

 [Enviar un comentario al artículo](#)

**Servando****Servicios en Angular**

17/12/2017

Me alegro mucho que hayáis comenzado ya a abordar este asunto en vuestro manual de Angular. Estaba echando en falta!

[Marcar como spam](#)


**Juan Carlos****Muy buena explicación**

14/12/2018

Había visto video del tema y no me quedaba claro, pero con tu magnífica explicación pude entender muy fácil lo que es un servicio. Excelente trabajo y muchas gracias. Saludos.

[Marcar como spam](#)

 [Enviar un comentario al artículo](#)

 Usuarios: [Login](#) | [Registro](#)

**Principales**[Manuales](#)[FAQs](#)[En directo](#)[Vídeos](#)**Monotemáticos**[Desde cero](#)[HTML, CSS](#)[Javascript, Ajax](#)[Diseño, ASP](#)



[Desarrolloweb.com](#) [Copyright](#) [Publicidad](#) [Acerca de](#) [Datos legales](#) [P. de cookies](#) [Contacta](#)