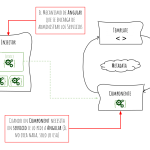
 agregar Servicios

en Angular



GH [Gustavo Hernán Dohara](https://gustavodohara.com/blogangular/)

**De Componentes solamente no vive la gente, también necesita Servicios. Se necesita obtener datos de alguna fuente o compartir datos entre Componentes o alguna lógica . El Componente ya tiene mucho que hacer, preparando los datos para que su Template los muestre en la pantalla, como para preocuparse de conseguir esos datos. Es por eso que los Servicios (o Services) entran en el mundo de Angular**

## ¿Qué es un Servicio?

Es una ****Clase**** con un propósito especifico.

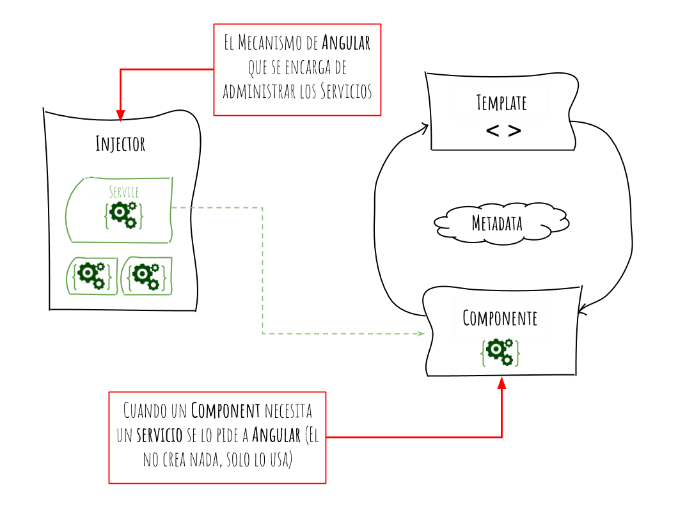
Los ****Servicios**** no dependen de ningún Componente (o mejor dicho no deberían hacerlo, así que varios ****Componentes**** deberían poder usar el mismo ****Servicio**** sin problemas.

Los srevicios ofrecen la posibilidad de encapsular las interacciones con el mundo exterior (por ejemplo servicios ****REST****) o de compartir datos en toda tu aplicación.

También pueden ofrecer una lógica particular que no dependa de ningún ****Componente.****

Como los servicios están pensados para ser usados por varios ****Componentes, Angular**** ofrece un mecanismo para facilitar el uso de estos ****Servicios****.

Lo que hacemos es definir los ****Servicios**** en los Módulos de ****Angular****, éste se encargara de crearlo cuando lo considere conveniente, y luego cuando necesitemos el ****Servicio,**** se lo pedimos a ****Angular.****



## ¿Para qué se usan los servicios?

Generalmente se usan para obtener los datos del mundo exterior o compartir datos en toda la aplicación, que luego se mostrarán en una página.

Por ejemplo, si tenés una página que muestra el clima de tu ciudad, y ese dato te lo traés de otro lado (un servicio ****REST****, por ejemplo). Tu página no quiere saber ****cómo**** conseguir los datos, sólo quiere ****usarlos****, de esta forma el ****Componente**** sólo sabe cómo mostrar los datos, el ****Servicio**** sólo sabe cómo conseguir esos datos, y ninguno se mete con el trabajo del otro.

Otro uso común de los ****Servicios**** es el de implementar una lógica en particular que no sea especifica de un ****Componente.**** Por ejemplo, si quisieras convertir de grados Centígrados a Farenheit, podrías tener un ****Servicio**** que haga esa lógica, y varios ****Componentes**** usarían el mismo servicio.

Se podría decir que los ****Servicios**** están para hacerle la vida mas fácil a los ****Componentes**** (y sus ****Templates)****.

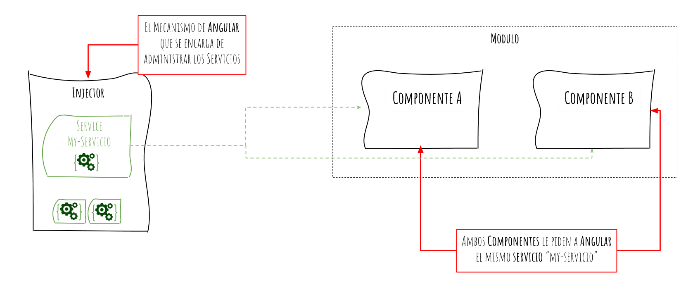
Los ****Componentes**** solo deberían saber cómo mostrar datos en su ****Template,**** ignorando cómo conseguir esos datos a mostrar.

Los ****Servicios**** sólo deberían saber cómo conseguir o procesar datos, ignorando quién los va a usar. Esto hace que los ****Componentes**** sean mucho más fáciles de leer y de testear. Y todos felices y contentos.

## Los servicios son Singleton

Singleton… ¿eso qué significa? Lo que significa es que en toda tu aplicación ****Angular**** existe un tipo de ****Servicio****, uno y sólo uno. A diferencia de cuando creás clases, que podés crear cuantas quieras, con los ****Servicios**** de ****Angular****, sólo una instancia del ****Servicio**** puede existir al mismo tiempo en tu aplicación.

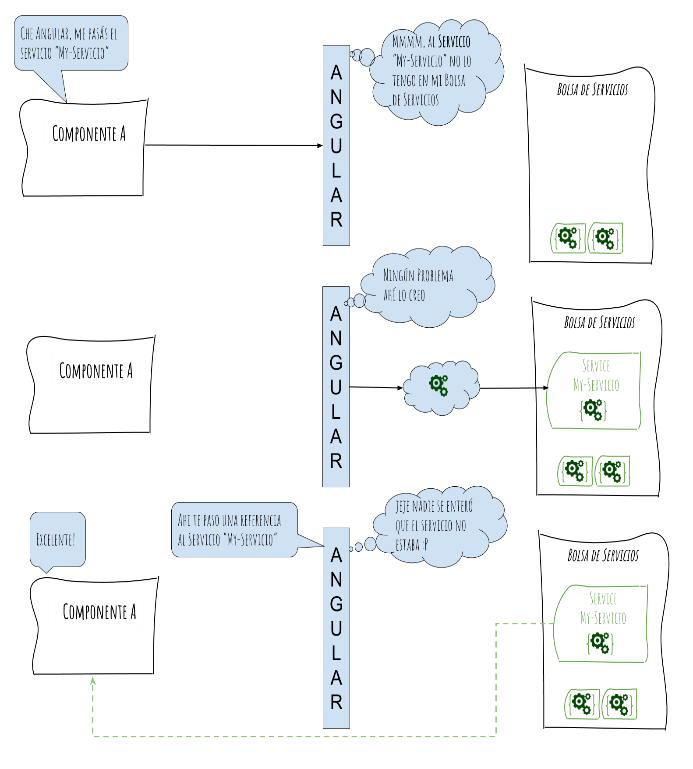
Así que mucho cuidado cuando guardás datos en un servicio y éste es usado por más de un ****Componente.**** Para remediar este problema, hay que usar ****Servicios**** que no guarden estado (que no guarden datos internos), o si los ****Servicios**** guardan estados, que los ****Componentes**** usen ese ****Servicio**** sabiendo que el estado puede modificarse. Otra cosa que se puede hacer es crear un ****Servicio**** nuevo a nivel del ****Componente**** (esto lo explicamos mas adelante)



## Explicación de cómo se crea y usa un Servicio

Los ****Servicios,**** al igual que los ****Component,****  son clases; la diferencia es que tienen la ****Metadata @Injectable()****.

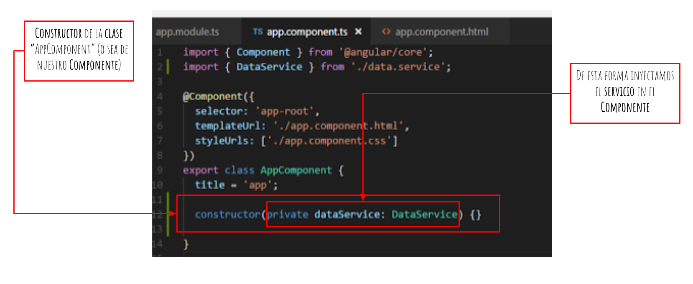
¿Qué significa esto de ****Injectable****? Significa (resumiendo ya que lo voy a explicar en otro post) que cualquier ****Componente****, ****Servicio****, ****Directiva**** o ****Pipe**** puede usar este «****Injectable****» sin preocuparse de crearlo o destruirlo, y que ****Angular**** se encarga de «inyectarle» el ****Servicio**** para ser usado por quien lo necesite. El que quiera usar el servicio lo usa y listo, ahorrándose muchos dolores de cabeza a la hora de crearlo o destruirlo, ya que ****Angular**** se encarga de todo eso.



 Lo único que hay que hacer es indicar que un ****Servicio**** es un «servicio» con la metadata ****@Injectable()****  registrándolo en ****Angular,**** y listo, ¡a usar el ****Servicio****!

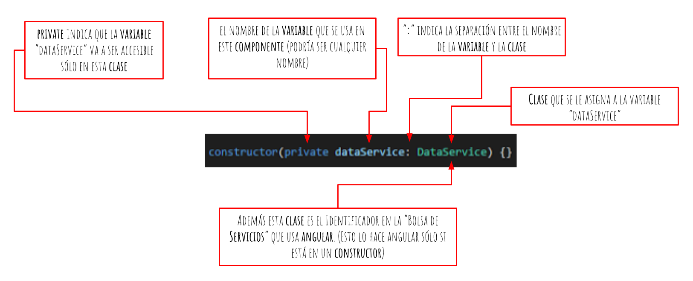


¡Bien! ya tenemos toda la parte de creación y registración del servicio en ****Angular****, ahora hay que usarlo. Acá viene la «magia» de ****Angular**** ya que lo único que tenés que hacer es inyectar el ****Servicio**** en el ****constructor**** de tu ****Componente**** (o sea el primer método que se ejecuta cuando se crea el ****Component****), de la siguiente forma:



### ¡Un momento! ¿cómo sabe Angular qué Servicio ir a buscar en esa «bolsa de Servicios»?

¡Muy buena pregunta! Lo hace usando el nombre de la clase, en nuestro ejemplo es «DataService» (ojo que dije el nombre de la clase y no el identificador que está en el componente). Analicemos un poco mas el código:

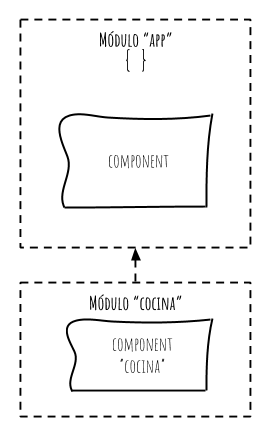


O sea, lo que hacemos en una sola línea de código es:

* Crear una ****variable**** «privada» para el ****Componente****
* Asignarle el tipo DataService
* INYECTARLE el ****Servicio**** DataService de la «Bolsa de ****Servicios****«

## ¡Ahora sí, a codear! seguimos con la casa inteligente

Para ver cómo hicimos esta casa visiten estos otros post ( [post 1](https://gustavodohara.com/blogangular/como-crear-tu-primera-aplicacion-angular-en-una-hora/), [post 2](https://gustavodohara.com/blogangular/como-agregar-un-modulo-angular-en-tu-aplicacion/), [post 3](https://gustavodohara.com/blogangular/agregar-componente-una-pagina-modulo-angular/), [post 4](https://gustavodohara.com/blogangular/navegar-las-paginas-aplicacion-angular-mucho-mas/)). Esto es lo que tenemos hasta ahora:

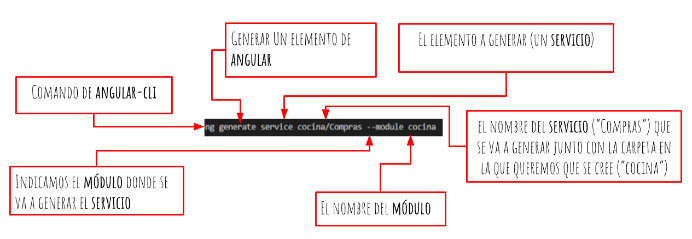


Lo que vamos a crear ahora es un ****Servicio**** que esté en el ****módulo**** «Cocina».

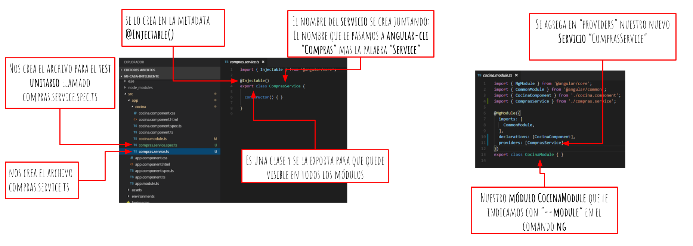
Este nuevo ****Servicio**** nos va a devolver una lista de elementos que hay que comprar (muy original :D).

El servicio se llamará CompraService y tendrá un método llamado «listaDeCompras()» y devolverá la lista de compras.

Para crearlo usamos angular-cli con el siguiente comando:



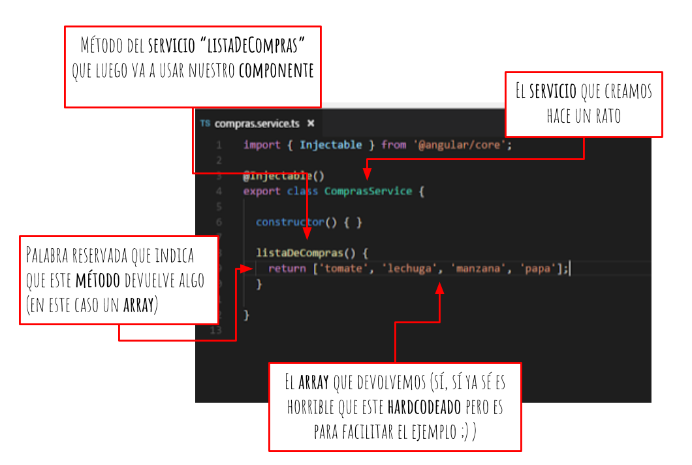
El comando nos genera lo siguiente:



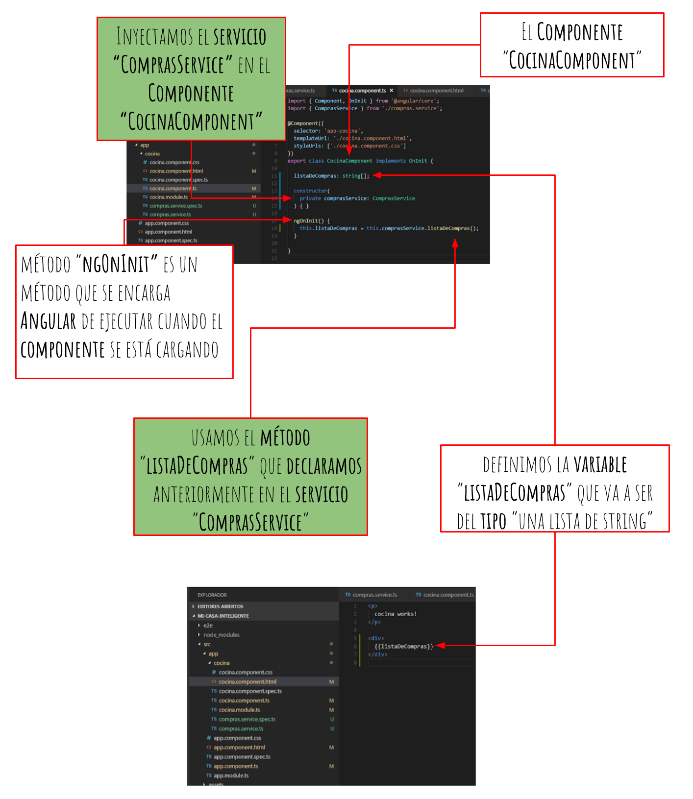
Como se ve, el ****Servicio**** se está anotando con el ****Metadato**** «****@Injectable()****«.

Al indicarle el ****módulo**** a la hora de crear el ****Servicio**** en ****angular-cli,****nos registra el ****Servicio**** automáticamente en nuestro ****módulo**** cocina. Como mencionamos antes, el ****Servicio**** va a ser el mismo para todos los ****Componentes**** de ese ****módulo**** (y dicho sea de paso, para cualquier otro ****módulo**** que se importe el****«CocinaModule»****).

Y ahora le agregamos el método «listaDeCompras()»



Por último, en el ****Component**** «cocina» vamos a mostrar esa lista en pantalla:



Y ésto nos muestra el browser:

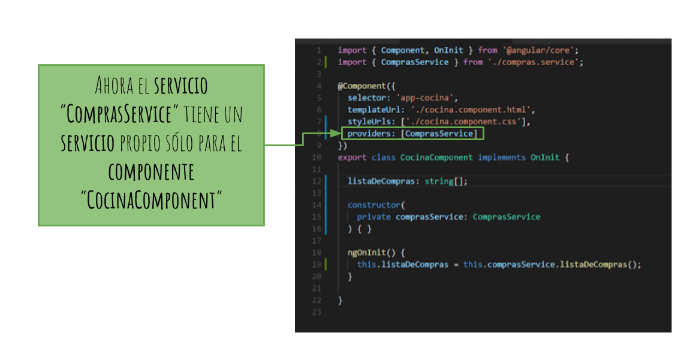


Y ya tenemos nuestra cocina un pelín mas completa.

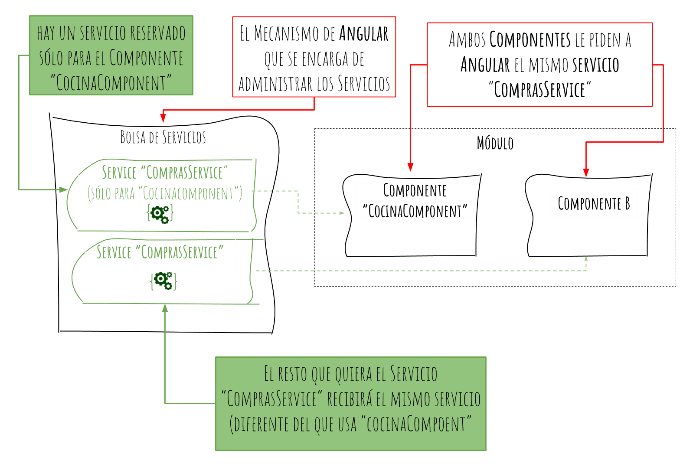
## BONUS TRACK

### ¡Un momento! y si NO quiero que un Servicio sea SINGLETON,¿cómo hago?

Ningún problema, como te había adelantado, los muchachos de ****Angular**** también pensaron en eso, si querés que un ****Component**** tenga un ****Servicio**** para él solito, lo único que tenés que hacer es definirlo en «****providers****» cuando se define la ****Metadata**** en el ****Component****:



De esta forma, el ****Component****tiene su instancia del ****Servicio**** para él solo, y puede usarla sin tener que preocuparse de que otra parte de tu aplicación use ese mismo ****Servicio****



## ERRORES COMUNES

Si por casualidad se te ocurre inyectar un ****Component**** como si fuera un ****Servicio,**** -o sea registrarlo en  ****«providers»-**** y después inyectarlo en un constructor, lamento decirte que eso no te va a funcionar :(. Aunque la aplicación va a levantar sin errores, cuando quieras usar el ****Component****te va a explotar por los aires, así que ya sabés, nada de ****Component**** en el array de «****providers****«.

## Conclusión

Como ya viste el «scope» de tu ****Servicio Singleton**** es muy importante, puede pasar de comportarse  como vos querés a hacer cualquier cosa, con sólo registrar tu Servicio en el lugar indicado. Así que prestá mucha atención y todo irá diez puntos.