**Inyección de Dependencias (*Dependency Injection* o DI)**

Es un patrón de diseño orientado a objetos en el que se suministran objetos a una clase, en lugar de ser la propia clase quien cree el objeto. El término fue acuñado por primera vez por [Martin Fowler](http://martinfowler.com/intro.html#_blank), y se considera parte del paradigma de **Programación Orientada a Aspectos** (AOP).

En programación, el primer paso para la organización del software lo constituyó la creación de bibliotecas de componentes reutilizables. No obstante, con ellas, el flujo se complica, saltando de componente a componente, y aparece un nuevo problema: la dependencia (acoplamiento) entre los componentes (basta con recordar el famoso ["infierno de las DLL's"](http://es.wikipedia.org/wiki/DLL_Hell#_blank) vinculado con la tecnología COM).

Para solucionarlo, se comienzan a crear nuevos patrones de diseño que puedan permitir reducir esta dependencia. Uno de los principales, es el conocido como Inversión de Control (**IoC**, o *Inversion of Control*, en inglés). De hecho, **Fowler** aplica estas ideas para crear la Inyección de Dependencias, que no es sino otra forma de IoC.

**Forma de implementación**



Según puede [verse en Wikipedia](http://es.wikipedia.org/wiki/Inyecci%C3%B3n_de_dependencias), la forma de implementarlo, es mediante un "Contenedor DI", que maneja las clases implicadas, de forma que es capaz de determinar sus relaciones **y sus dependencias**, suministrándoselas a cada clase según la **Ley de Deméter**, o **Principio de Menos Conocimiento** , que no es sino un caso específico de *Loose Coupling* o Acoplamiento Débil, donde, por resumirlo se deben dar los siguientes supuestos:

* Cada unidad debe tener un limitado conocimiento sobre otras unidades y solo conocer aquellas unidades estrechamente relacionadas a la unidad actual.
* Cada unidad debe *hablar* solo a sus amigos y no *hablar* con extraños
* Ante cualquier duda, solo *hablar* con los amigos inmediatos

Para una explicación más detallada de esta ley, puede verse " [Law of Demeter](http://en.wikipedia.org/wiki/Law_of_Demeter)" (inglés), en Wikipedia.

En nuestro caso, **AngularJS actuará como contenedor** y suministrará a nuestros recursos de programación los objetos necesarios para su labor sin necesidad de preocupaciones extra por parte del programador, así que, no te preocupes porque la implementación de esta característica es más fácil de usar que de comprender teóricamente.

Pero, básicamente, se trata de una capa más que va a eliminarnos los problemas de acoplamiento de dependencias típicos de cualquier desarrollo donde los componentes son parte fundamental.