# What is dependency injection?

Dependency injection (DI) in object-oriented computer programming is a technique that indicates to a part of a program which other parts it can use, i.e. to supply an external dependency (i.e. a reference) to a software component. In technical terms, it is a design pattern that separates behavior from dependency resolution, thus decoupling highly dependent components.  
  
Developers of software strive to reduce dependencies between components in software for various reasons. This leads to a new problem, though: How can a component know all the other components it needs to fulfill its purpose?  
  
The traditional approach was to hard-code the dependency. As soon as the database driver was necessary, the component would execute a piece of code that would load a specific driver, configure it and call the necessary methods to interact with the database. If a second database must be supported, this piece of code would have to be modified or, even worse, copied and modified (violating the DRY principle).  
  
Dependency injection offers a solution. Instead of hard-coding the dependencies, a component just lists the necessary services and a DI framework supplies these. At runtime, an independent component will load and configure the database driver and offer a standard interface to interact with the database. Again, the details have been moved from the original component to a set of new, small, database specific components, reducing the complexity of them all.  
  
In DI terms, these new components are called "service components" because they render a service (database access) for one or more other components.  
Dependency injection is a specific form of inversion of control where the concern being inverted is the process of obtaining the needed dependency. The term was first coined by Martin Fowler to describe the mechanism more clearly.