# Intro into dependency injection

## What is dependency injection

In my own words, Dependency injection is a software technique/pattern that supplies an object with every other object it needs (depends on) to execute its own logic.

This helps with enforcing the “Single-responsibility principle” of SOLID and if used in conjunction with Interfaces, it also helps with enforcing the “Dependency inversion principle” of SOLID

It can also help you keep your code DRY as you reduce the chance of duplicating the logic to instantiate the dependency objects.

## A real-life analogy

Let’s say we are the owners of a construction company and we have a multitude of carpenters working for us. If we rely on the carpenters to bring/create their own drills we might get varying results when they all follow the same procedure for the same unit of work. To elevate these inconsistencies the company can give them all the same type of drill. If an issue arises that stems from a fault in the drills design, the company can swap out all the drills and the next day the carpenters just pick up the new drills and continue what they do best, carpentry, not buying and maintaining drills.

## What practical advantages does DI give us

* Easier to maintain code because the creation of the same object is not spread out over multiple other objects
* Easier to swap out implementations because of the above statement
* Easier to unit test because of the above statement

## IOC containers and what they bring to DI

Inversion of Control containers are an optional tool that can be used by a framework (mvc, umbraco, …) to grab the implementation of a dependency and supply it to the objects defined by your custom code (controllers). Both the framework and the application that uses the framework can add implementations to the container to satisfy all dependencies in the application.

# Dependency injection and Umbraco 8

## Registering implementations into the Umbraco Container

Usually registration into the IOC container is done somewhere in your app startup, however. Because we typically want to register things after Umbraco has registered its own implementations, Umbraco supplies the ability to trigger a registration in a component. So the steps we need to take are

* Create a component
* Register our implementations in said component

## Requesting dependencies in Umbraco controllers

So, how do we request our dependencies in a controller? The easiest way is to use constructor injection, by defining the type and variable in the constructor. Umbraco will take it from there.

## Example

* Show working surface controller with a mail client
* Show changed solution with mail client injected into the controller
* Show changed solution with a different mail client registration