Spring / Spring Boot Master Notes

# Dependency Injection / Inversion of Control

Best explanation taken from [here](https://stackoverflow.com/questions/3058/what-is-inversion-of-control).

The **Inversion-of-Control (IoC)** pattern, is about providing *any kind* of callback, which "implements" and/or controls reaction, instead of acting ourselves directly (in other words, inversion and/or redirecting control to the external handler/controller). The **Dependency-Injection (DI)** pattern is a more specific version of IoC pattern, and is all about removing dependencies from your code.

Every DI implementation can be considered IoC, but one should not call it IoC, because implementing Dependency-Injection is harder than callback (Don't lower your product's worth by using the general term "IoC" instead).

For DI example, say your application has a text-editor component, and you want to provide spell checking. Your standard code would look something like this:

What we've done here creates a dependency between the TextEditor and the SpellChecker. In an IoC scenario we would instead do something like this:

public class TextEditor {

private SpellChecker checker;

public TextEditor() { this.checker = new SpellChecker(); }

}

In the first code example we are instantiating SpellChecker (this.checker = new SpellChecker();), which means the TextEditor class directly depends on the SpellChecker class.

public class TextEditor {

private IocSpellChecker checker;

public TextEditor(IocSpellChecker checker) { this.checker = checker; }

}

In the second code example we are creating an abstraction by having the SpellChecker dependency class in TextEditor's constructor signature (not initializing dependency in class). This allows us to call the dependency then pass it to the TextEditor class like so:

SpellChecker sc = new SpellChecker(); // dependency

TextEditor textEditor = new TextEditor(sc);

Now the client creating the TextEditor class has control over which SpellChecker implementation to use because we're injecting the dependency into the TextEditor signature.

**Note** that just like IoC being the base of many other patterns, above sample is only one of many Dependency-Injection kinds, for example:

* Constructor Injection.

Where an instance of IocSpellChecker would be passed to constructor, either automatically or similar to above manually.

* Setter Injection.

Where an instance of IocSpellChecker would be passed through setter-method or public property.

* Service-lookup and/or Service-locator

Where TextEditor would ask a known provider for a globally-used-instance (service) of IocSpellChecker type (and that maybe without storing said instance, and instead, asking the provider again and again).