The @Configuration annotation indicates to Spring that this is a configuration class that will provide beans to the Spring application context.

The configuration’s class methods are annotated with @Bean, indicating that the objects they return should be added as beans in the application context (where, by default, their respective bean IDs will

be the same as the names of the methods that define them).

At its core, IoC, and therefore DI, aims to offer a simpler mechanism for provisioning **component dependencies (often referred to as an object’s collaborators)** and managing these dependencies throughout

their life cycles.

**A component that requires certain dependencies is often referred to as the dependent object or, in the case of IoC, the target**. In general, IoC can be decomposed into two subtypes: dependency injection and dependency lookup.

With dependency lookup–style IoC, a component must acquire a reference to a dependency, whereas with dependency injection, the dependencies are injected into the component by the IoC container. **Dependency lookup comes in two types:**

**dependency pull and contextualized dependency lookup (CDL).**

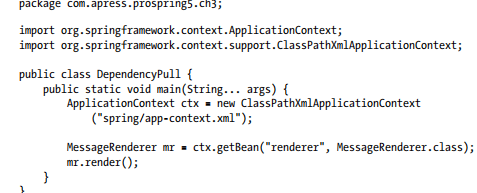
**Dependency injection also has two common flavors:**

**constructor and setter dependency injection.**

Dependency Pull :

In dependency pull, dependencies are pulled from a registry as required.

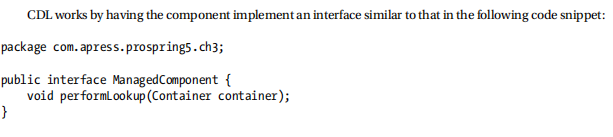
Spring also offers dependency pull as a mechanism for retrieving the components that the framework manages;



Contextualized Dependency Lookup ( CDL ).

in CDL, lookup is performed against the container that is managing the resource, not from some central registry, and it is usually performed at some set point.

using CDL requires your classes to implement a specific interface and look up all dependencies manually.



Constructor Dependency Injection:

An obvious consequence of using constructor injection is that an object cannot be created without its

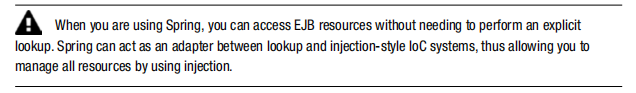
dependencies; thus, they are mandatory.

Injection vs. Lookup

Choosing which style of IoC to use—injection or lookup—is not usually a difficult decision. In many cases, **the type of IoC you use is mandated by the container** you are using.

For instance, if you are using EJB 2.1 or prior versions, you must use lookup-style IoC (via JNDI) to obtain an EJB from the JEE container.

**In Spring, aside from initial bean lookups, your components and their dependencies are always wired together using injection-style IoC.**



The real question is this: given the choice, which method should you use, injection or lookup?

The answer is most definitely injection. If you look at the code in the previous code samples, you can clearly see that using injection has zero impact on your components’ code. The dependency pull code, on the other hand, must actively obtain a reference to the registry and interact with it to obtain the dependencies, and using CDL requires your classes to implement a specific interface and look up all dependencies manually.

When you are using injection, the most your classes have to do is allow dependencies to be injected by using either constructors or setters.