Spring Dependency Injection

Dependency Injection

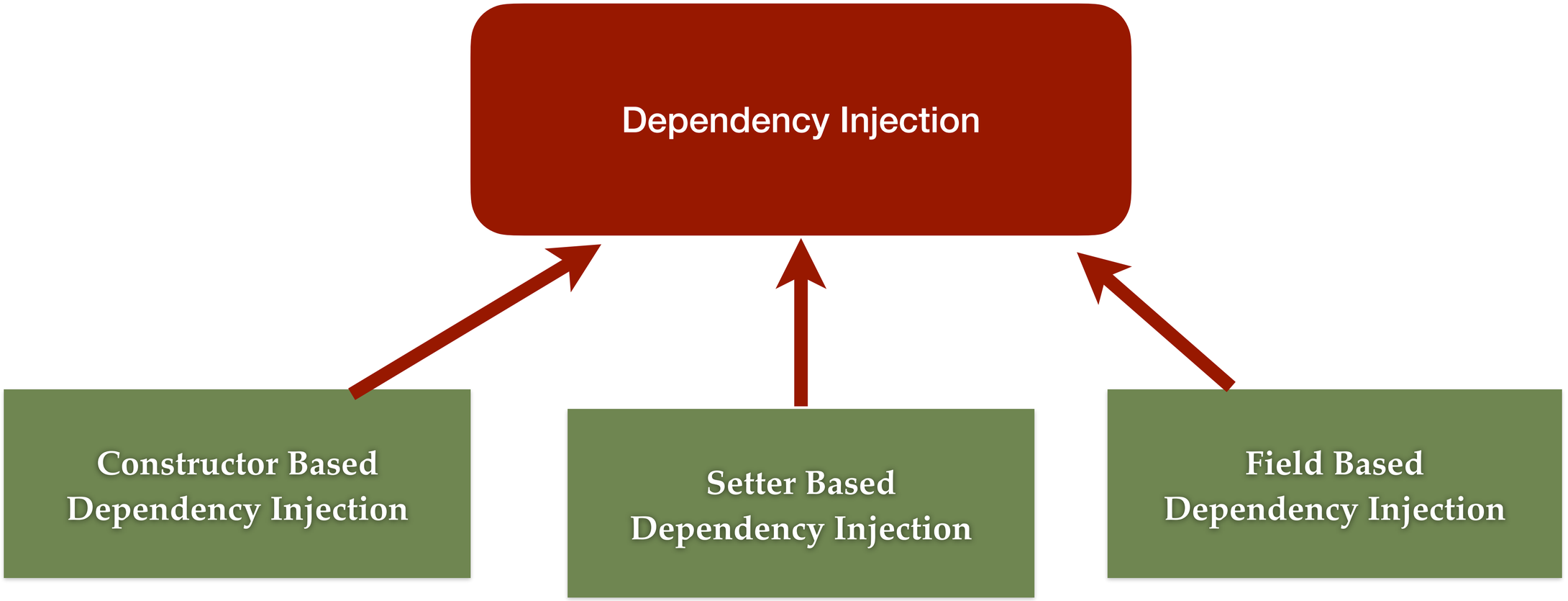
1. Dependency Injection is a design pattern on which dependency of the object is injected by the framework rather than created by Object itself - It is also called IOC (**Inversion of Control**)
2. Dependency Injection reduces coupling between multiple objects as its dynamically injected by the framework.
3. Spring IoC Container uses DI to inject one object into another object.
4. There are mainly three types of Dependency Injection: **Constructor Injection**, **Setter Injection** and **Field Injection**.

Dependency Injection

1. **Dependency:** An object usually requires objects of other classes to perform its operations. We call these objects dependencies.

2. **Injection:** The process of providing the required dependencies to an object.

Dependency Injection Types



Constructor Injection

- Constructor injection uses the constructor to inject dependency on any Spring-managed bean.

- Before Spring 4.3, we had to add an **@Autowired** annotation to the constructor. With newer versions, this is optional if the class has only one constructor.

- When we have a class with multiple constructors, we need to explicitly add the @Autowired annotation to any one of the constructors so that Spring knows which constructor to use to inject the dependencies.

Why Should We Use Constructor Injection?

Here are advantages of using constructor injection:

1. All Required Dependencies Are Available at Initialization Time
2. Identifying Code Smells
3. Preventing Errors in Tests
4. Immutability

Setter Injection

- Setter injection uses the setter method to inject dependency on any Spring-managed bean.
- We have to annotate the setter method with the **@Autowired** annotation.
- Spring will find the @Autowired annotation and call the setter to inject the dependency.

Field Injection

- As the name says, the dependency is injected directly in the field, with no constructor or setter needed. This is done by annotating the class member with the **@Autowired** annotation.

- Spring container uses reflection to inject the dependencies, which is costlier than constructor-based or setter-based injection.

Field Injection Drawbacks

- You cannot create immutable objects, as you can with constructor injection (you can't make field **final**).
- Your classes have tight coupling with Spring IoC container and cannot be used outside of it.
- Your classes cannot be instantiated (for example in unit tests) without reflection. You need the Spring IoC container to instantiate them, which makes your tests more like integration tests.
- Having too many dependencies is a red flag that the class usually does more than one thing, and that it may violate the Single Responsibility Principle.

When to Use Constructor-based and Setter-based DI in Spring?

- Use constructor-based DI for **mandatory dependencies** so that your bean is ready to use when it is first time called.
- Use setter-based DI only for **optional dependencies**.
- Use setter injection to avoid **circular dependencies**

Which one is recommended

- Spring team recommended to use **constructor based-dependency** injection.

  Here are advantages of using constructor injection:

  1. All required dependencies are available at initialization time (this reduces the code as well)
  2. Immutability and avoid NullPointerException
  3. Preventing errors in Tests