WHAT IS IOC AND DEPENDENCY INJECTION?

These are the design patterns that are used to remove dependency from the programming code. They make the code easier to test and maintain. Let's understand this with the following code:

**class** Employee{

Address address;

Employee(){

address=**new** Address();

}

}

In such case, there is dependency between the Employee and Address (tight coupling). In the Inversion of Control scenario, we do this something like this:

DEPENDENCY INJECTION: when two classes are tightly coupled then don’t create instance of child class inside parent instead the IOC container will take care of object creation and do setting via construtor injection or setter- getter injection

**class** Employee{

Address address;

Employee(Address address){

**this**.address=address;

}

}

Thus, IOC makes the code loosely coupled. In such case, there is no need to modify the code if our logic is moved to new environment.

In Spring framework, IOC container is responsible to inject the dependency. We provide metadata to the IOC container either by XML file or annotation.

#### Advantage of Dependency Injection

makes the code loosely coupled so easy to maintain

makes the code easy to test.

Class A{

B b=new B();// using construtor

}

Class A{

B b;

}

1. what is @Component, @Service, @Repository, @Transaction?

@Component: generic stereotype for any Spring-managed component

@Service: stereotype for service layer

@Repository: stereotype for persistence layer.The @Repository annotation is a marker for any class that fulfils the role or stereotype of a repository (also known as Data Access Object or DAO).

@Transaction: Spring **dynamically creates a proxy that implements the same interface(s) as the class you're annotating**. And when clients make calls into your object, the calls are intercepted and the behaviors injected via the proxy mechanism.

1. difference between @Component and @Bean?

| **Sr. No.** | **Key** | **@Bean** | **@Component** |
| --- | --- | --- | --- |
| 1 | Auto detection | It is used to explicitly declare a single bean, rather than letting Spring do it automatically. | If any class is annotated with @Component it will be automatically detect by using classpath scan. |
| 2 | Spring Container | Bean can be created even class is outside the spring container | We can’t create bean if class is outside spring container |
| 3 | Class/Method  Level Annotation | It is a method level annotation | It is a class level annotation |
| 4 | @Configuration | It works only when class is also annotated with @Configuration | It works without @Configuration annotation |
| 5 | Use Case | We should use @bean, if you want specific implementation based on dynamic condition. | We can’t write specific implementation based on dynamic condition |

### **What is spring boot and what is @SpringBootApplication?**

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need minimal Spring configuration.

If you’re looking for information about a specific version.

Spring Boot @springbootApplication annotation is used to mark a configuration class that declares one or more @bean  methods and also triggers auto-configuration and component scanning. It’s same as declaring a class with @Configuration, @EnableAutoConfiguration and @ComponentScan annotations.

1. how mamy types of autowire and what is default?

****no****: It’s the default autowiring mode. It means no autowiring.

****byName****: The byName mode injects the object dependency according to name of the bean. In such a case, the property and bean name should be the same. It internally calls the setter method.

**byType**: The byType mode injects the object dependency according to type. So it can have a different property and bean name. It internally calls the setter method.

****constructor****: The constructor mode injects the dependency by calling the constructor of the class. It calls the constructor having a large number of parameters.

****autodetect****: In this mode, Spring first tries to autowire by the constructor. If this fails, it tries to autowire by using byType.

bytype is default

6. what is qualifier used for?

The @Qualifier annotation is used to resolve the autowiring conflict, when there are multiple beans of same type. The @Qualifier annotation can be used on any class annotated with @Component or on method annotated with @Bean . This annotation can also be applied on constructor arguments or method parameters.

7. what is scope of bean, what is default scope?

Scopes a single bean definition **to any number of object instances**. Scopes a single bean definition to the lifecycle of a single HTTP request; that is each and every HTTP request will have its own instance of a bean created off the back of a single bean definition.

**Singleton** is the default scope for a Bean, the one that will be used if nothing else is indicated. This scope implies that Spring container will create an only shared instance of the class designated by this bean, so each time the Bean is required the same object will be injected.

1. what is circular dependency issue in spring how to avoid it?

Circular dependency in Spring happens when two or more beans require instance of each other through constructor dependency injections.

For example: There is a ClassA that requires an instance of ClassB through constructor injection and ClassB requires an instance of class A through constructor injection.

In that sort of configuration where beans for both classes need to be injected into each other, Spring container can't decide which bean should be created first. The Spring IoC container will detect this circular reference at runtime while trying to inject dependencies and throw a BeanCurrentlyInCreationException.

1. what is difference between application context and beanfactory and how many types of application context?

**BeanFactory :**

Does not support the Annotation based dependency Injection.

Doesn't Support I18N.

By default its support Lazy loading.

it doesn't allow configure to multiple configuration files.

ex: BeanFactory context=new XmlBeanFactory(new Resource("applicationContext.xml"));

**ApplicationContext:**

Support Annotation based dependency Injection.-@Autowired, @PreDestroy

Support I18N

Its By default support Aggresive loading.

It allow to configure multiple configuration files.

ex:ApplicationContext context=new ClasspathXmlApplicationContext("applicationContext.xml");

Types of ApplicationContext

1.. AnnotationConfigApplicationContext. First, let's see the AnnotationConfigApplicationContext class, which was introduced in Spring 3.0. ...

2. AnnotationConfigWebApplicationContext. ...

3. XmlWebApplicationContext. ...

4. FileSystemXMLApplicationContext. ...

5. ClassPathXmlApplicationContext.

10 . how to write constructor injection in spring?

We can inject the dependency by constructor. The <constructor-arg> subelement of <bean> is used for constructor injection. Here we are going to inject.

primitive and String-based values

Dependent object (contained object)

Collection values etc.

Injecting primitive and string-based values

Let's see the simple example to inject primitive and string-based values. We have created three files here:

Employee.java

applicationContext.xml

Test.java