**1. what is IOC, Dependency injection**

Inversion of Control (IoC) is a principle by which the control of objects is transferred to a container or framework. Dependency injection is a pattern through which IoC is implemented and the act of connecting objects with other objects or injecting objects into objects is done by container rather than by the object themselves.

**2. what is @Component, @Service, @Repository, @Transaction**

***@Component:*** Spring Component annotation is used to denote a class as Component that allows Spring to automatically detect our custom beans. In other words, without having to write any explicit code, Spring will Scan our application for classes annotated with @Component. Instantiate them and inject any specified dependencies into them.

***@Service:*** marks a Java class that performs some service, such as executing business logic, performing calculations, and calling external APIs. This annotation is a specialized form of the @Component annotation intended to be used in the service layer.

***@Repository:*** Spring @Repository annotation is used to indicate that the class provides the mechanism for storage, retrieval, search, update and delete operation on objects.

***@Transaction:*** The @Transactional annotation is the metadata that specifies the semantics of the transactions on a method. We have two ways to rollback a transaction: declarative and programmatic. The default rollback behavior in the declarative approach will rollback on runtime exceptions

**3. Difference between @Component and @Bean**

***@Component*** (and @Service and @Repository) are used to auto-detect and auto-configure beans using classpath scanning. There's an implicit one-to-one mapping between the annotated class and the bean (i.e. one bean per class). Control of wiring is quite limited with this approach, since it's purely declarative.

***@Bean*** is used to explicitly declare a single bean, rather than letting Spring do it automatically as above. It decouples the declaration of the bean from the class definition, and lets you create and configure beans exactly how you choose.

**4. What is spring boot and what is @SpringBootApplication**

Spring Boot is an open source Java-based framework used to create a Micro Service. It is developed by Pivotal Team. It is easy to create a stand-alone and production ready spring applications using Spring Boot.

**@SpringBootApplication:** to use auto-configuration, component scan and be able to define extra configuration on their "application class". A single @SpringBootApplication annotation can be used to enable those three features, that is:

* @EnableAutoConfiguration: enable [Spring Boot’s auto-configuration mechanism](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/using-boot-auto-configuration.html)
* @ComponentScan: enable @Component scan on the package where the application is located (see [the best practices](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/using-boot-structuring-your-code.html))
* @Configuration: allow to register extra beans in the context or import additional configuration classes

The @SpringBootApplication annotation is equivalent to using  @Configuration,  @EnableAutoConfiguration, and @ComponentScan with their default attributes

**5. How many types of autowire and what is default**

Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection. Autowiring can't be used to inject primitive and string values. It works with reference only. There are 5 types.

* **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.

**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 methods 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**

There are 5 types of spring bean scopes.

* **singleton** – only one instance of the spring bean will be created for the spring container. This is the default spring bean scope. While using this scope, make sure bean doesn’t have shared instance variables otherwise it might lead to data inconsistency issues.
* **prototype** – A new instance will be created every time the bean is requested from the spring container.
* **request** – This is same as prototype scope, however it’s meant to be used for web applications. A new instance of the bean will be created for each HTTP request.
* **session** – A new bean will be created for each HTTP session by the container.
* **global-session** – This is used to create global session beans for Portlet applications.

**8. what is cyclic 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. If you find that components are in a cycle with each other, there are three things you can do: Repackage them so they are no longer mutually dependent. Combine them into a single component.

**9. what is difference between application context and beanfactory and how many types of application context**

A BeanFactory pretty much just instantiates and configures beans. An ApplicationContext also does that, and it provides the supporting infrastructure to enable lots of enterprise-specific features such as transactions and AOP.

Types of application context

* AnnotationConfigApplicationContext container
* AnnotationConfigWebApplicationContext
* XmlWebApplicationContext

**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.