# Spring Core Interview Questions

java Spring Spring Boot Spring Core *by devs5003 - February 28, 2023*



Last Updated on January 4th, 2024

If you are preparing for a job interview in the field of Java development, it is essential to have a strong insight on Spring Core. Even being a Java developer, you must have Spring Boot knowledge to get shortlisted for a job interview. Furthermore, while working with a Spring Boot project, you must be using Spring Core Concepts. This article on 'Spring Core Interview Questions' will also benefit you as a Spring Core Concept refresher.


Spring is one of the most popular Java frameworks, widely used in developing enterprise-level applications. The Spring Core module is the foundation of the Spring Framework, providing essential features such as dependency injection and inversion of control. In this article, we'll cover some essential Spring Core interview questions that you must encounter during your job interview. By familiarizing yourself with these questions and their answers, you will be better prepared to explain your skills and knowledge during your interview.



## RECENT POSTS

Table of Contents

# Who can get benefit from these questions?

1) **Job seekers**: Spring Core has become a must have skill for a Java developer if you are going to work in a Spring/Spring Boot based Project. If you are looking for a job as a Java developer or software engineer, you must be asked about your knowledge of Spring Core and its related concepts during the interview process.

2) **Students and learners**: If you are studying Java or learning to code, understanding Spring Core concepts will help you build better applications, reduce time to write the code, and improve your development skills.

3) **Experienced developers**: Even if you have been working with Java for years, it is important to refresh your Spring Core concepts and stay up-to-date with the skills and best practices. Going through these Spring Core Interview Questions can help you become proficient and also in identifying areas where you may need to improve your knowledge or skills.

4) **Managers and recruiters**: If you are responsible for hiring Java developers, understanding Spring Core concepts can help you evaluate a candidate's knowledge and expertise. Asking Spring Core Interview Questions can also help you identify candidates who are up-to-date with the required skills and ready to work on Java development project.

In summary, Spring Core Interview Questions can benefit anyone who wants to improve their understanding of Spring Core concepts, whether they are job seekers, students, experienced developers, or managers and recruiters.

# Spring Core Interview Questions and Answers

### What is Spring Core?

Spring Core is a sub-module of Spring framework that provides a set of features and functionalities that helps in building enterprise applications using the Java programming language. It is the key module of the Spring Framework. We can't expect to develop a Spring based application without applying the concepts of Spring Core Module.

## What are the key features of Spring Core Container?

The Spring Core Container is the heart of the whole ~~Spring ecosystem~~ that primarily covers four components—core, beans, context, and expression language. Core and beans are responsible for providing the most fundamental features of the Spring framework, including Dependency Injection (DI) and Inversion Of Control (IoC). These modules are responsible for managing the IoC container, and the primary responsibilities are: bean instantiation, bean configuration, and destruction of the bean residing in the Spring container.

## What is Spring Dependency Injection?

Spring Dependency Injection (DI) is the feature of the core module of the Spring Framework, where the Spring Container injects other objects into an object. Here, other objects are nothing but dependencies (instance variables) which are declared in the class (Spring Bean). In Spring Core, dependency injection is achieved through the use of annotations and XML configurations.

## What is Inversion of Control (IoC)?

Inversion Of Control (IoC) is a concept of software engineering, where the control of objects or a block of a program is transferred to a container or framework. IoC allows a container to take control of the flow of a program and make calls to our custom code rather than making calls to a library. To implement IoC, the Spring Dependency Injection concept is used.

### What are the responsibilities of the Spring Context module?

The Context module inherits its features from the Beans module and includes support for internationalization (I18N), event-propagation, resource-loading, and the transparent creation of contexts. The Context module also contains support for some Java EE features like EJB, JMX and basic remoting support. The ApplicationContext interface (also called a Spring Container) is the key point of the Context module that provides these features.

### What are the responsibilities of the Spring Expression Language module?

The Expression Language module provides a powerful expression language for querying and manipulating an object graph at runtime. It can be considered as an extension of the unified expression language (unified EL) as specified in the JSP 2.1 specification. The language supports setting and getting of property values, property assignment, method invocation, accessing the context of arrays, collections and indexers, logical and arithmetic operators, named variables, and retrieval of objects by name from Spring's IoC container. It also supports list projection and selection, as well as common list aggregators.

### What is a bean in Spring Core?

A bean in Spring Core is a java class that is created and managed by the Spring container. It can be accessed and used by other classes/objects in the application.

### Can we declare any class or an interface as a Spring Bean?

We can consider any class as a Spring Bean, except an abstract class and an interface. It can also be a POJO class, a simple Java Bean class or any other class as well.

### What is the Spring Container?

The Spring container is the core component of the Spring framework that is responsible for creating, configuring, and managing spring beans in the application. It is also responsible for wiring dependencies between objects.

### What is the difference between the Spring container and other Java containers?

The Spring container is a specialized container that is designed to manage beans in a Spring application. It provides features such as bean creation, bean management, bean destruction, dependency injection, etc. On the other hand, other Java containers such as Servlet container, JSP container are generic containers that are used to manage respective objects in any Java application.

### What are the different types of Spring containers?

There are two types of Spring containers: the BeanFactory and the ApplicationContext. The BeanFactory is a lightweight container that provides the basic features of the Spring framework. The ApplicationContext is a more advanced container that provides additional features such as internationalization, message resolution, and application event handling etc.

### When to use the BeanFactory vs. ApplicationContext?

If you are using small scale, light weight spring based application, prefer using BeanFactory container as it takes less memory to work. In other words, if there is a memory limitation in place, prefer using BeanFactory container. For example, mobile apps, embedded system apps, IOT apps etc.

In all other heavy weight apps where memory limitation is not in place, such as web apps, enterprise apps, distributed apps, desktop apps etc., prefer using ApplicationContext container.

In general practice, we should prefer using ApplicationContext container wherever it is possible to use. You must have a genuine reason of not using ApplicationContext Container.

### What is a configuration class in Spring Core?

A configuration class in Spring Core is a Java class that is used to define the configuration for a Spring application. It is annotated with the @Configuration annotation and contains one or more @Bean methods that define the beans for the application.

### What is a bean scope in Spring Core?

A bean scope in Spring Core defines the scope or lifetime of a bean in the application. It controls the instance creation of the bean and it is managed by the spring container. Basically, it controls when the object of Bean will be instantiated, how long does that object live, and how many objects will be created for that bean throughout. There are different bean scopes available in Spring, such as singleton, prototype, request, session, and global session.

### What is a singleton bean scope?

A singleton bean scope is a bean scope in which there is only one instance of the bean in the application context. It is the default bean scope in Spring.

### What is a prototype bean scope?

A prototype bean scope is a bean scope in which a new instance of the bean is created every time it is requested.

### What is a request bean scope?

A request bean scope is a bean scope in which a new instance of the bean is created for every HTTP request.

### What is a session bean scope?

A session bean scope is a bean scope in which a new instance of the bean is created for every HTTP session.

### What is a global session bean scope?

A global session bean scope is a bean scope in which a new instance of the bean is created for every global HTTP session.

### What is bean wiring in Spring Framework?

Bean wiring in Spring refers to the process of creating dependencies between beans in the application context. In other words, it is a mechanism by which Spring container manages the relationships between different dependent objects in a Spring application. There are various approaches of Bean wiring such as XML configuration, Java-based configuration, and annotation-based configuration.

### What is constructor injection?

Constructor injection is a form of dependency injection in which dependencies are injected into the constructor of a class.

### What is setter injection?

Setter injection is a form of dependency injection in which dependencies are injected into the setter methods of a class.

### What is autowiring in Spring?

Autowiring in Spring is a mechanism for automatically wiring dependencies between beans based on their types. Spring Container detects the relationship between the beans either by reading the XML Configuration file or by scanning the Java annotations at the time of booting up the application. Further, it will create the objects & wire the dependencies. Since Spring Container does this process automatically, it is referred to as Autowiring i.e., Automatic Wiring.

### What are the different types of autowiring in Spring?

There are several autowiring modes in Spring, including no autowiring, by name, by type, and constructor.

### What is the difference between 'no autowiring' and 'by name autowiring'?

With no autowiring, dependencies are not automatically wired between beans. With by name autowiring, dependencies are automatically wired based on the names of the beans.

### What is the difference between 'by type autowiring' and 'constructor autowiring'?

With by type autowiring, dependencies are automatically wired based on their types. With constructor autowiring, dependencies are automatically wired through the constructor of the class.

### What is the @Autowired annotation?

The @Autowired annotation is used to automatically inject dependencies in Spring. It can be used on properties, setters, and constructors. When Spring encounters this annotation, it

looks for a bean of the same type and injects it into the class.

### What is the @Qualifier annotation?

The @Qualifier annotation is used to specify which bean to use for autowiring when there are multiple beans of the same type.

### What is the @Value annotation?

The @Value annotation is used to inject values into properties of a bean in Spring.

### What is the @Component annotation?

The @Component annotation is used to mark a class as a Spring component/Spring Bean. Moreover, the @Component annotation is used to indicate that a class should be managed by the Spring container same as a Spring bean. It is a generic stereotype annotation that can be used for any class.

### What is the difference between @Component, @Service, and @Repository annotations?

The @Component annotation is a generic stereotype annotation that can be used for any class. The @Service annotation is used to mark a class as a service component, while the @Repository annotation is used to mark a class as a data access component.

### What is the @Configuration annotation?

The @Configuration annotation is used to mark a class as a configuration class in Spring Core. It contains one or more @Bean methods that define the beans for the application.

### What is the @Bean annotation?

The @Bean annotation is used to declare a bean in Spring.

### What is the difference between @Bean and @Component annotations?

The @Component annotation is used to mark a class as a Spring component, while the @Bean annotation is used to declare a bean. The @Component annotation is used for classes that are not configured explicitly in the application context, while the @Bean annotation is used for classes that are configured explicitly in the application context.

### What is the purpose of the @Import annotation?

The @Import annotation is used to import one or more configuration classes into another configuration class.

### What is the purpose of the @PropertySource annotation?

The @PropertySource annotation is used to specify the location of a properties file that contains configuration properties for the application.

### What is the difference between a properties file and a YAML file in Spring?

Both properties files and YAML files are commonly used to configure Spring applications, but they have some differences in terms of syntax and features. Properties files use a simple key-value format to store configuration information, with each key and value separated by an equals sign (=). On the other hand, YAML uses whitespace indentation to indicate structure. YAML supports more complex data structures, making it a more flexible and expressive format for configuration. In general, properties files are simpler and easier to use for basic

configuration, while YAML is more powerful and better suited for more complex configuration scenarios.

### What is the purpose of the @Profile annotation?

The @Profile annotation is used in Spring to specify which beans should be loaded based on the active profiles. A profile is a way to define a set of configurations that are used in different environments or situations, such as development, testing, or production. The @Profile annotation can be applied to a class or a method, and its value specifies which profile or profiles the bean should be associated with.

### What is the difference between a bean definition and a bean instance in Spring?

A bean definition is a blueprint for a bean, while a bean instance is an actual object that is created from the bean definition.

### What is the purpose of the BeanFactory in Spring?

The BeanFactory is the core container interface in Spring Core. It is responsible for creating and managing various beans residing in the container.

### What is the purpose of the ApplicationContext in Spring?

The ApplicationContext is a more advanced container interface in Spring Core. It extends the functionality of the BeanFactory and provides additional features, such as internationalization and application event handling.

### What is the purpose of the BeanPostProcessor interface in Spring?

The BeanPostProcessor interface is used to customize the instantiation and initialization of beans in Spring.

### What is the purpose of the BeanFactoryPostProcessor interface in Spring?

The BeanFactoryPostProcessor interface is used to customize the configuration of the BeanFactory in Spring.

### What is the purpose of the InitializingBean interface in Spring?

The InitializingBean interface is used to perform initialization tasks after a bean has been instantiated and all its dependencies have been set. It is used in declaring the bean life cycle method.

### What is the purpose of the DisposableBean interface in Spring Core?

The DisposableBean interface is used to perform cleanup tasks before a bean is destroyed. It is also used in declaring the bean life cycle method.

### What is the purpose of the @PostConstruct annotation?

The @PostConstruct annotation is used to specify a method that should be called after a bean has been instantiated and all its dependencies have been set. It is an alternative to the InitializingBean interface.

### What is the purpose of the @PreDestroy annotation?

The @PreDestroy annotation is used to specify a method that should be called before a bean is destroyed. It is an alternative to the DisposableBean interface.

### Which annotation is generally used to inject dependencies into a bean in Spring?

The @Autowired annotation is commonly used in Spring Dependency Injection to automatically inject dependencies into a bean/class. When Spring encounters this annotation, it looks for a bean of the same type and injects it into the class.

### How does Spring Dependency Injection help with unit testing?

Spring Dependency Injection can help simplify unit testing by allowing developers to easily mock or substitute dependencies during testing. This makes it easier to isolate the behavior of the class being tested and ensures that tests are not dependent on the behavior of other classes.

### What are the advantages of using Spring Dependency Injection?

The advantages of using Spring Dependency Injection include reduced coupling between classes, improved testability, increased modularity and flexibility, and easier maintenance of the codebase.

### What is the difference between @Autowired and @Qualifier annotations in Spring Dependency Injection?

@Autowired is used to automatically inject dependencies into a class, while @Qualifier is used to specify which bean to use when multiple beans of the same type are present in the container.

### How does Spring Dependency Injection handle circular dependencies?

Spring Dependency Injection can handle circular dependencies by using lazy initialization, where beans are only initialized when they are needed. Another approach is to use setter injection, where one of the dependencies is set after the object is constructed.

### What are the best practices for using Spring Dependency Injection?

Some best practices for using Spring Dependency Injection include keeping the number of dependencies low, avoiding circular dependencies, using interfaces to define dependencies, and keeping the dependency injection container configuration simple and organized.

### What is lazy initialization in Spring container?

Lazy initialization is a feature in Spring container that delays the creation of beans until they are actually needed. This helps to improve performance and reduce resource consumption.

### What are the different ways to configure a Spring Dependency Injection?

Spring Dependency Injection can be configured using XML configuration files, Java configuration classes, or annotations. XML configuration files provide a declarative way to configure the container, while Java configuration classes and annotations provide a more programmatic way to configure the Dependency Injection.

### What is the purpose of a bean post-processor in Spring container?

A bean post-processor is a mechanism in Spring container that allows custom processing of beans before and after they are instantiated. This can be used to customize the behavior of beans or perform additional initialization tasks.

### How does Spring determine which bean to autowire?

Spring uses a set of rules to determine which bean to autowire, based on the mode of autowiring. For by type autowiring, Spring looks for a single bean of the same type as the property. For by name autowiring, Spring looks for a bean with the same name as the property. For constructor autowiring, Spring looks for a bean with a constructor argument of the same type as the property.

### What happens if Spring finds multiple beans that match the autowiring criteria?

If Spring finds multiple beans that match the autowiring criteria, it will throw an exception. To resolve this, you can use the @Qualifier annotation to specify which bean to use, or use the @Primary annotation to indicate the preferred bean.

### What is the difference between @Autowired and @Inject annotations in Spring?

@Autowired and @Inject are both used for autowiring in Spring, but they are defined by different frameworks. @Autowired is a Spring-specific annotation, while @Inject is a standard annotation defined by JSR-330 specification. The two annotations are functionally equivalent in most cases.

### How do you disable autowiring for a specific bean in Spring?

To disable autowiring for a specific bean in Spring, you can use the @Autowired annotation with the ‘required’ attribute set to ‘false’. This tells Spring to not throw an exception if the bean cannot be autowired.

### What is the default autowiring mode in Spring?

The default autowiring mode in Spring is by type. If Spring cannot find a bean of the same type as the property, it will look for a bean with the same name as the property.

♥ If you want to attempt frequently asked Multiple choice questions on Spring Core, kindly visit MCQs on Spring Core with Answers Explained.

Like

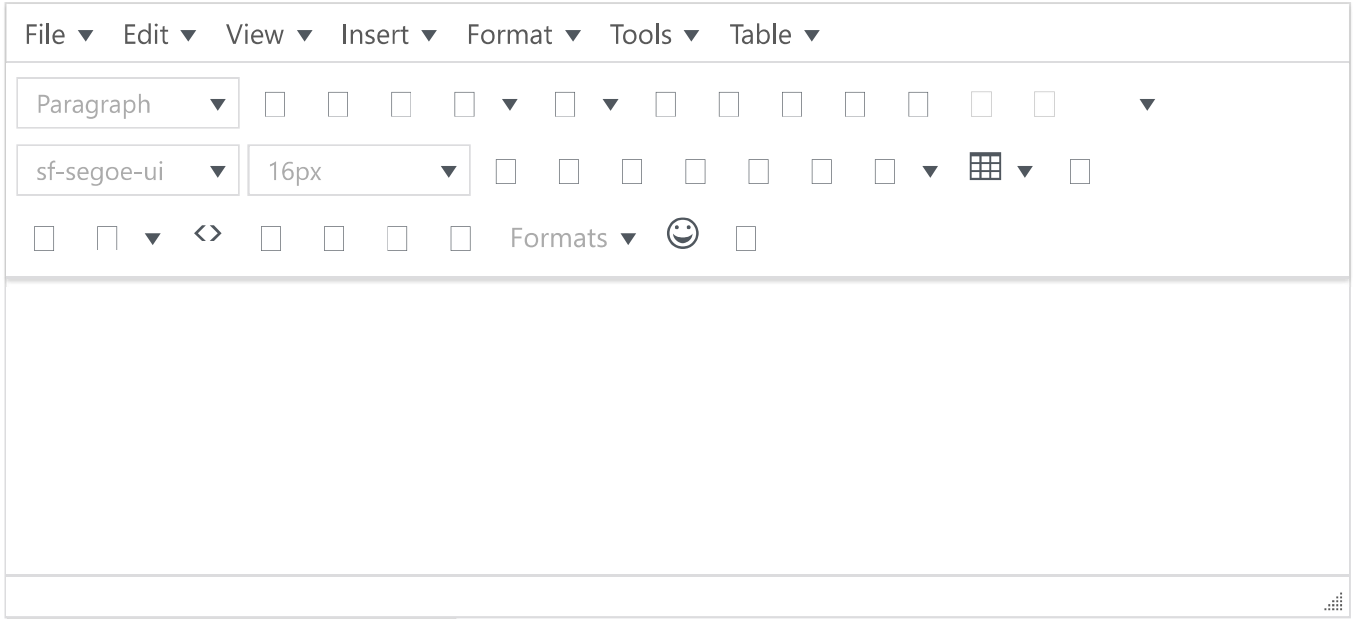


Tagged Spring Core Interview Questions


‹ Previous article
Spring Boot Annotations With Examples

Next article ›
Spring Boot MongoDB using MongoTemplate Examples


## Leave a Reply




**Name ***

**Email Address ***

**Website**

**Post Comment**

Ad

**Personalized Content Tailored to Your Preferences - All in One Place**

DiscoveryFeed

Ad

**See What Personalised Content We Have Based…**

DiscoveryFeed