Spring Framework Most Asked Interview Questions and Answers

What is Spring?

Spring is a Java framework that helps in building enterprise applications. It is a powerful toolkit for making software using Java. It's like having a set of tools that help developers build programs more easily. With Spring, tasks like connecting to databases or managing different parts of a program become simpler. It's a big help for developers because it takes care of many technical details, allowing them to focus on creating great software. It provides support for dependency injection, aspect-oriented programming, and various other features.

What are the advantages of the Spring framework?

The Spring framework has many benefits. It helps manage objects in a program, making the code simpler and easier to write. It supports transactions, which helps in managing database operations smoothly. It also integrates well with other technologies and makes testing easier. With tools like Spring Boot and Spring Cloud, developers can quickly create, deploy, and maintain scalable and reliable applications.

What are the modules of the Spring framework?

The Spring framework has many modules, such as Core for managing objects, AOP for adding extra features, Data Access for working with databases, Web for creating web applications, Security for handling security, and Test for making testing easier. There are also modules for messaging, transactions, and cloud support. Each module helps developers build strong and easy-to-maintain applications.

Difference between Spring and Spring Boot?

Spring is a framework that helps build Java applications with many tools for different tasks. Spring Boot makes using Spring easier by providing ready-made setups, reducing the need for a lot of extra code. It includes an embedded server, so we can quickly start and run applications, making development faster and simpler.

What Is a Spring Bean?

A Spring Bean is an object that is created and managed by the Spring framework. It is a key part of a Spring application, and the framework handles the creation and setup of these objects. Beans allow our application components to work together easily, making our code simpler to manage and test.

What is IOC and DI?

Inversion of Control (IoC) is a concept where the framework or container takes control of the flow of a program. Dependency Injection (DI) is a way to implement IoC, where the necessary objects are provided to a class instead of the class creating them itself. This makes the code easier to manage, test, and change.

What is the role of IOC container in Spring?

The IoC container in Spring manages the creation and setup of objects. It provides the required dependencies to these objects, making the code easier to manage and change. The container automatically connects objects and their dependencies, helping developers build applications in a more organized and efficient way.

What are the types of IOC container in Spring?

In Spring, there are two main types of IoC containers: BeanFactory and ApplicationContext. BeanFactory is the basic container that handles creating and managing objects. ApplicationContext is more advanced, adding features like event handling and easier integration with Spring's tools. Most developers prefer ApplicationContext because it offers more capabilities and is easier to use.

What is the use of @Configuration and @Bean annotations in Spring?

@Configuration indicates that a class contains @Bean definitions, and Spring IoC container can use it as a source of bean definitions. @Bean is used on methods to define beans managed by the Spring container. These methods are called by Spring to obtain bean instances.

Which Is the Best Way of Injecting Beans and Why?

The best way to inject beans in Spring is using constructor injection. It ensures that all necessary parts are provided when the object is created. This makes the object more reliable and easier to test because its dependencies are clear and cannot change.

Difference between Constructor Injection and Setter Injection?

Constructor injection gives dependencies to an object when it is created, ensuring they are ready to use immediately. Setter injection gives dependencies through setter methods after the object is created, allowing changes later. Constructor injection makes sure all needed dependencies are available right away, while setter injection allows for more flexibility in changing or adding optional dependencies later.

What are the different bean scopes in Spring?

In Spring, bean scopes define how long a bean lives. The main types are Singleton (one instance for the whole application), Prototype (a new instance each time it's needed), Request (one instance per web request), Session (one instance per user session), and Global Session (one instance per global session, used in special cases like portlet applications). These scopes help control bean creation and usage.

In which scenario will you use Singleton and Prototype scope?

Use Singleton scope when we need just one shared instance of a bean for the whole application, like for configuration settings. Use Prototype scope when we need a new instance every time the bean is requested, such as for objects that hold user-specific data or have different states for different uses.

What Is the Default Bean Scope in Spring Framework?

The default bean scope in the Spring Framework is singleton. This means that only one instance of the bean is created and shared across the entire Spring application context.

Are Singleton Beans Thread-Safe?

No, singleton beans in Spring are not thread-safe by default. Because they are shared by multiple parts of the application at the same time, we need to add extra code to make them safe for use by multiple threads. This usually means using synchronized methods or thread-safe data structures.

Can We Have Multiple Spring Configuration Files in One Project?

Yes, we can have multiple Spring configuration files in one project. This allows us to organize and manage our bean definitions and configurations more effectively by separating them into different files based on their purpose or module. We can then load these configuration files into our application context as needed.

Name Some of the Design Patterns Used in the Spring Framework?

I have used the Singleton Pattern to ensure a single instance of beans, which helps manage resources efficiently. I have also used the Factory Pattern to create bean instances, making it easier to manage and configure objects in a flexible way.

How Does the Scope Prototype Work?

The prototype scope in Spring means that a new instance of a bean is created each time it is needed. Unlike the singleton scope, which uses the same instance, the prototype scope gives a fresh, separate bean for every request. This is useful when we need a new instance for each user or operation.

What are Spring Profiles and how do you use them?

Spring Profiles provide a way to segregate parts of our application configuration and make it only available in certain environments. They can be activated via the spring.profiles.active property in application properties, JVM system properties, or programmatically. Use @Profile annotation to associate beans with profiles.

What is Spring WebFlux and how is it different from Spring MVC?

Spring WebFlux is a part of Spring 5 that supports reactive programming. It is a non-blocking, reactive framework built on Project Reactor. Unlike Spring MVC, which is synchronous and blocking, WebFlux is asynchronous and non-blocking, making it suitable for applications that require high concurrency with fewer resources.

You are starting a new Spring project. What factors would you consider when deciding between using annotations and XML for configuring your beans?

Annotations provide more concise and readable code, easier to maintain and understand, and are part of the code itself. XML configuration is better for complex configurations, offers separation of concerns, and can be modified without recompiling the code.

So, I would first consider team familiarity, project requirements, and configuration complexity and would take decision as per these cretierias.

You have a large Spring project with many interdependent beans. How would you manage the dependencies to maintain clean code and reduce coupling?

I would:

- Use dependency injection to manage dependencies.
- Utilize Spring Profiles for environment-specific configurations.
- Group related beans in separate configuration classes.
- Use @ComponentScan to automatically discover beans.

You have a singleton bean that needs to be thread-safe. What approaches would you take to ensure its thread safety?

I would:

- Use synchronized methods or blocks to control access to critical sections.
- Use ThreadLocal to provide thread-confined objects.
- Implement stateless beans where possible to avoid shared state.
- Use concurrent utilities from java.util.concurrent.