Spring Interview Questions and Answers

Introduction

Spring-developed applications are more scalable, more reliable, and easier to build and manage. Spring Framework expertise is required of Java developers. Knowledge of the Spring Framework is a must for Java developers who want to develop scalable enterprise applications. These top 20 spring interview questions and answers will help you understand the basics of IoC, AOP, and Spring Boot so that you are ready for your interview. Are you ready to learn more about it? Begin with our detailed Spring Tutorial for Beginners.

List of Spring Interview Questions for Freshers

1. What is Spring Framework?
2. What benefits does the Spring Framework offer?
3. What modules make up the Spring Framework?
4. Define dependency injection.
5. How Can Spring Beans Be Injected?

6. Which injection method is best for beans, and why?
7. What is a spring bean?
8. Are singleton beans thread-safe?
9. What's the Java-Based Spring Configuration?
10. What is Spring Security?
11. What is Spring Boot?
12. How does the prototype scope work?

Explore our Spring course syllabus to get started.

Spring Interview Questions and Answers for Freshers

1. What is Spring Framework?

The most popular framework for creating Java Enterprise Edition apps is called Spring. Moreover, any Java application may be developed using Spring's fundamental capabilities.

2. What benefits does the Spring Framework offer?

The Spring Framework has the following benefits:

- Quick Development
- Lightweight
- Predefined Templates
- Powerful Abstraction
- Declarative support
- Loose Coupling
- Easy to test

3. What modules make up the Spring Framework?

- Test
- Spring Core Container
- AOP, Aspects, and Instrumentation
- Data Access/Integration
- Web

4. Define dependency injection.

As a component of Inversion of Control (IoC), dependency injection is a broad idea that says we should define our objects' creation methods rather than creating them by hand. If necessary, an IoC container will then instantiate the necessary classes.

5. How Can Spring Beans Be Injected?

There are several methods available for injecting spring beans:

- Setter injection
- Constructor injection
- Field injection

Annotations or XML files can be used for configuration.

6. Which injection method is best for beans, and why?

It is advised to utilize setters for optional dependencies and constructor arguments for required ones. Constructor injection facilitates testing by enabling the insertion of values into immutable fields.

7. What is a spring bean?

The Spring IoC container initializes Java objects, which are known as Spring Beans.

8. Are singleton beans thread-safe?

No, singleton beans are not thread-safe since the singleton is a creation-focused design pattern, whereas thread safety is concerned with execution. The bean implementation alone determines thread safety.

9. What's the Java-Based Spring Configuration?

It's one method of type-safe application configuration for Spring-based apps. It serves as an alternative to configurations based on XML.

10. What is Spring Security?

The purpose of Spring Security, a stand-alone module of the Spring framework, is to give Java applications mechanisms for permission and authentication. Additionally, it resolves the majority of typical security flaws, including CSRF attacks.

11. What is Spring Boot?

The goal of the Spring Boot project is to minimize boilerplate configuration by offering a pre-configured set of frameworks. In this manner, we can launch a Spring application with the least amount of code possible.

12. How does the prototype scope work?

The scope prototype indicates that Spring will generate a fresh instance of the bean and return it each time we call for an instance. This is not the same as the usual singleton scope, in which each Spring IoC container instantiates a single object instance.

Explore our Spring project ideas and learn from scratch.

List of Spring Interview Questions for Experienced

1. Explain IOC and DI.
2. Describe autowiring and list its several modes.
3. In the Spring Bean Factory Container, describe the bean life cycle.
4. Does the @Controller or @RestController annotation need to be used to create a controller?
5. What kinds of dependency injections are there in Spring MVC?
6. Is spring-mvc.jar already included in Spring-Core, or is it necessary to keep it on the classpath?
7. How does the Spring Web MVC Framework handle form data validation?
8. What is MultipartResolver?

Spring Technical Interview Questions and Answers for Experienced

1. Explain IOC and DI.

A design pattern for loose coupling is IOC (Inversion of Control) with DI (Dependency Injection). The program no longer relies on it.

Without adhering to IOC and DI, let's develop some code:

```
public class Employee{

Address address;

Employee(){

address=new Address();//creating instance

}

}
```

Because "*Employee"* is required to use the same address instance, there is now a dependency between "*Employee"* and "*Address*."

Now let's create the DI or IOC code.

```
public class Employee{

Address address;

Employee(Address address){

this.address=address;//not creating instance

}

}
```

2. Describe autowiring and list its several modes.

The IoC container autowires relationships among the application beans. Colleagues can use Spring to browse the contents of the BeanFactory and identify which beans need to be wired automatically.

Some of this process's modes are:

No: This is the default setting, meaning there will be no autowiring. For wiring, a clear bean reference must be utilized.

byName: Following the bean's name, the dependency is injected. Under the configuration, this matches and wires its attributes with the beans defined by the same names.

byType: This injects the dependence on the bean according to type.

constructor: In this case, the bean dependency is injected by invoking the class's constructor. There are a lot of factors involved.

autodetect: The container attempts to autowire by type if it cannot be wired using autowire by the constructor.

3. In the Spring Bean Factory Container, describe the bean life cycle.

The life cycle of a bean looks like this:

- The IoC container instantiates the bean using its definition from the XML file.
- Then, as instructed in the bean definition, Spring uses dependency injection to fill all of the properties.
- The bean factory container invokes setBeanName(), which requires the relevant bean to implement the BeanNameAware interface and takes as input the bean ID.
- If the bean implements the BeanFactoryAware interface, the factory then passes an instance of itself to setBeanFactory().
- The preProcessBeforeInitialization() methods are called if a bean is connected with BeanPostProcessors.
- An init-method that is defined will be invoked if it is.

- Finally, if there are any BeanPostProcessors connected to the bean that are to be executed after instantiation, postProcessAfterInitialization() methods will be invoked.

4. Does the @Controller or @RestController annotation need to be used to create a controller?

Yes, you do not need to use the @Controller or @RestController annotations when creating a controller with Spring.

The @Controller and @ResstController annotations are merely convenience annotations that offer particular features; other annotations or configurations can accomplish the same behavior.

The following method can be used to construct a controller without the need for @Controller or @RestController:

- **Implement Controller Logic:** Make a standard Java class with the functionality needed to process HTTP requests and produce answers.
- **Use Appropriate Annotations:** You can specify the request mappings and the response type using other annotations in place of @Controller or @RestController.

5. What kinds of dependency injections are there in Spring MVC?

In Spring MVC, there are three different kinds of dependency injection:

- **Constructor Injection:** This technique involves injecting dependencies via a class's constructor. Since it guarantees that all necessary dependencies are present at the time the object is generated, it is regarded as the best technique for dependency injection.

- **Setter Injection:** Setter methods are used to insert dependencies in setter injection. Dependencies can be optional thanks to setter injection because not all setters must be called when creating an object. Because the dependencies can be altered after the object is formed, it may result in changeable objects.
- **Field Injection:** Dependencies are directly injected into the class fields using a technique known as field injection. Since it avoids constructor-based or setter-based DI, it is the least recommended approach because it makes it more difficult to enforce necessary dependencies and testability.

6. Is spring-mvc.jar already included in Spring-Core, or is it necessary to keep it on the classpath?

The spring-mvc.jar file fulfills distinct functions and is not a component of the spring-core library.

Spring-essential provides two of the essential Spring framework features, such as dependency injection and inversion of control.

spring-webmvc: This is the Model-View-Controller (MVC) architecture framework for creating web applications. It is commonly designated as spring-webmvc.jar rather than spring-mvc.jar. It is constructed on top of the main Spring framework.

Therefore, you would need to have both spring-core and spring-webmvc on your classpath if you were developing an application with the Spring MVC framework.

These libraries are often managed using build tools such as Maven or Gradle, and when added as dependencies to your build file, they are immediately included.

7. How does the Spring Web MVC Framework handle form data validation?

Form data validation in the Spring Web MVC Framework is handled through the following measures:

Define Validation Rules: To define the validation rules, use Bean Validation API (JSR-303) annotations on your model fields, such as @NotNull, @Size, @Min, @Max, and so on.

Activating Validation: When handling the form submission in your controller, annotate the model attribute with @Valid. This starts the process of validation.

Managing Errors in Validation: Spring MVC verifies the form data, and any issues are noted in a BindingResult object. If there are any mistakes, you can inspect this object and respond appropriately, usually by bringing the user back to the form with error messages.

Error Displaying: Show the user any validation error messages from the BindingResult in your view (such as Thymeleaf or JSP).

8. What is MultipartResolver?

The Spring MVC framework defines the MultipartResolver interface, which is used to upload files. The MultipartResolver implementation handles the file upload component of the request when a form with enctype=”multipart/form-data” is submitted in a Spring web application.

You will define the MultipartResolver in your Spring setup because it is a component of the Spring DispatcherServlet’s configuration.

Conclusion

Passing a Spring interview needs a thorough grasp of Inversion of Control (IoC), Bean lifecycles, and the optimized efficiency of Spring Boot. Your ability to implement Spring Security and Data JPA to develop robust microservices will show your preparedness for a senior position in an enterprise organization. Ready to become an expert in the Java ecosystem? Boost your knowledge with our Spring Framework Masterclass from the best software training institute in Chennai.