Interview Questions: Introduction to Spring Core and Spring Boot

1. **What is the Spring Framework and why should one use frameworks?**

Spring is an open-source framework which reduces the complexity of developing enterprise-level applications. It eases the development process by taking care of the boilerplate codes associated with creating loosely coupled applications. A framework provides us with a well-tested generic boilerplate code written by some of the best software developers who follow design principles and best practices. Using a framework will not only save time but also insert the correct components in our applications.

2. **Explain what inversion of control is?**

Inversion of control (IoC) is a design principle. This principle states that dependent classes should not create objects of the dependency classes. Dependency objects would be created by the IoC container or framework and provided to the dependent object.

3. **What is dependency injection? What are the different ways in which dependency injection can be performed in Spring? Which annotation is used to perform dependency injection?**

Dependency injection (DI) helps us to inject dependencies into an object. DI is one of the implementations of IoC and instructs how to provide dependency objects to the dependent objects.

In Spring, DI is performed using the @Autowired annotation. We can perform DI in the following three ways:

- Field or property injection
- Setter injection
- Constructor injection

Since DI can be done in three ways in Spring, the @Autowired annotation can be applied over the following:

- Fields or properties
- Setter methods
- Constructors

4. **What are Spring containers and Spring beans? Which annotation is used to create beans in Spring?**

The IoC framework creates objects of the dependency classes that can be used by the dependent classes. The IoC frameworks behave like containers for such objects and are also called IoC containers. The IoC container of the Spring Framework is called the Spring container. In Spring, the dependency objects contained inside the Spring container are called Spring beans or, simply, beans.
Beans in Spring can be created by making use of the @Component annotation.

5. **What are the different ways in which you can provide metadata to Spring for bean creation?**

There are three ways to provide metadata to Spring, which are as follows:

- **XML-based configuration:** We can provide an XML file in which all the dependency classes are listed.
- **Java-based configuration:** We can provide a type of factory class containing methods that return the object of the dependency classes. The Spring container executes these methods to instantiate dependency classes. This factory class needs to be marked with the @Configuration annotation.
- **Annotation-based configuration:** Annotations are a way to provide metadata about our code. Annotations are similar to comments, not for other developers but for the tools that process our source code; here, that tool is the Spring Framework. Annotation-based configuration helps us provide a shorter and more concise configuration.

6. **If there are two beans of the same type, how will you instruct Spring about which bean needs to be injected?**

If there is more than one bean of the same type, then we can use the @Qualifier annotation to specify which bean needs to be injected by using the bean name.

7. **Explain in brief the different bean scopes.**

The Spring Framework supports five types of bean scopes. These are as follows:

- **Singleton:** Singleton is the default scope of Spring bean inside the Spring IoC container. The Spring IoC container gives the same bean reference every time a request is made for a bean using the getBean() method.
- **Prototype:** The Spring IoC container gives a new bean instance reference every time a request is made for a bean using the getBean() method.
- **request:** The Spring container gives a single bean instance per HTTP request.
- **session:** The Spring container gives a single bean instance per HTTP session (user-level session).

- **global-session:** The Spring container gives a single bean instance per global HTTP session (application-level session).

8. **What is Spring Boot and how it is different from Spring?**

Spring eliminates the need for 'boilerplate code for loosely coupled applications', whereas Spring Boot eliminates the need for 'boilerplate configurations for Spring applications' so that we can focus only on the business logic. Spring Boot is one of the sub-projects of the Spring Framework, which eliminates the need to provide boilerplate configurations by taking an opinionated view of building Spring applications.

9. **What are the three main annotations in Spring Boot?**

In Spring Boot, auto-configuration is triggered by annotating the Main class with @SpringBootApplication annotation. This annotation is a combination of the following three annotations:

- @SpringBootConfiguration: It enables us to provide a Java-based configuration in the Main class.
- @EnableAutoConfiguration: It triggers auto-configuration.
- @ComponentScan: It enables us to scan the @Component classes, which are present in the package that contains the Main class or its sub-packages.

10. **What are Spring Boot starters?**

Spring Boot starters contain a list of all related dependency jars, which are used together to implement a type of feature. So, instead of adding maven dependencies for each jar, we can just mention the starter POM using the <dependency> tag and it will help us download all the related dependencies into the project.

11. **What is a layered architecture? Explain in brief the three layers of a web application.**

Layered architecture is a technique that helps us build loosely coupled applications by breaking down an application into layers and organising its different components into different layers based on the roles and responsibilities carried out by those components. Typically, a web application is distributed into the following three layers:

- Data access layer: Data access layer is a part of the back-end segment. All the components responsible for interacting with the database will be a part of this layer.
- Service/Business layer: Service layer is also a part of the back-end segment. It consists of all components that are responsible for processing the client request, based on the business logic, and preparing the response. The components inside this layer use the

components of the data access layer to process the client request and prepare the response.

- Presentation/Controller layer: Presentation layer is a part of the front-end as well as the back-end. In the back-end, it consists of the controller layer or the endpoints, which are used by the front-end to send the request to the back-end and wait for the response. This layer will consist of components that expose the endpoints to the front-end. The components inside this layer use the components of the service layer to process the request and generate the response.

12. Where do you put all the properties of your application for Spring Boot?

We can put all the properties of our application for Spring Boot in the application.properties file present in the resources folder.

13. Mention the possible sources of external configuration for your Spring Boot project.

Following are the possible ways in which we can configure our Spring Boot project:

- application.properties file
- Command-line argument
- Java system properties
- Environment variables
- YAML files

14. What are Spring profiles?

An application goes through multiple stages while development, typically ‘dev’, ‘testing’ and ‘prod’. At different stages, we would want to configure our application differently. This is where profiles help us provide environment-specific properties for applications. We can provide environment-specific custom configurations in the application-{profile}.properties file. We can also control bean creation for specific profiles by marking a component class with the @Profile("profile-name") annotation.