**Spring Frameworks**

**Introduction and IOC**

**Dependancy Injection**

**Autowiring + bean life cycle Dependancy checking depends-on Required**

Q1) what is bean and scope of bean what?

**Ans-** In Spring, **the objects that form the backbone of your application and that are managed by the Spring IoC container** are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container.

Bean Scopes refers to the lifecycle of Bean that means 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. Basically, it controls the instance creation of the bean and it is managed by the spring container.

Q2) Bean Scopes in Spring

Ans-When defining a <bean> you have the option of declaring a scope for that bean. For example, to force Spring to produce a new bean instance each time one is needed, you should declare the bean's scope attribute to be **prototype**. Similarly, if you want Spring to return the same bean instance each time one is needed, you should declare the bean's scope attribute to be **singleton**.

|  |  |
| --- | --- |
| 1 | **Singleton**  This scopes the bean definition to a single instance per Spring IoC container (default). |
| 2 | **Prototype**  This scopes a single bean definition to have any number of object instances. |
| 3 | **Request**  This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext. |
| 4 | **Session**  This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |
| 5 | **global-session**  This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |

Q3) is @component scan,@ configuration

Ans- Using @ComponentScan in a Spring Application. With Spring, we use the @ComponentScan annotation along with the @Configuration annotation to specify the packages that we want to be scanned. **@ComponentScan without arguments tells Spring to scan the current package and all of its sub-packages**.

Q4) which annotation u have to used in spring ?

Ans- Below

Q5) What is Spring?

Ans- Spring is **one of the most popular open source framework for developing enterprise applications**. It provides comprehensive infrastructure support for developing Java based applications. Spring also enables the developer to create high performing, reusable, easily testable and loose coupling enterprise Java application.

Q6) Which annotations you use in spring?

Ans-Below

Q7) what is @controller ?

Ans- The @Controller annotation **indicates that a particular class serves the role of a controller**. Spring does not require you to extend any controller base class or reference the Servlet API.

Q8) Why we use @componentscan and @qualifier

Ans- The @ComponentScan annotation is used with the @Configuration annotation **to tell Spring the packages to scan for annotated components**. @ComponentScan also used to specify base packages and base package classes using thebasePackageClasses or basePackages attributes of @ComponentScan.

The @Qualifier annotation in Spring is used **to differentiate a bean among the same type of bean objects**. If we have more than one bean of the same type and want to wire only one of them then use the @Qualifier annotation along with @Autowired to specify which exact bean will be wired

Q9) Advantages of spring

Ans-There are the following advantages of the Spring framework:

1. **Light Weight:** Spring is a lightweight framework because of its POJO implementation. It does not force the programmer to inherit any class and implement any interface. With the help of Spring, we can enable powerful, scalable applications using POJOs (Plain Old Java Object).
2. **Flexible:** It provides flexible libraries trusted by developers all over the world. The developer can choose either XML or Java-based annotations for configuration options. The IoC and DI features provide the foundation for a wide-ranging set of features and functionality. It makes the job simpler.
3. **Loose Coupling:** Spring applications are loosely coupled because of dependency injection. It handles injecting dependent components without a component knowing where they came from.
4. **Powerful Abstraction:** It provides a powerful abstraction to JEE specifications such as JMS, JDBC, JPA, and JTA.
5. **Declarative Support:** It provides declarative support for caching, validation, transaction, and formatting.
6. **Portable:** We can use server-side in web/EJB app, client-side in swing app business logic is completely portable.
7. **Cross-cutting behavior:** Resource management is a cross-cutting concern, easy to copy and paste everywhere.
8. **Configuration:** It provides a consistent way of configuring everything, separate configuration from application logic, varying configuration.
9. **Lifecycle:** Responsible for managing all your application components, particularly those in middle-tier container sees components through well-defined lifecycle: init(), destroy().
10. **Dependency Injection:** The use of dependency injection makes the easy development of JavaEE.
11. **Easier Testing:** The use of dependency injection makes the testing easy. The spring framework does not require a server while the EJB and Struts application requires a server.
12. **Fast:** The team of Spring engineers deeply cares about the performance. Its fast startup, fast shutdown, and optimized execution maintain performance make it fast. Even, we can start a new Spring project in seconds by using Spring Initializr.
13. **Secure:** It monitors third-party dependencies closely. The regular update is issued that make our data and application secure. We can make our application secure by using the Spring Security framework. It provides industry-standard security schemes and delivers a trustworthy solution that is secure by default.
14. **Supportive:** The Spring community provides support and resources to get you to the next level QuickStart guides, tutorials, videos, and meetup helps a lot.
15. **Productive:** It is more productive because the spring application can integrate with other Spring-based applications. For example, we can combine the Spring Boot application with Spring Cloud.

Q10) Spring IOC container?

Ans- The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

* to instantiate the application class
* to configure the object
* to assemble the dependencies between the objects

There are two types of IoC containers. They are:

1. **BeanFactory**
2. **ApplicationContext**

Q11) Autowiring concept

Ans-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.

Q12) What is the difference between @Service and @ Component?

Ans-@Component is a generic stereotype for any Spring-managed component. **@Service annotates classes at the service layer**. @Repository annotates classes at the persistence layer, which will act as a database repository.Spring – When creating object by default what is the scope of the object

Q13) Dependency checking?types

Ans- In Spring,you can use dependency checking feature to **make sure the required properties have been set or injected**.  
...

* none dependency checking. ...
* simple dependency checking. ...
* objects dependency checking. ...
* all dependency checking.

Q14) Dependency injection

Ans-Dependency Injection is **a fundamental aspect of the Spring framework, through which the Spring container “injects” objects into other objects or “dependencies”**. Simply put, this allows for loose coupling of components and moves the responsibility of managing components onto the container.

Spring framework provides two ways to inject dependency

* By Constructor
* By Setter method

Q15) Bean life cycle

Ans- Bean life cycle is **managed by the spring container**. When we run the program then, first of all, the spring container gets started. After that, the container creates the instance of a bean as per the request, and then dependencies are injected. And finally, the bean is destroyed when the spring container is closed

Q16) Default type of scope

Ans-**Singleton**. Singleton is the default scope for a Bean, the one that will be used if nothing else is indicated. This scope implies that Spring container will create an only shared instance of the class designated by this bean, so each time the Bean is required the same object will be injected.

Q17) What types of filters we use?

Ans-A filter is **an object used to intercept the HTTP requests and responses of your application**. By using filter, we can perform two operations at two instances − Before sending the request to the controller. Before sending a response to the client.

**There are five types of filters available for ComponentScan.** **Filter :**

* ANNOTATION.
* ASSIGNABLE\_TYPE.
* ASPECTJ.
* REGEX.
* CUSTOM.

Q18) what is returned type of action result ?

Ans-Actionresult is an abstract class and it is the base class for all types of action results like **ViewResult,PartialViewResult,JsonResult,ContentResult,** **JavascriptResult,RedirectResult,RedirectToRouteResult,FileResult and EmptyResult**.

Q19) How spring create object? - on what bases spring create class for object?

Ans-In Spring, **the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans**. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. Otherwise, a bean is simply one of many objects in your application.

1. create the class.
2. create the xml file to provide the values.
3. create the test class.
4. Load the spring jar files.
5. Run the test class.

**There are three different ways in which you can define a Spring bean:**

1. annotating your class with the stereotype @Component annotation (or its derivatives)
2. writing a bean factory method annotated with the @Bean annotation in a custom Java configuration class.
3. declaring a bean definition in an XML configuration file.

Q20) If spring bean object not found then it will throw which exception

Ans- BeanCreationException. . This means that **Spring found a bean to create, but was unable to fulfill the dependencies needed to create this this Spring bean**

**Q21)** diffrence bet spring and spring bean

**Ans-** A Spring bean is **an object that form the backbone of your application and that is managed by the Spring IoC container**. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. Otherwise, a bean is simply one of many objects in your application

Q21) IOC with Bean

Ans-**A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container**. Otherwise, a bean is simply one of many objects in your application. Beans, and the dependencies among them, are reflected in the configuration metadata used by a container.

Q22) How to create bean in Spring in application context

Ans- The first step is to create factory object where we used framework APIFileSystemXmlApplicationContext to create the factory bean after loading the bean configuration file from the given path. ...

The second step is used to get the required bean using getBean() method of the created context.

Q23) dependancy injection

Ans-Dependency Injection is **a fundamental aspect of the Spring framework, through which the Spring container “injects” objects into other objects or “dependencies”**. Simply put, this allows for loose coupling of components and moves the responsibility of managing components onto the container.

Q24)What is IOC ?

Ans-Spring IoC Container is **the core of Spring Framework**. It creates the objects, configures and assembles their dependencies, manages their entire life cycle. The Container uses Dependency Injection(DI) to manage the components that make up the application

Q25)How spring is light weight ?

Ans- Lightweight. The Spring Framework is **very lightweight with respect to its size and functionality**. This is due to its POJO implementation, which doesn't force it to inherit any class or implement any interfaces

Q26)Difference between Autowiring and dependency injection

Ans-Short answer: **Dependency Injection is a design pattern, and @autowired is a mechanism for implementing it**

Q27) How dependency injection works

Ans-Dependency Injection (DI) is a design pattern used to implement IoC. It allows the creation of dependent objects outside of a class and provides those objects to a class through different ways. **Using DI, we move the creation and binding of the dependent objects outside of the class that depends on them**.

Q28) Can we use both setters base and constructor base injection & in this situation which will work?

Ans-The good thing about Spring is that it doesn't restrict you to use either Setter Injection or Constructor Injection and **you are free to use both of them in one Spring configuration file**. Use Setter injection when a number of dependencies are more or you need readability

Q29)what is circular dependancy exception

Ans-Circular dependency in Spring happens **when two or more beans require instance of each other through constructor dependency injections**. For example: There is a ClassA that requires an instance of ClassB through constructor injection and ClassB requires an instance of class A through constructor injection.

Q30)What is Dependancy injection and inversion of control in spring?

Ans- What is Dependency Injection and Inversion of control?

Inversion of Control is a design concept that enables the creation of dependent objects to be inverted. On the flipside, **Dependency Injection, a software architectural pattern, is an implementation of the Inversion of control principle**.

Q31) What is bean ? What are the tags ?

Ans- The bean tag is **a combination of the set and push tags**, it allows you create a new instance of an object and then set the values of the variables. It then makes the bean available in the valuestack, so that it can be used in the JSP page. The Bean tag requires a java bean to work with.

Q31)Real time example of dependency injection ?

Ans- Dependency Injection can exist between two objects, for instance, **a flashlight and a battery**. The flashlight needs the battery to function. However, any changes made to the battery, such as switching it with another brand/set of batteries, does not mean the dependent object (flashlight) also needs to be changed

Q32) singleton is default scope and inner bean is prototype then how many object is created?

Ans-When you use singleton-scoped beans with dependencies on prototype beans , be aware that dependencies are resolved at instantiation time. Thus if you dependency-inject a prototype-scoped bean into a singleton-scoped bean, **a new prototype bean is instantiated and then dependency-injected into the singleton bean**.

Q33) init() and destroy() of bean scenario ?

Ans- **The init-method attribute specifies a method that is to be called on the bean immediately upon instantiation**. Similarly, destroymethod specifies a method that is called just before a bean is removed from the container.

Q34) What is application level events in spring ?

Ans-What is an Application Event? Spring application events **allow us to throw and listen to specific application events that we can process as we wish**. Events are meant for exchanging information between loosely coupled components.

Q35)i have inject one class to another how can i inject

Ans-In **setter injection** strategy, we trust the IoC container that it will first create the bean first but will do the injection right before using the bean using the setter methods. And the injection is done according to your configuration.