

1. ***What is spring framework?***

Answer->

*Spring Framework is a powerful lightweight application development framework used for Enterprise Java (JEE).*

The core features of the Spring Framework can be used in developing any Java application. It can be described as complete and modular framework. The Spring Framework can be used for all layer implementations of a real time application. It can also be used for the development of particular layer of a real time application unlike Struts and Hibernate, but with Spring we can develop all layers.

***2.What are the benefits of Spring:***

* **Lightweight:**Spring Framework is lightweight with respect to size and transparency.
* **Inversion Of Control (IOC):**In Spring Framework, loose coupling is achieved using Inversion of Control. The objects give their own dependencies instead of creating or looking for dependent objects.
* **Aspect Oriented Programming (AOP):**By separating application business logic from system services, Spring Framework supports Aspect Oriented Programming and enables cohesive development.
* **Container:**Spring Framework creates and manages the life cycle and configuration of application objects.
* **MVC Framework:**Spring Framework is a MVC web application framework. This framework is configurable via interfaces and accommodates multiple view technologies.
* **Transaction Management:**For transaction management,Spring framework provides a generic abstraction layer. It is not tied to J2EE environments and it can be used in container-less environments.
* **JDBC Exception Handling:**The JDBC abstraction layer of the Spring Framework offers an exception hierarchy, which simplifies the error handling strategy.

***3. How many types of dependencies?***

* **Setter Injection:**

Setter-based DI is realized by calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

* **Constructor Injection:**

Constructor-based DI is realized by invoking a constructor with a number of arguments, each representing a collaborator.

***4.How many types of injections?***

<https://www.mkyong.com/spring/spring-dependency-injection-di/>

***5. What is the difference between Bean Factory and application context?***

*Before seeing difference between ApplicationContext and BeanFactory, let see some similarity between both of them. Spring provides two kinds of IOC container, one is BeanFactory and other is ApplicationContext. Syntactically BeanFactory and ApplicationContext both are*[*Java interfaces*](http://javarevisited.blogspot.in/2012/04/10-points-on-interface-in-java-with.html)*and ApplicationContext extends BeanFactory. Both of them are configuration using*[*XML configuration file*](http://javarevisited.blogspot.in/2012/03/how-to-read-properties-file-in-java-xml.html)*. In short BeanFactory provides basic IOC and DI features while ApplicationContext provides advanced features. Apart from these, Here are few more difference between BeanFactory and ApplicationContext which is mostly based upon features supported by them.*

**Bean Factory**

* Bean instantiation/wiring

**Application Context**

* Bean instantiation/wiring
* Automatic BeanPostProcessor registration
* Automatic BeanFactoryPostProcessor registration
* Convenient MessageSource access (for i18n)
* ApplicationEvent publication

So if you need any of the points presented on the Application Context side, you should use ApplicationContext.

***6.What is the bean lifecycle? And what are the scopes for the bean?***

## 1. Spring Bean Scopes Type

In Spring, scope can be defined using spring bean **@Scope** annotation. Let’s quickly list down all six inbuilt bean scopes available to use in spring application context. These same scope apply to spring boot bean scope as well.

|  |  |
| --- | --- |
| **SCOPE** | **DESCRIPTION** |
| singleton (default) | Single bean object instance per spring IoC container |
| prototype | Opposite to singleton, it produces a new instance each and every time a bean is requested. |
| request | A single instance will be created and available during complete lifecycle of an HTTP request.  Only valid in web-aware Spring ApplicationContext. |
| session | A single instance will be created and available during complete lifecycle of an HTTP Session.  Only valid in web-aware Spring ApplicationContext. |
| application | A single instance will be created and available during complete lifecycle of ServletContext.  Only valid in web-aware Spring ApplicationContext. |
| Websocket | A single instance will be created and available during complete lifecycle of WebSocket.  Only valid in web-aware Spring ApplicationContext.  <https://howtodoinjava.com/spring5/core/spring-bean-scopes-tutorial/>  ***7.Difference between <bean id=””> and <bean name=””>?***  The name attribute allows for multiple **'aliases'** that becomes a collection of identifiers that can be used to identify the bean whereas there can be only one id per container.  <bean id="myBeanID" name = "myBean1,myBean2,myBean2" class="net.javapedia.bean.EmpBean">  ***8.How many phases are in bean lifecycle?***    ***9. What Auto wiring and what are the types in it?***    ***Link->***  [***https://howtodoinjava.com/spring-core/spring-beans-autowiring-concepts/***](https://howtodoinjava.com/spring-core/spring-beans-autowiring-concepts/)  ***10. How to pass list obj in application context?***    ***11.*** What is difference between Setterinjection and constructor injection? **11.Difference between constructor and setter injection**  1. [Difference between constructor and setter injection](https://www.javatpoint.com/difference-between-constructor-and-setter-injection)   There are many key differences between constructor injection and setter injection.   1. **Partial dependency**: can be injected using setter injection but it is not possible by constructor. Suppose there are 3 properties in a class, having 3 arg constructor and setters methods. In such case, if you want to pass information for only one property, it is possible by setter method only. 2. **Overriding**: Setter injection overrides the constructor injection. If we use both constructor and setter injection, IOC container will use the setter injection. 3. **Changes**: We can easily change the value by setter injection. It doesn't create a new bean instance always like constructor. So setter injection is flexible than constructor injection. |
|  |  |
|  |  |