[**http://javarevisited.blogspot.com/2011/09/spring-interview-questions-answers-j2ee.html**](http://javarevisited.blogspot.com/2011/09/spring-interview-questions-answers-j2ee.html)

**Question1: What is IOC or inversion of control?**

Ans: This question is first step towards spring and mostly interviewer starts from this question. As the name implies **inversion of control** means now we have inverted the control of creating the object from our own using new operator to container or framework

Now it’s the responsibility of container to create object as required.

We maintain one xml file where we configure our components, services, all the classes and their property. We just need to mention which service is needed by which component and container will create the object for us. This concept is known as **dependency injection** because all object dependency (resources) is injected into it by framework.

Example:

  <bean id="createNewStock" class="springexample.stockMarket.CreateNewStockAccont">   
        <property name="newBid"/>

    </bean>

In this example CreateNewStockAccont class contain getter and setter for newBid and container will instantiate newBid and set the value automatically when it is used.

**Question 2: Explain Bean-LifeCycle.**

Ans: Spring framework is based on IOC so we call it as IOC container also So Beans reside inside the IOC container beans are nothing but Plain old java object (POJO).

Following steps explain their lifecycle inside container:

1. Container will look the bean definition inside configuration file (eg.bean.xml).
2. Using reflection container will create the object and if any property is defined inside the bean definition then it will also be set.
3. If the bean implements the BeanNameAware interface, the factory calls setBeanName() passing the bean’s ID.
4. If the bean implements the BeanFactoryAware interface, the factory calls setBeanFactory(), passing an instance of itself.
5. If there are any BeanPostProcessors associated with the bean, their post- ProcessBeforeInitialization () methods will be called before the properties for the Bean are set.
6. If an init-method is specified for the bean, it will be called.
7. If the Bean class implements the DisposableBean interface, then the method destroy() will be called when the Application no longer needs the bean reference.
8. If the Bean definition in the Configuration file contains a 'destroy-method' attribute, then the corresponding method definition in the Bean class will be called.

**Question 3: what is Bean Factory, have you used XMLBean factory?**

Ans: BeanFactory is factory Pattern which is based on IoC. It is used to make a clear separation between application configuration and dependency from actual code.

XmlBeanFactory is one of the implementation of bean Factory which we have used in our project.

**org.springframework.beans.factory.xml.XmlBeanFactory is used to creat bean instance defined in our xml file.**

BeanFactory factory = new XmlBeanFactory(new FileInputStream("beans.xml"));

Or

ClassPathResource resorce = new ClassPathResource("beans.xml");   
XmlBeanFactory factory = new XmlBeanFactory(resorce);

**Question 4: What are the difference between BeanFactory and ApplicationContext in spring?**

Ans: This one is very popular spring interview question and often asks in entry level interview. ApplicationContext is preferred way of using spring because of functionality provided by it and interviewer wanted to check whether you are familiar with it or not.

|  |  |
| --- | --- |
| **ApplicationContext.** | **BeanFactory** |
| Here we can have more than one config files possible | In this only one config file or .xml file |
| Application contexts can publish events to beans that are registered as listeners | Doesn’t support. |
| Support internationalization (I18N) messages | It’s not |
| Support application lifecycle events, and validation. | Doesn’t support. |
| Support  many enterprise services such JNDI access, EJB integration, remoting | Doesn’t support. |

**Question 6: What is difference between singleton and prototype bean?**

Ans: This is another popular *spring interview questions* and an important concept to understand. Basically a bean has scopes which defines their existence on the application

**Singleton:** means single bean definition to a single object instance per Spring IoC container.  
**Prototype**: means a single bean definition to any number of object instances.

Whatever beans we defined in spring framework are singleton beans. There is an attribute in bean tag named ‘singleton’ if specified true then bean becomes singleton and if set to false then the bean becomes a prototype bean. By default it is set to true. So, all the beans in spring framework are by default singleton beans.

  <bean id="createNewStock"     class="springexample.stockMarket.CreateNewStockAccont" **singleton=”false”**>   
        <property name="newBid"/>   
    </bean>

**Question 7: What type of transaction Management spring support?**

Ans: This spring interview questions is little difficult as compared to previous questions just because **transaction management** is a complex concept and not every developer familiar with it. Transaction management is critical in any applications that will interact with the database. The application has to ensure that the data is consistent and the integrity of the data is maintained.  Two type of transaction management is supported by spring

1. Programmatic transaction mgt.

2. Declarative transaction mgt.