# General Questions:

## What is Spring framework?

### Spring is a powerful open source, application framework created to reduce the complexity of enterprise application development.

### It is light-weighted and loosely coupled.

### It has layered architecture, which allows you to select the components to use, while also providing a cohesive framework for J2EE application development.

### Spring framework is also called the framework of frameworks as it provides support to various other frameworks such as Struts, Hibernate, Tapestry, EJB, JSF etc.

## Advantages of Spring?

### Because of Spring Frameworks layered architecture, you can use what you need and leave which you don’t.

### Spring Framework enables POJO (Plain Old Java Object) Programming which in turn enables continuous integration and testability.

### Dependency Injection and Inversion of Control.

### Eliminate Boiler plate code.

### It is open-source and has no vendor lock-in.

## Spring module?

There are around 20 modules which are generalized into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation and Test.

* **Spring Core Container –**This layer is basically the core of Spring Framework.It contains the following modules :

1. Spring Core
2. Spring Bean
3. SpEL (Spring Expression Language)
4. Spring Context

* **Data Access/Integration –**This layer provides support to interact with the database. It contains the following modules :

1. JDBC (Java DataBase Connectivity)
2. ORM (Object Relational Mapping)
3. OXM (Object XML Mappers)
4. JMS (Java Messaging Service)
5. Transaction

* **Web –**This layer provides support to create web application. It contains the following modules :

1. Web
2. Web – MVC
3. Web – Socket
4. Web – Portlet

* **Aspect Oriented Programming (AOP) –** In this layer you can use Advices, Pointcuts etc., to decouple the code.
* **Instrumentation –**This layer provides support to class instrumentation and classloader implementations.
* **Test –**This layer provides support to testing with JUnit and TestNG.

Few Miscellaneous modules are given below:

* **Messaging –**This module provides support for STOMP. It also supports an annotation programming model that is used for routing and processing STOMP messages from WebSocket clients.
* **Aspects –**This module provides support to integration with AspectJ.

## Spring version

|  |
| --- |
|  |
| **Version** | **Date** | **Notes** |
| 0.9 | 2002 |  |
| 1.0 | 2003 |  |
| 2.0 | 2006 |  |
| 3.0 | 2009 |  |
| 4.0 | 2013 |  |
| 5.0 | 2017 |  |

The first version was written by [Rod Johnson](https://en.wikipedia.org/wiki/Rod_Johnson_(programmer)),

|  |  |  |
| --- | --- | --- |
| **Version** | **Logo** | **Feature** |
| **Spring 2.5** | spring 2.5 logo - Spring Interview Questions - Edureka! | This version was released in 2007. It was the first version which supported annotations. |
| **Spring 3.0** | spring 3.0 logo - Spring Interview Questions - Edureka! | This version was released in 2009. Itmade full-fledged use of improvements in Java5 and also provided support to JEE6. |
| **Spring 4.0** | Spring 4.0 logo - Spring Interview Questions - Edureka! | This version was released in 2013. This was the first version to provide full support to Java 8. |

### Spring 5: - <https://springframework.guru/what-is-new-with-spring-framework-5/>

### JDK baseline update

### Core framework revision

### Core container updates

### Functional programming with Kotlin

### Reactive Programming Model

### Testing improvements

### Library support

### Discontinued support

Spring 4 - <https://docs.spring.io/spring/docs/4.2.x/spring-framework-reference/html/new-in-4.0.html>

## What is DI?

## Design patterns used in Spring?

### 1. Dependency injection or inversion of control (IOC):

### Dependency injection is a technique in software engineering where an object can supply the dependencies of another object. Such dependency which can be used by an object is known as Service and the injection is the passing of the dependency to an object that uses it. Inversion of control or IOC is a design principle in software engineering through which custom-written computer program portions can receive the control flows from a generic framework. The Spring framework has an IOC container which is responsible for the creation of the objects, wiring the objects together, configuring these objects and handling the entire life cycle of these objects from their creation until they are completely destroyed. The container has the Dependency Injection (DI) which is used to manage the components present in an application. Such objects are known as Spring Beans. The dependency injection or IOC container is the main principle which is used in the spring framework for the decoupling process.

### 2. Factory Design Pattern:

### The Spring framework uses the factory design pattern for the creation of the objects of beans by using the following two approaches.

### **Spring BeanFactory Container:** It is the simplest container present in the spring framework which provides the basic support for DI (Dependency Injection). We use the following interface to work with this container.[org.springframework.beans.factory.BeanFactory].

### **Spring ApplicationContext Container:** It is another container present in spring container which adds extra enterprise-specific functionality. These functionalities include the capability to resolve textual messages from a properties file and publishing application events to the attentive event listeners. We use the following interface to work with this container. [org.springframework.context.ApplicationContext]. Below are the most commonly used ApplicationContext implementations. FileSystemXmlApplicationContext (need to provide the full path of the XML bean configuration file to the constructor). ClassPathXmlApplicationContext (need to set CLASSPATH of the bean configuration XML file in order to load the metadata of the beans from an XML file). WebXmlApplicationContext (the container loads the XML file within a web application which has metadata of all beans).

### **Example: –**

### 

|  |  |
| --- | --- |
| 1234567891011121314 | package com.eduonix.springframework.applicationcontext;import org.springframework.context.ApplicationContext;import org.springframework.context.support.FileSystemXmlApplicationContext;public class App {public static void main(String[] args) {ApplicationContext context = new FileSystemXmlApplicationContext("C:/work/IOC Containers/springframework.applicationcontext/src/main/resources/bean-factory-config.xml");HelloApplicationContext obj = (HelloApplicationContext) context.getBean("helloApplicationContext");obj.getMsg();}} |

### 3. Proxy Design Pattern: In the proxy design pattern, a class is used to represent the functionality of another class. It is an example of a structural pattern. Here, an object is created that has an original object to interface its functionality to the outer world. Proxy design pattern is widely used in AOP, and remoting.

### 4. Singleton Design Pattern: Singleton design pattern ensures that there will exist only the single instance of the object in the memory that could provide services. In the spring framework, the Singleton is the default scope and the IOC container creates exactly one instance of the object per spring IOC container. Spring container will store this single instance in a cache of singleton beans, and all following requests and references for that named bean will get the cached object as return bean. It is recommended to use the singleton scope for stateless beans. We can set up the bean scope as Singleton or prototype (which creates a new bean object for every new request) in the configuration XML file as shown below.

### 

|  |  |
| --- | --- |
| 1234 | <!-- A bean definition with singleton scope --><bean id = "..." class = "..." scope = "singleton/prototype">   <!-- collaborators and configuration for this bean go here --></bean> |

### 5. Model View Controller (MVC): It is a design pattern which comes into picture when we use the spring framework for web programming. Spring MVC is known to be a lightweight implementation as controllers are POJOs against traditional servlets which makes the testing of controllers very comprehensive. A controller returns a logical view name and the view selection with the help of a separate ViewResolver. Therefore, Spring MVC controllers can be used along with different view technologies such as JSP, etc.

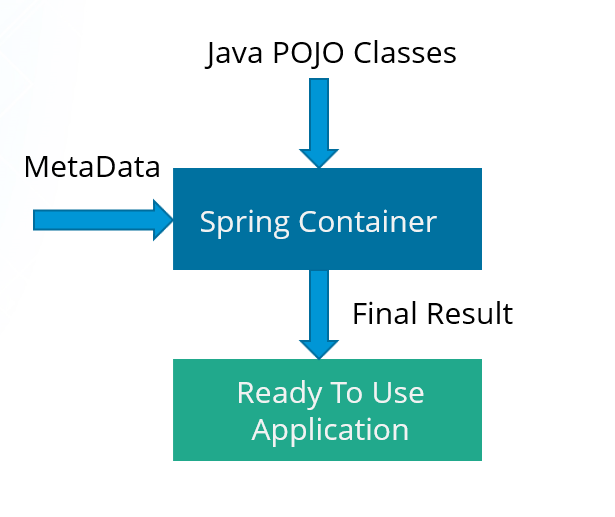
### 6. Front Controller Design Pattern: The front controller design pattern is a technique in software engineering to implement centralized request handling mechanism which is capable of handling all the requests through a single handler. Such handler can perform the authentication, authorization, and logging or request tracking (i.e., pass the requests to the corresponding handlers). Spring framework provides support for the DispatcherServlet that ensure to dispatch an incoming request to your controllers.

### 7. View Helper: Spring framework provides a large number of custom JSP tags as well as velocity macros which assist in the separation of code from the presentation (i.e., views).

### 8. Template method: Spring framework provides a number of templates to kick start work and complete that piece of work as the best programming practice such as opening and closing connection for JDBC or JMS, etc. E.g., JdbcTemplate, JmsTemplate, and JpaTemplate.

## Spring container?

### At the core of the Spring Framework, lies the Spring container. The container creates the object, wires them together, configures them and manages their complete life cycle. The Spring container makes use of Dependency Injection to manage the components that make up an application. The container receives instructions for which objects to instantiate, configure, and assemble by reading the configuration metadata provided. This metadata can be provided either by XML, Java annotations or Java code.



## What is dependency injection?

### In Dependency Injection, you do not have to create your objects but have to describe how they should be created. You don’t connect your components and services together in the code directly, but describe which services are needed by which components in the configuration file. The IoC container will wire them up together.

## **In how many ways can Dependency Injection be done?**

### In general, dependency injection can be done in three ways, namely :

### Constructor Injection

### Setter Injection

### Interface Injection

### In Spring Framework, only constructor and setter injections are used.

## Which injection is better?

For mandatory properties use Constructor Injection, for non-mandatory use Setter injection. However we can use @Required in setter as well.

## Types of IOC container?

### **BeanFactory**: BeanFactory is like a factory class that contains a collection of beans. It instantiates the bean whenever asked for by clients.

### **ApplicationContext**: The ApplicationContext interface is built on top of the BeanFactory interface. It provides some extra functionality on top BeanFactory.

#### **BeanFactory vs ApplicationContext**

|  |  |
| --- | --- |
| **BeanFactory** | **ApplicationContext** |
| It is an interface defined in org.springframework.beans.factory.**BeanFactory** | It is an interface defined in org.springframework.context.**ApplicationContext** |
| It uses Lazy initialization | It uses Eager/ Aggressive initialization |
| It explicitly provides a resource object using the syntax | It creates and manages resource objects on its own |
| It doesn’t supports internationalization | It supports internationalization |
| It doesn’t supports annotation based dependency  .  Do not support event publish. | It supports annotation based dependency .  Event publish is supported. |

## What is spring bean?

### They are the objects that form the backbone of the user’s application.

### Beans are managed by the Spring IoC container.

### They are instantiated, configured, wired and managed by a Spring IoC container

### Beans are created with the configuration metadata that the users supply to the container.Bean generation - Spring Interview Questions - Edureka!

## What is bean wiring?

### The act of creating these associations between application objects is the essence of dependency injection (DI) and is commonly referred to as wiring.

Different ways of wiring-

### Explicit configuration in XML

###  Explicit configuration in Java

###  Implicit bean discovery and automatic wiring

### **XML-Based configuration:**In Spring Framework, the dependencies and the services needed by beans are specified in configuration files which are in XML format. These configuration files usually contain a lot of bean definitions and application specific configuration options. They generally start with a bean tag. For example:

|  |  |
| --- | --- |
| 123 | <bean id="studentbean" class="org.edureka.firstSpring.StudentBean"> <property name="name" value="Edureka"></property></bean> |

### **Annotation-Based configuration**: Instead of using XML to describe a bean wiring, you can configure the bean into the component class itself by using annotations on the relevant class, method, or field declaration. By default, annotation wiring is not turned on in the Spring container. So, you need to enable it in your Spring configuration file before using it. For example:

|  |  |
| --- | --- |
| 1234 | <beans><context:annotation-config/><!-- bean definitions go here --></beans> |

### **Java-based configuration:**The key features in Spring Framework’s new Java-configuration support are @Configuration annotated classes and @Bean annotated methods.

### 1. @Bean annotation plays the same role as the <bean/> element.

### 2.@Configuration classes allows to define inter-bean dependencies by simply calling other @Bean methods in the same class.

@Runwith(SpringJunit4ClassRunner)

@ContextConfiguration(class=)