Let's begin with the first section of Spring interview questions that is the General Questions.

General Questions – Spring Interview Questions

1. What are the major features in different versions of Spring Framework?

Version	Logo	Feature
Spring 2.5		This version was released in 2007. It was the first version which supported annotations.
Spring 3.0		This version was released in 2009. It made full-fledged use of improvements in Java5 and also provided support to JEE6.
Spring 4.0		This version was released in 2013. This was the first version to provide full support to Java 8.

Features of Spring Framework

2. What is a 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.

3. List the advantages of Spring Framework.

- 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.
- JDBC is simplified due to Dependency Injection and Inversion of Control.
- It is open-source and has no vendor lock-in.

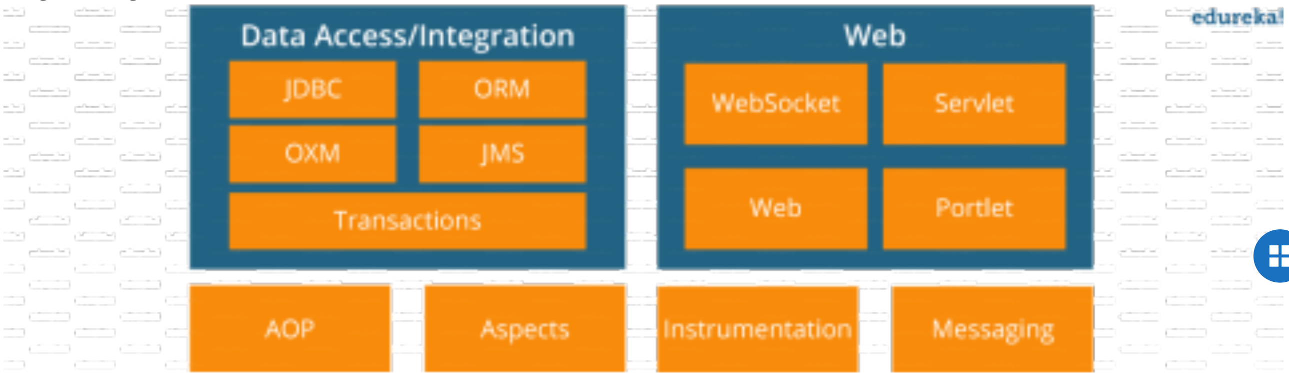
4. What are the different features of Spring Framework?

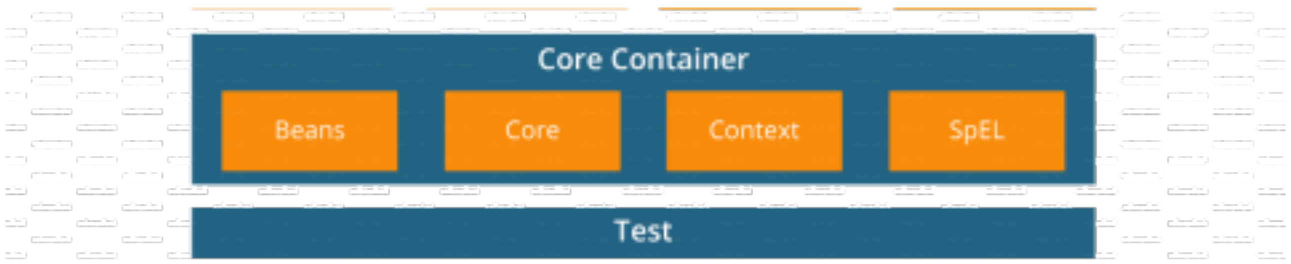
Following are some of the major features of Spring Framework :

- **Lightweight:** Spring is lightweight when it comes to size and transparency.
- **Inversion of control (IOC):** The objects give their dependencies instead of creating or looking for dependent objects. This is called Inversion Of Control.
- **Aspect oriented Programming (AOP):** Aspect oriented programming in Spring supports cohesive development by separating application business logic from system services.
- **Container:** Spring Framework creates and manages the life cycle and configuration of the application objects.
- **MVC Framework:** Spring Framework's MVC web application framework is highly configurable. Other frameworks can also be used easily instead of Spring MVC Framework.
- **Transaction Management:** Generic abstraction layer for transaction management is provided by the Spring Framework. Spring's transaction support can be also used in container less environments.
- **JDBC Exception Handling:** The JDBC abstraction layer of the Spring offers an exception hierarchy, which simplifies the error handling strategy.

5. How many modules are there in Spring Framework and what are they?

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 :

a. Spring Core
b. Spring Bean
c. SpEL (Spring Expression Language)
d. Spring Context

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

a. JDBC (Java DataBase Connectivity)
b. ORM (Object Relational Mapping)
c. OXM (Object XML Mappers)
d. JMS (Java Messaging Service)
e. Transaction

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

a. Web
b. Web – MVC
c. Web – Socket
d. 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.

6. What is a Spring configuration file?

A Spring configuration file is an XML file. This file mainly contains the classes information. It describes how those classes are configured as well as introduced to each other. The XML configuration files, however, are verbose and more clean. If it's not planned and written correctly, it becomes very difficult to manage in big projects.

7. What are the different components of a Spring application?

A Spring application, generally consists of following components:

- Interface: It defines the functions.
- Bean class: It contains properties, its setter and getter methods, functions etc.
- Spring Aspect Oriented Programming (AOP): Provides the functionality of cross-cutting concerns.
- Bean Configuration File: Contains the information of classes and how to configure them.
- User program: It uses the function.

8. What are the various ways of using Spring Framework?

Spring Framework can be used in various ways. They are listed as follows:

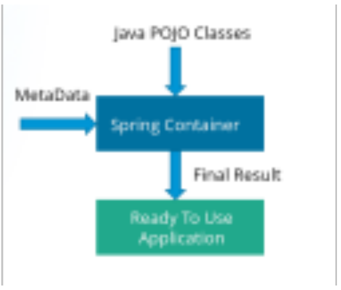
1. As a Full-fledged Spring web application.
2. As a third-party web framework, using Spring Frameworks middle-tier.
3. For remote usage.
4. As Enterprise Java Bean which can wrap existing POJOs (Plain Old Java Objects).

The next section of Spring Interview Questions is on *Dependency Injection and IoC container*.

Dependency Injection/ IoC Container – Spring Interview Questions

9. What is Spring IOC 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.



10. What do you mean by 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.

11. 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.

12. Differentiate between constructor injection and setter injection.

Constructor Injection vs Setter Injection

Constructor Injection	Setter Injection
There is no partial injection.	There can be partial injection.
It doesn't override the setter property.	It overrides the constructor property.
It will create a new instance if any modification is done.	It will not create new instance if any modification is done.
It works better for many properties.	It works better for few properties.

13. How many types of IOC containers are there in spring?

a. **BeanFactory**: BeanFactory is like a factory class that contains a collection of beans. It instantiates the bean whenever asked for by clients.
b. **ApplicationContext**: The ApplicationContext interface is built on top of the BeanFactory interface. It provides some extra functionality on top BeanFactory.

14. Differentiate between BeanFactory and ApplicationContext.

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	It supports annotation based dependency

15. List some of the benefits of IoC.

Some of the benefits of IoC are:

- It will minimize the amount of code in your application.
- It will make your application easy to test because it doesn't require any singletons or JNDI lookup mechanisms in your u test cases.
- It promotes loose coupling with minimal effort and least intrusive mechanism.
- It supports eager instantiation and lazy loading of the services.

Let's move on to the next section of Spring Interview Questions, that is *Spring Beans Interview Questions*.

Spring Beans – Spring Interview Questions

16. Explain Spring Beans?

- 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.

MetaData
IoC Container
Beans

17. How configuration metadata is provided to the Spring container?

Configuration metadata can be provided to Spring container in following ways:

- **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:

```
<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:

```
<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.

For example:

```
@Configuration
public class StudentConfig
{
@Bean
public StudentBean myStudent()
{ return new StudentBean(); }
}
```

18. How many bean scopes are supported by Spring?

The Spring Framework supports five scopes. They are:

- **Singleton:** This provides scope for the bean definition to single instance per Spring IoC container.
- **Prototype:** This provides scope for a single bean definition to have any number of object instances.
- **Request:** This provides scope for a bean definition to an HTTP-request.
- **Session:** This provides scope for a bean definition to an HTTP-session.
- **Global-session:** This provides scope for a bean definition to an Global HTTP-session.

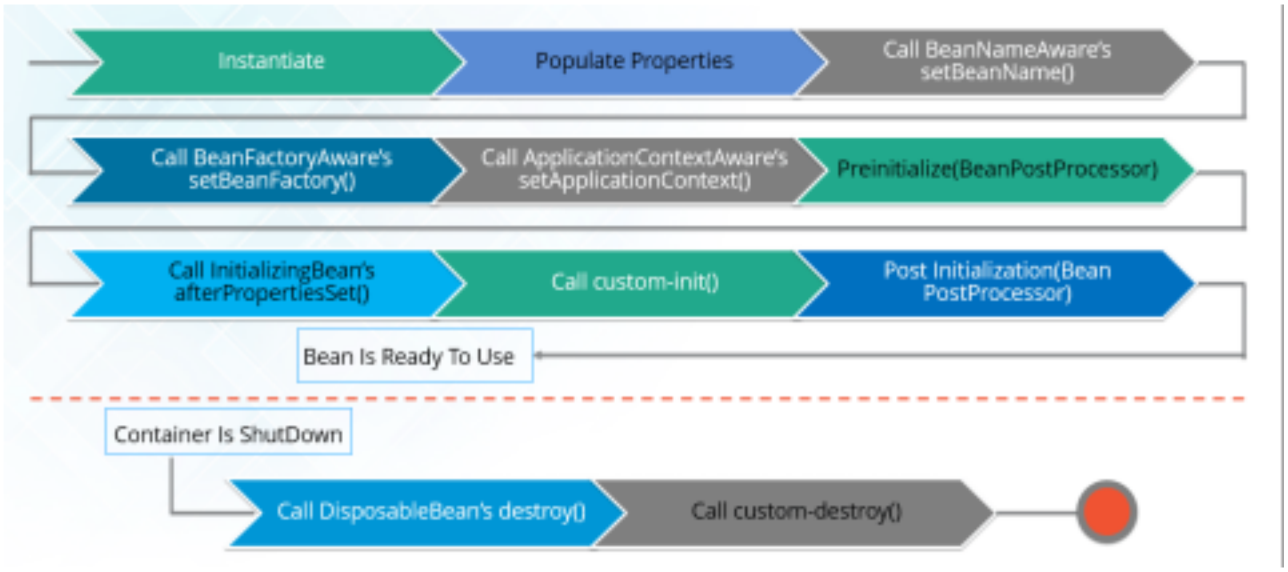
The last three are available only if the users use a web-aware ApplicationContext.

19. What is the Bean life cycle in Spring Bean Factory Container?

Bean life cycle in Spring Bean Factory Container is as follows:

1. The Spring container instantiates the bean from the bean's definition in the XML file.
2. Spring populates all of the properties using the dependency injection, as specified in the bean definition.
3. The factory calls setBeanName() by passing the bean's ID, if the bean implements the BeanNameAware interface.
4. The factory calls setBeanFactory() by passing an instance of itself, if the bean implements the BeanFactoryAware interface.
5. preProcessBeforeInitialization() methods are called if there are any BeanPostProcessors associated with the bean.
6. If an init-method is specified for the bean, then it will be called.
7. Finally, postProcessAfterInitialization() methods will be called if there are any BeanPostProcessors associated with the bean.

To understand it in better way check the below diagram:


Instantiate
Populate Properties
Call BeanNameAware's setBeanName()
Call BeanFactoryAware's setBeanFactory()
Call ApplicationContextAware's setApplicationContext()
Preinitialize(BeanPostProcessor)
Call InitializingBean's afterPropertiesSet()
Call custom-init()
Post Initialization(Bean PostProcessor)
Bean Is Ready To Use
Container Is ShutDown
Call DisposableBean's destroy()
Call custom-destroy()

20. Explain inner beans in Spring.

A bean can be declared as an inner bean only when it is used as a property of another bean. For defining a bean, the Spring's XML based configuration metadata provides the use of <bean> element inside the <property> or <constructor-arg>. Inner beans are always anonymous and they are always scoped as prototypes. For example, let's say we have one Student class having reference of Person class. Here we will be creating only one instance of Person class and use it inside Student.

Here's a Student class followed by bean configuration file:

Student.java