What is Spring? [answer](https://www.geeksforgeeks.org/introduction-to-spring-framework/)

· Lightweight java framework to create high performing, easily testable, and reusable code.

· The Spring framework comprises several modules such as IOC, AOP, DAO, Context, ORM, WEB MVC etc

· Testing an application written with Spring is simple

What are Spring Projects and Spring modules

[Spring Modules](https://www.javatpoint.com/spring-modules)

Spring projects: spring.io/projects/spring-framework. The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE (Enterprise Edition) platform.

What is IOC and what does the IOC Container do?

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. It gets the information about the objects from a configuration file(XML) or Java Code or Java Annotations and [Java POJO class](https://www.geeksforgeeks.org/pojo-vs-java-beans/). These objects are called Beans. Since the Controlling of Java objects and their lifecycle is not done by the developers, hence the name Inversion Of Control.

What is dependency injection and what are some of the benefits of using dependency injection?

The Dependency Injection is a design pattern that removes the dependency of the programs. In such case we provide the information from the external source such as XML file. It makes our code loosely coupled and easier for testing. The Spring-Core module is responsible for injecting dependencies through Constructor or Setter methods.

benefits

· The dependency injection approach makes the code **loosely coupled.** If resource is changed, we need not perform a lot of modification in the code.

· This approach makes problems testing in the application especially in black box **testing easier.**

What are some differences between BeanFactory and ApplicationContext? Which one eagerly instantiates your beans?

| Feature | BeanFactory | ApplicationContext |
| --- | --- | --- |
| Annotation Support | No | Yes |
| **Bean Instantiation/Wiring** | **Yes** | **Yes** |
| Internationalization | No | Yes |
| Enterprise Services | No | Yes |
| ApplicationEvent publication | No | Yes |
| Automatic BeanPostProcessor  registration | No | Yes |
| Loading Mechanism | Lazy loading | Aggressive loading |
| Automatic BeanFactoryPostProcessor  registration | No | Yes |

What is the Spring Bean lifecycle? [answer](https://www.geeksforgeeks.org/bean-life-cycle-in-java-spring/)

What is bean wiring? What about autowiring?

**Bean wiring:** Combining together beans within the Spring container is known as bean wiring or wiring. When wiring beans, you should tell the container what beans are needed and how the container should use dependency injection to tie them together.

**AutoWiring:** Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection.

**Advantage of Autowiring**

It requires the less code because we don't need to write the code to inject the dependency explicitly.

**Disadvantage of Autowiring**

No control of programmer.

can't be used to inject primitive and string values. It works with reference only

What are the different ways that Spring can wire beans?

Spring offers three primary wiring mechanisms:

1. Implicit bean discovery and automatic wiring.
2. Explicit configuration in Java.
3. Explicit configuration in XML.

What are the scopes of Spring beans? Which is default?

* Singleton(default)
* Prototype
* Request
* Session
* Global session
* Application

Request, Session, and Application are for beans in Web-aware applications. Global session beans are for portlets.

## **The Singleton Bean Scope**

When you create a bean with the Singleton scope, the Spring Framework will create an instance of the bean only once. The Framework returns that instance each time the bean is requested by your application code.

## **The Prototype Bean Scope**

When you create a bean with the Prototype scope, the Spring Framework will create a bean instance each time the bean is requested by your application code.

## **The Request Bean Scope**

The request scope is applicable to beans of Web aware applications. This scope defines a single bean definition that lives within a single HTTP request. This means every HTTP request will have its own instance of a bean.

## **The Session Bean Scope**

The session scope defines a single bean definition which lives within the lifecycle of an HTTP Session. Similar to the Request scope, the Session scope is applicable to beans in Web applications.

There are two other lesser used scopes: Global-Session and Application scopes

Global session scope defines a single bean definition to the lifecycle of a global HTTP Session. This scope is valid when used in a portlet context.

In the application scope, Spring creates a bean instance per web application runtime. It is similar to singleton scope, with one major difference. Singleton scoped bean is singleton per ApplicationContext where application scoped bean is singleton per ServletContext.

What is the concept of component scanning and how would you set it up?

If you understand Component Scan, you understand Spring.

Spring is a dependency injection framework. It is all about beans and wiring in dependencies.

The first step of defining Spring Beans is by adding the right annotation — @Component or @Service or @Repository.

*However, Spring does not know about the bean unless it knows where to search for it.*

*This part of “telling Spring where to search” is called a Component Scan.*

You define the packages that have to be scanned.

Once you define a Component Scan for a package, Spring would search the package and all its sub packages for components/beans.

**Defining a Component Scan**

If you are using Spring Boot, check the configuration in Approach 1.

If you are doing a JSP/Servlet or a Spring MVC application without using Spring Boot, use Approach 2.

Approach 1: Component Scan in a Spring Boot Project

If your other package hierarchies are below your main app with the @SpringBootApplication annotation, you’re covered by the implicit Component Scan.

If there are beans/components in other packages that are not sub-packages of the main package, you should manually add them as @ComponentScan

You have two options:

Define @ComponentScan(“com.in28minutes.springboot”)

This would scan the entire parent tree of com.in28minutes.springboot.

Or define two specific Component Scans by using an array.

@ComponentScan({“com.in28minutes.springboot.basics.springbootin10steps”,”com.in28minutes.springboot.somethingelse”})

Approach 2: Non-Spring Boot Project

In a non-Spring Boot Project, we would typically define the component scan explicitly in an XML application context or a Java Application Context.

@Component vs. @ComponentScan

@Component and @ComponentScan are for different purposes.

@Component indicates that a class might be a candidate for creating a bean. It's like putting a hand up.

@ComponentScan is searching packages for Components. Trying to find out who all put their hands up.

What are the benefits and limitations of Java configuration?

**Advantages of Java Config over XML Config**

1. Compile-Time Feedback due to Type-checking

2. Refactoring Tools for Java without special support/plugins

**Advantages of Java Config over Annotation Based Config**

1. Separation of concerns – beans configuration is separated from beans implementation. That means only configuration class know about beans. And each of the bean implementations does not have any information that is a component, what is the scope and etc. All of this information is in configuration class.

2. Technology agnostic – beans may not depend on concrete IoC/DI implementation – makes it easier to switch technology. That's because any of the beans have not dependency directly on the Spring framework.

3. Ability to integrate Spring with external libraries. Any class of the eternal library can be a Bean. Simply create a method with @Bean annotation that returns a new object.

4. The more centralized location of bean list

**Limitations of Java Config**

1. Configuration class cannot be final

2. Configuration class methods cannot be final

3. All Beans have to be listed, for big applications, it might be a challengecompared toComponent Scanning

What does the @Configuration and @Bean annotations do?

@Configuration indicates that **the class has @Bean definition methods**. So Spring container can process the class and generate Spring Beans to be used in the application.

@Bean Annotation is applied **on a method to specify that it returns a bean to be managed by Spring context**. Spring Bean annotation is usually declared in Configuration classes methods. In this case, bean methods may reference other @Bean methods in the same class by calling them directly.

What is @Value used for?

@Value is a Java annotation that is used at the field or method/constructor parameter level and it indicates a default value for the affected argument. It is commonly used for **injecting values into configuration variables**

What is Spring Expression Language? What can you reference using SpEL?(beans) What is the difference between $ and # in @value expressions?

The Spring Expression Language (SpEL) is a powerful expression language that supports querying and manipulating an object graph at runtime. It can be used with XML or annotation-based Spring configurations.

There are several operators available in the language:

| **Type** | **Operators** |
| --- | --- |
| Arithmetic | +, -, \*, /, %, ^, div, mod |
| Relational | <, >, ==, !=, <=, >=, lt, gt, eq, ne, le, ge |
| Logical | and, or, not, &&, ||, ! |
| Conditional | ?: |
| Regex | matches |