**Ques: - what is the scope of bean?**

**Ans :-**  The [spring framework](https://www.geeksforgeeks.org/introduction-to-spring-framework/) provides five scopes for a bean. We can use three of them only in the context of web-aware Spring ApplicationContext and the rest of the two is available for both IoC container and Spring-MVC container. The following are the different scopes provided for a bean:

1. **Singleton**: Only one instance will be created for a single bean definition per Spring IoC container and the same object will be shared for each request made for that bean.
2. **Prototype**: A new instance will be created for a single bean definition every time a request is made for that bean.
3. **Request**: A new instance will be created for a single bean definition every time an HTTP request is made for that bean. But Only valid in the context of a web-aware Spring ApplicationContext.
4. **Session**: Scopes a single bean definition to the lifecycle of an HTTP Session. But Only valid in the context of a web-aware Spring ApplicationContext.
5. **Global**-**Session**: Scopes a single bean definition to the lifecycle of a global HTTP Session. It is also only valid in the context of a web-aware Spring ApplicationContext.

**Ques :- what is bean?**

**Ans :-** In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container.

**IoC Container**

The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets information’s from the XML file and works accordingly. The main tasks performed by IoC container are: to instantiate the application class

* to configure the object
* to assemble the dependencies between the objects

There are two types of IoC containers. They are:

Bean Factory (core)

ApplicationContext (j2ee)

**Ques: - Explain @SpringBootApplication?**

**Ans: -** Spring Boot @SpringBootApplication annotation is used to mark a configuration class that declares one or more @Bean methods and also triggers auto-configuration and component scanning.

It’s same as declaring a class with or @springbootApplication consist of the following

@Configuration,

@EnableAutoConfiguration and

@ComponentScan annotations.

**Ques: - what is @componentscan?**

**Ans: -** @ComponentScan annotation enables component scanning in Spring. Java classes that are decorated with stereotypes such as @Component, @Configuration, @Service are auto-detected by Spring. The @ComponentScan's basePackages attribute specifies which packages should be scanned for decorated beans.

**@ComponentScan enables Spring to scan for things like configurations, controllers, services, and other components we define.**

**Ques: - what is @EnableAutoConfiguration?**

**Ans: - The @EnableAutoConfiguration annotation enables Spring Boot to auto-configure the application context. Therefore, it automatically creates and registers beans based on both the included jar files in the classpath and the beans defined by us.**

**Ques: - what is @Configuration?**

**Ans: -** Spring @Configuration annotation is part of the spring core framework. Spring Configuration annotation 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.

**Ques: - what is @Autowired or Autowiring?**

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

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

## Autowiring Modes

|  |  |  |
| --- | --- | --- |
| **No.** | **Mode** | **Description** |
| 1) | no | It is the default autowiring mode. It means no autowiring bydefault. |
| 2) | byName | The byName mode injects the object dependency according to name of the bean. In such case, property name and bean name must be same. It internally calls setter method. |
| 3) | byType | The byType mode injects the object dependency according to type. So, property name and bean name can be different. It internally calls setter method. |
| 4) | constructor | The constructor mode injects the dependency by calling the constructor of the class. It calls the constructor having large number of parameters. |
| 5) | autodetect | It is deprecated since Spring 3. (For both setter & constructor) |

**Ques: - Explain dialect in yaml?**

**Ans: -** The dialect specifies the type of database used in hibernate so that hibernate generate appropriate type of SQL statements. For connecting any hibernate application with the database, it is required to provide the configuration of SQL dialect.

Example

|  |  |
| --- | --- |
| PostgreSQL | org.hibernate.dialect.PostgreSQLDialect |





