**Spring Annotations: -**

A screenshot of a computer

Description automatically generated

Above annotation use to create spring bean automatically in application context, it also manage spring bean life cycle, here all three annotation @service, @restController, and @Repository derived from @component annotation.

@service/RestController/Repository say the roll of the classes.

Here Rest Controller is used for exposed end point/rest api where @Service annotation show written some business logic.

@repository layer we interact with database

With the above annotation someone can easily identify whether it is service classes or Controller or Database related operation layer

**Spring Core Annotations:-**

A screenshot of a computer

Description automatically generated

**@Configuration and @Bean** used when you want to do java base configuration means don’t want to handle spring bean life cycle either by annotation or xml

**@Autowired**: -Here you tell spring boot IOC container to create object of the injected class and give it to me so I can use.

If you don’t mention Autowired annotation, then you get null pointer exception.

**@Qualifier: -** Suppose you have StudentService interface, and you have one implementation class i.e StudentServiceImpl. Now you have created one more implementation class StudentServiceImplV2.

In this case you will get the exception below.

A white background with black text

Description automatically generated

To resolve this problem, you need to use the above annotation and tell spring IOC container about which impl class you are going to use.

**@Primary: -**

One more solution for the above problem is to use @primary annotation.

One best example is when you have multiple data source, and you want to use one of them.

A screenshot of a computer program

Description automatically generated

**@Lazy: -**by default spring bean is eager loading which increase to heap size. We avoid this using @Lazy Loading. Here beans are created on demand only.

Object is created only when you use it.

**@Value:** -here suppose you want to use mail related attributes from prop file

A screenshot of a computer

Description automatically generated

A close-up of a computer code

Description automatically generated

**@PropertySource: -**

application.prop file automatically load by spring container but if you want to use any custom property then spring won’t load. You need to use @PropertySource to use/load this custom prop file.

A close-up of a logo

Description automatically generated

**@ConfigurationProperties: -**

Here you want to use bunch of mail related attributes, so you create one dto class which automatically load attributes with prefix mail.

A close-up of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of a computer code

Description automatically generated

**@Profile: -**

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**@Scope: -** It shows bean scope whether it is singleton or prototype or request or session.

A screenshot of a computer program

Description automatically generated

A white background with black text

Description automatically generated

If you use @RequestParam without using param then don’t get 404 error but in case of @Pathvariable getting 404 error

**Spring Auto Configuration: -**

[Spring Boot auto-configuration](https://www.javadevjournal.com/spring/spring-boot-auto-configuration/) automatically configure a Spring application based on the dependencies present on the class path. *Spring Boot* detects classes in the class path and auto-configuration mechanism will ensure to create and wires necessary beans for us. This is one of the most powerful features of the ***Spring Boot*** and most of the work happens silently in the background.

**Name Some of the Design Patterns Used in the Spring Framework?**

* **Singleton Pattern** – singleton-scoped beans
* **Factory Pattern** – Bean Factory classes
* **Prototype Pattern** – prototype-scoped beans
* **Adapter Pattern** – Spring Web and Spring MVC
* **Proxy Pattern** – Spring Aspect-Oriented Programming support
* **Template Method Pattern** – *JdbcTemplate*, *HibernateTemplate*, etc.
* **Front Controller** – Spring MVC *DispatcherServlet*
* **Data Access Object** – Spring DAO support
* **Model View Controller**– Spring MVC

**Q9. What Is the Default Bean Scope in Spring Framework?**

By default, a Spring Bean is initialized as a *singleton*.

**Q10. How to Define the Scope of a Bean?**

In order to set Spring Bean’s scope, we can use *@Scope* annotation or “scope” attribute in XML configuration files. Note that there are five supported scopes:

* **Singleton**
* **Prototype**
* **Request**
* **Session**
* **Global-session**