**Spring short Note**

* **S**pring version latest 5.0.0-release
* **Java** configuration
* **Loose coupling**
* **Inversion of control**
* **Spring module**
* **Spring project**
* **Important annotation**

**@primary**

**@Qualifier**

**@Autowire:-**

* You can apply the @Autowired annotation to constructors:
* It is also possible to provide all beans of a particular type from the ApplicationContext by adding the annotation to a field or method that expects an array of that type

**@ComponentScan**

**@Configuration**

@Scope(ConfiguarbleBeanFactory=Siggelton , proxyMode=ScopedProxyMode.Targetclass)

@Component

@Service

@Repository

@Bean

@PostConstruct

@PreDestory

@Required:- The @Required annotation applies to bean property setter methods.

@Resource

@order

@priority

@Autowire(required=false):- By default, the autowiring fails whenever zero candidate beans are available; the default behavior is to treat annotated methods, constructors, and fields as indicating required dependencies. This behavior can be changed as demonstrated below.

Types:- no , ByType , ByName , constructor

* Core dependency

Core and context

* Spring data
* Spring MVC

**Spring boot Anotation:-**

@SpringBootApplication

@Runwith

@EnableAutoConfiguartion

@path

@RequestMapping

@pathVariable

@RestContoller

@Contoller

[**CID:-**](CID:-)

@Name (Component)

@Inject (Autowire)

Question:- Difference between Spring singleton and GOF singleton

Spring :-

Spring is a framework use to receive benefit of lose coupling, standard architechcturing and easy to maintain large software solution. Spring core provide the solution to the problem with Inversion of control. It is a process that object resolve dependency with constructor argument set injection and argument with factory method.

Spring container :- Applicationcontext

Spring container is responsible managing object lifecycle which are define in configuration metadata.

For java configuration :-

AnnotationConfigApplicationContext ctx=new AnnotationConfigApplicationContext (Spring.class);

Close application context using try with resource

Spring Junit:-

@ContextConfiguration(classes=ConfigurationDetails)

@ContextConfiguration(locations="/applicationxontext.xml")

@RunWith(SpringJUnit4ClassRunner.class)

@Test

@Before :- Will run before every test

@BeforeClass:- Run only once very first time

@After:- will run after every test

@AfterClass:- will run only once super last

Spring JDBC :-

Datasource :- responsible for creating and managing connection

JDBC Template:- simplest jdbc template

Simple JDBC Template:-

NamedParameterJDbc Template