IOC : Inversion of Control

DI : Dependency Injection

Constructor base

Setter base

With configuration

Xml

Annotation

If object creation taken care by spring container by default spring container create singleton object.

Singleton is type of design pattern. If we want to create only one object of any class then we need to create singleton design pattern. Reference can be many but only one memory.

singleton : only one memory

prototype : each time new memory

setter base DI. If we want to achieve setter base di in pojo or java bean class we need to write setter method mandatory.

In parameterized constructor base DI order matter. We can't achieve partial dependency.

But in setter base DI order don't matter while doing DI. We can partial dependency.

Auto – wired :

By default spring framework do di for primitive property. But if our class contains user defined objects we need to do explicitly di using ref attribute.

With help of Auto wired we can avoid ref attribute. Means we can do DI for complex property using auto wired. If we set autowired as byType. Spring container automatically scan and inject the object.

If we use byType. In xml file we need to provide only one bean details of that type.

If it contains more than one bean details of that type. Then we need to use byName. In byName reference name and bean id name must be match.

DI using annotation

We need to write @Component annotation on pojo class. it is equal to <bean class="com.bean.Empoyee"></bean>

By default id name is class name in lower case if class contains one word.

If class name is Employee. Then id name is employee

If class name is EmployeeDetails. Then id name is employeeDetails. (came naming rules).

By default @Component annotation is not enable we need to enable using

Xml file or using another class with few annotation.

Beans.xml file to enable @Component annontation

```
<context:component-scan base-
package="com"></context:component-scan>
```

Spring framework provided

BeanFactory : core interface to pull the object from container.

ApplicationContext : pre defined interface which internally extends BeanFactory.

`AnnotationConfigApplicationContext`
Purely java base configuration.

1. XML Configuration
2. Partial xml configuration to enable annotation
3. Pure java base.

@Configuration annotation is equal to beans.xml file

@ComponentScan annotation is equal to `<context:component-scan base-package="com"></context:component-scan>`

Spring framework with JDBC concept with DataSource features. (In this example we are planning to improve Model layer).