**Beans Auto wiring / Beans Collaboration**

if we want to inject dependent bean objects to another bean object automatically without providing <property> / <contructor-arg> tags then we can use “Auto Wiring”.

This feature of spring fw will make the IOC Container to inject dependent objects to the bean objects automatically on the basis of the **property names** or on the basis of **property types** without checking <property> / <contructor-arg> tags.

There 4 ways to manage autowiring

1. Xml based.
2. Annotation based.
3. Auto-Discovery [Stereo Types]
4. Java Based

XML Based Auto Wiring:-

In this approach, if we want to provide autowiring in spring applications then we have to use “autowire” attribute in <bean> tag.

<bean id=”….” Class=”-----” autowire=”value” />

There are different modes of autowire attribute

1. no
   1. it is representing “no” autowiring for the beans injection, we must provide explicit configuration for the bean injection.
2. byName
   1. it will provide autowiring on the basis of the property names. Here IOC Container will search for dependent bean objects by matching bean property name with the identity values of the bean configuration in config file.
3. byType
   1. it will provide on the basis of property data type. IOC Container identify the dependent objects by matching property data type with the bean types[class attribute values] in bean configuration.
   2. Here config file should have only one bean definition must be existed with same type. If it is having multiple with same type it will raise an exception.
4. Constructor
   1. It is same as “byType” mode, byType mode for setter methods and “constructor” mode will provide constructor dependency injection on the basis of the types.

Class Address

{

String Hno, Street, City, State;

}

Class Account

{

String accNo, accName, accType;

Long balance;

}

Class Employee

{

String eid, ename;

Address eaddr;

Account eacc;

Void getEmpDetails()

{

}

}

Xml file

<beans>

<bean id=”eaddr” class=”----” >

-------

</bean>

<bean id=”eacc” class=”-----” >

------

</bean>

<bean id=”empbean” class=”-----“ autowire=”byName”>

</bean>

</beans>

**Annotation Based Auto-Wiring: -**

To implement auto wiring in spring applications without providing auto wiring configuration in config file, we have to use the following annotations.

@Required

@Autowried

@Qualifier

**@Required**

This annotation will make IOC Container to inject a particular bean object in another bean object is mandatory. We have to use this annotation at method level, i.e. just before setXXX() method. After providing it, if we are not providing respective bean injection, it will raise exception.

**@Autowried**

It is able to represent auto-wiring in bean classes. It will be used at method level, field level and local variable level in constructor dependency injection.

If we provide “required” argument with “false” value in @Autowired then it is not required to use @Requried

It follows “byType” autowired

**@Qualifier**

In the case of “byType” mode, it will raise exception if more than one bean annotations exist. To resolve this problem, we have to use “@Qualifier”. It will be used to specify a particular bean object among multiple object same type.

@Qualifier(“bean\_identity”)

Class Course

{

String cid, cname;

Int cost

}

Class student

{

String sid, sname;

Course scourse

@Autowired(required=true)

@Qualifier(“bean id”)

setScoure(….)

}

<beans xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-3.0.xsd"* >

<context:annotation-config />

</beans>

**3. Auto-Discovery (Stereo Types)**

It will provide the auto wiring bean objects without using <bean> configuration in configuration file.

To use this, we have to use the following annotations provided by spring framework in the package,

“org.springframework.stereotype”

1. @Component :- it will represent a component which is recognized by Spring Container.
2. @Repository :- it will represent a class as Model Drive, i.e DAO
3. @Service :- it will represent a class as Service Class
4. @Controller :- It will represent a class as Controller class, it will be used in Spring Web-MVC Module.

Note :- if you want to use these annotations, we have use following statement in config file

<context:component-scan base-package=”-----” />

Ex:<conext:component-scan base-package=”com.application.service” />

<conext:component-scan base-package=”com.application.dao” />

<conext:component-scan base-package=”com.application.controller” />

Steps

Step-1

Com.dto pack

Create AccountBean

Com.dao pack

Create AccountDao Interface

@Component("bean\_name")

Create AccountDaoImpl class and implements AccountDao

Com.ServicePack

Create AccountService Interface

@Service("bean\_name")

Create AccountServiceImpl class implements AccountService

{

@Autowired(required=**true**)

**private** AccountDao dao;

}

xml

<context:component-scan base-package=*"com.peit.service"* />

<context:component-scan base-package=*"com.peit.dao"* />

AccountService aServ = (AccountService)appfact.getBean("accService");