Autowiring in spring

B

A

Autowiring is nothing but injecting dependencies nut its being taken care spring, it will be done automatically.

In previous examples

<ref=”aref”>

Imp point 🡪 Autowiring is having one drawback that, it will be injecting only object. Not primitive and string values

Autowiring can be done using two approaches

1. XML base
2. Annotation
3. XML base autowiring 🡪
   1. ByName
   2. ByType
   3. Constructor
   4. Autodetect – deprecated

What we did in previous examples

<bean class=”com.blabla.A” name=”aref”>

<bean class=”com.blabla.B” name=”bref” >

<property name=”obj” ref=”aref” />

1. Annotation

We will have to use annotation for doing same.

@Autowired

@Qualifier 🡪 if there will be two bean we have created so at the time compilation compiler will get confuse which bean suppose to consider , in such scenarios we can use @Qualifier.

We will have to use @Qualifier along with @Autowired annotation

Sysntax for @Qualifier is

@Autowired

@Qualifier(“name\_of\_bean”)

Standalone Collection

I want to create a separate collection which I can use in any bean.

Example: how to create standalone collection.

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

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

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

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

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

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

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

*http://www.springframework.org/schema/beans/spring-beans-2.5.xsd*

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

*http://www.springframework.org/schema/aop/spring-aop-2.5.xsd*

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

*http://www.springframework.org/schema/tx/spring-tx-2.5.xsd*

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

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

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

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

<util:list list-class=*"java.util.Vector"* id=*"projectlist"*>

<value>Exam Management System</value>

<value>School Management System</value>

</util:list>

<util:map map-class=*"java.util.TreeMap"* id=*"dept\_details"*>

<entry key=*"4"* value=*"Service"*/>

<entry key=*"2"* value=*"Development"*/>

<entry key=*"1"* value=*"HR"*/>

</util:map>

<util:properties id=*"dbconnection"*>

<prop key=*"driver"*>com.microsoft.sqlserver.jdbc.SQLServerDriver</prop>

<prop key=*"username"*>sa</prop>

<prop key=*"password"*>password\_123</prop>

<prop key=*"instanceName"*>SQLSERVEREXPRESS2019</prop>

</util:properties>

We were creating spring applications and specifying beans in config.xml file

<Bean class=” ” name=””>

</Bean>

StereoType Annotation

@Component 🡪 to specify that it is a bean class we will have to use this annotation.

@Component

Class Employee{

}

If you are using simply

@Component then you have to be careful while getting bean by name.

Follow standard name as below🡪

Employee employee = new Employee();

Car car=new Car();

But if you don’t want to tension of this naming then simply specify name inside

@Component

Syntax is

@Component(“emp”).

For letting spring scan all packages in our application which are specified as

Components we will have to add

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

In our xml file.

@Value for setting values to properties of bean

Ex- @Value(“101”)

SpEl if there will be object type value

@Values(“#{projectlist}”)

Bean Scope

1. Singleton
2. Prototype
3. Request
4. Session
5. Globalsession

Singleton – Spring will create only one object and refer same every time

Prototype – Spring will create seprate objects for us.

Syntax for specifying scope.

@Scope("prototype")