Auto wiring The main objective of this is to automatically inject the secondary types. It's no applicable primitives.

We ll consider an banking example

A banking applicationn will have so many forms and fro each form there will a controrller attached to it so that it can accept the request and and pass the response back to user in order that it can be understandable by Users. For each Controller there might be Business class on which might be dependenet on likewise The business class might have the dependancy of DAO class which fetch the data from the database and produce the data to Business class.

Lets say there ara 100 Forms and 100 Controller classes are there for these form also there 100 Business and 100 DAO Classes in order to assist former in their execution.

our spring.xml file will be configured as follows

<DTD>

<beans>

<bean id="d1" class="beans.DAO">

</bean>

<bean id="b1" class="beans.Business">

<property name="dao" ref="d1">

</bean>

<bean id="c1" class="beans.Controller">

<property name="dao" ref="d1">

</bean>

<bean id="d2" class="beans.DAO">

</bean>

<bean id="b2" class="beans.Business">

<property name="dao" ref="d1">

</bean>

<bean id="c2" class="beans.Controller">

<property name="dao" ref="d1">

</bean>

//Here the above bean config we have to dove 98 more bean classes which will be tedious task for the developers

</beans>

if we use auto wiring property then we dont need to pass the refernece of the bean a taht a another is dependent on. IOC is designed to sense the object a class is dependenet on and will accordingly inject the same bean but that bean class property must be inside the bean class which is depending on it then only IOC can create the bean object.

Autowire attribute of bean tag (or can be used in beans tag as defalut-autowire) can accept following type of values

1. ByType Setter Injection the must have the setter method of the bean it is dependent on otherwise it will not inject the the same because when it will setter then only IOC will consider the parameter as dependent parameter. If IOC finds more than one object fro the same dependent property in the bean tag then ambiguity occurs as which to choose to inject in the bean class it is depenedent on and will throw beancreationException. In such case we need to set autowire-candiate of bean tag to be false which true by default. So the object whose autowire is true only those object is eligible for the autore candidate. ByType is the best approach bcz we dont have to remember all the reference types as becomes hectic task we remeber all these ref types.

2. ByName Setter Injection: If we define this in autowire then it will serach the reference in the bean matching the same Type and name and if both of these things are met then IOC will inject that reference in the bean object which is depending on it. It doesnt find the bean with same reference name a used in Bean property then no bean ref is eligible for injection. As Ids are unique So this type of autowiring is considered safe

3. Constructor Constructor D Injection is used when we have Constructor D Injection If there is Constructor of a bean class then IOC will search in the parameter which needs to be injected and then by parameter type it will inject the reference of the same bean. Internally it will do the same operation as that ByType Injection and thus will have the same issue as with ByType.

4. Autodetect Setter and Constructor D Injection after writing 100 Dao, Business and Controller classes we might not know which class has constructor and which class has setters for the its parameters. In such case this value of autowire is very useful. It'll automatically find the dependncies are in the form of setter or Constructors and hence it will find out the sane and accordingly inject the required dependencies.

5. No (Default)

<beans default autowire="ByType">

<bean id="d1" class="">

</bean>

<bean id="b1" class="">

</bean>

<bean id="c1" class="">

</bean>

//Same as above remaining 99 classes

<beans>

above spring.xml will inject the depencdencies in the beans if any automatically by checkinng the type of bean class it is dependent on and will inject the appropriate class.

Liecycle of dao classes

StereoType annotations also given in our J2EE container

1. @Named have same functinality as of @Component of spring

To load properties file data into the class in order to inject the dependency we use class whose name is propertyPlaeHolderConfigurer on object of which we have to invoke the mehod setLocation(Path of prop file). This method will load the file in the IOC container and IOC will get the data from properties file and store the same at context scope and we can use the expression to get the data from IOC context.

<beans>

<bean class="PropertyPlaceHolderConfigurer">

<property name="location" value="Path of properties File"/>

</beasn>

<bean id="cp" class="beans.ConnectionPool">

<property name="driver" value="${driver}"/>

<property name="url" value="${driver}"/>

<property name="userName" value="${userName}"/>

<property name="password" value="${password}"/>

</bean>

</beans>

I18N Support to users so that the product can be used by different peoples with different linguitic regions

L10N Support also needed to provided in order provide for I18N Support. here in l10n support we need to provide business support amd provide validations. For eaxmple we are buidling the application for Bank which can be used by India and US users. If the user from India wants depossit money then he can deposit in Rupees while user can deposit in US Dollars. For interest calculation business policy for India may differ from that of US. So in such case also we need to provide the appropriate business logic for both the countries to comply with their interest rate calculation standards.

Here in spring J2EE container applicationConetxt weill provide for I18N support. In core Java also I18N support is provided and can be implemented as follows.

Locale l = new Local("hi") Its onstructor must contain the language which you want to give support for

ResourceBundle rb = ResourceBundle.getResource("properties file", l) In its parameters we need to pass the fully eligible name of the properties file ie it path and the locale object we created before this line.

String value=rb.getString("key") for a specified string it will return the value the coressponding string

S.o.pln(value)

The same thing we can achieve in our AplplicationContext J2EE Container for Spring

Locale l = new Locale("hi")

ApplicationContext ap= new ClassPathXmlApplicationContext("spring.xml");

String value=ap.getMessage("key", null, l);

In spring.xml we have to configure the location of properties file by instantiating class called ResourceBundleMessage

<beans>

<bean id="messageSource" class="ResourceBundleMessage">

<property name="baseName" vlaue="Data file location"/>

</bean>

</beans>

In spring.xml file we compulsory need write the ref name fro the ResourcBundleMesage class as messageResource or spring will force us to write the name as above in order that application functions as intended

Spring MVC

1. Form backup Support: It will store the data of form in a bean class and we need to submit the data so that it can be validated and if any field in form is invlaid then it will populate the validated data and will remove the invlalid from the bean and will return the validation error against the invlaid fields. to get this support we need to use spring tag libraries.

2.Controller (MutltiAction Controller and MultiForm Controllers)

For example if we have 3 forms student registeration, student update and student search operations and all these opertion we want to execute with same controllers then we can write the controllers for the same and we can write the each event in single controller class and on button click the appropriate event will invoked.

3. Validation Support

4. I18N Support

5. Interceptor Support

6. ViewResolvers

7. ExceptionHandling

For MVC Application the container is WebApplicationContext.

Spring MVC is designed using jsp model architecture. As per Jsp model arhcitecture we have 4 models

3 form data

StuReg Controller

StuReg Form

Validation Read Data

RR

EmpReg Form

Login Bean

EmpReg Bean

StuReg Bean

Login Form

EmpReg Controller

Controllers

Login Controller