Spring Auto wiring:-

Autowiring is a concept of injecting a spring bean automatically without writing <ref/> tag by programmer. Programmer does not required to write explicitly Dependency Injection.

Autowiring can be configured in 2 ways :

1) XML based

2) Annotation Based

1) XML Based: XML based autowiring can be classified as

a) byName : compares the spring bean (java code) filed name (variable name) and configuration code (XML file) bean name (bean tag name) , if they are matched then automatically those objects will b bonded with each other using setter injection.

b) byType: It compares bean class variable Data type and spring bean class type. If both are matched then it will do setter injection.

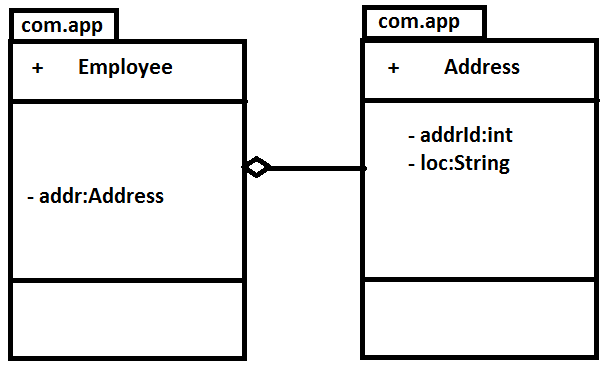
c) constructor: It check for parameterized constructor for creating object with reference type as parameter. If not found then uses default constructor at last. Always checks More number of parameters first, if not matched then next level less no of parameters constructor will be compared, and then goes on up to default constructor.

d) no : it will not do any auto wiring. It is disabled by default.

e) default (default value is no, even we can change this)

f) autodetect (works only in older versions like 2.X) : It works like byType if default constructor is available in spring bean, if not works like constructor if parameterized constructor is available.

ex: consider the below example for above concept. Employee class has a Address class Dependency.



Java code:

**package** com.app;

**public** **class** Address {

**private** **int** addrId;

**private** String loc;

**public** **int** getAddrId() {

**return** addrId;

}

**public** **void** setAddrId(**int** addrId) {

**this**.addrId = addrId;

}

**public** String getLoc() {

**return** loc;

}

**public** **void** setLoc(String loc) {

**this**.loc = loc;

}

@Override

**public** String toString() {

**return** "Address [addrId=" + addrId + ", loc=" + loc + "]";

}

}

=====

**package** com.app;

**public** **class** Employee{

**private** Address addr;

**public** Address getAddr() {

**return** addr;

}

**public** **void** setAddr(Address addr) {

**this**.addr = addr;

}

@Override

**public** String toString() {

**return** "Employee [addr=" + addr + "]";

}

}

======

code byType:

<?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: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.xsd*

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

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

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

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

*"*>

<bean class=*"com.app.Address"* name=*"addrObj"*>

<property name=*"addrId"* value=*"9856"*/>

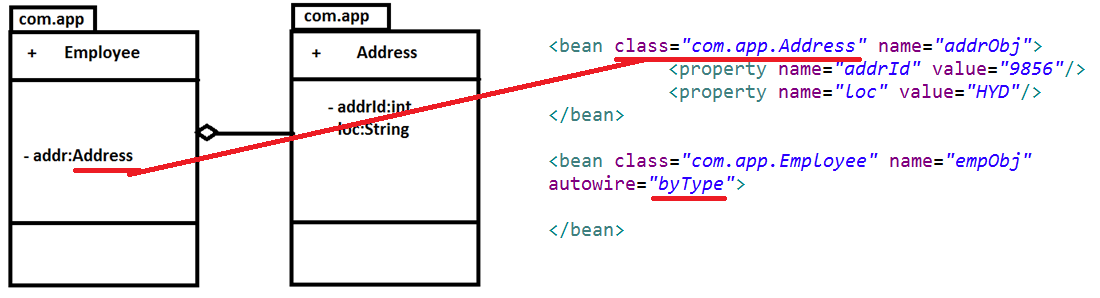
<property name=*"loc"* value=*"HYD"*/>

</bean>

<bean class=*"com.app.Employee"* name=*"empObj"* autowire=*"byType"*>

</bean>

</beans>



=========

code byName:

<?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: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.xsd*

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

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

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

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

*"*>

<bean class=*"com.app.Address"* name=*"addr"*>

<property name=*"addrId"* value=*"9856"*/>

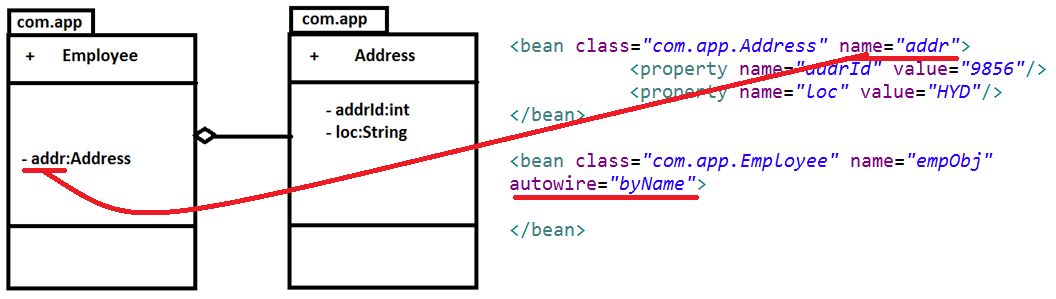
<property name=*"loc"* value=*"HYD"*/>

</bean>

<bean class=*"com.app.Employee"* name=*"empObj"* autowire=*"byName"*>

</bean>

</beans>



========

constructor:

to use this option class must have at least one parameterized constructor.

Always constructor with more parameters will be compared first , if not matched then it goes to next level more parameters constructor (like 4 parameters, 3 parameters ..) until zero/default constructor. Even If default constructor not found then spring container throws an exception saying that no constructor found.

Java Code:

**package** com.app;

**public** **class** Employee{

**private** Address addr;

**public** Employee() {

**super**();

System.*out*.println("In default");

}

**public** Employee(Address addr) {

**super**();

**this**.addr = addr;

System.*out*.println("in Param");

}

**public** Address getAddr() {

**return** addr;

}

**public** **void** setAddr(Address addr) {

**this**.addr = addr;

}

@Override

**public** String toString() {

**return** "Employee [addr=" + addr + "]";

}

}

====

**package** com.app;

**public** **class** Address {

**private** **int** addrId;

**private** String loc;

**public** **int** getAddrId() {

**return** addrId;

}

**public** **void** setAddrId(**int** addrId) {

**this**.addrId = addrId;

}

**public** String getLoc() {

**return** loc;

}

**public** **void** setLoc(String loc) {

**this**.loc = loc;

}

@Override

**public** String toString() {

**return** "Address [addrId=" + addrId + ", loc=" + loc + "]";

}

}

====

XML Code:

<?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: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.xsd*

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

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

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

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

*"*>

<bean class=*"com.app.Address"* name=*"addr"*>

<property name=*"addrId"* value=*"9856"*/>

<property name=*"loc"* value=*"HYD"*/>

</bean>

<bean class=*"com.app.Employee"* name=*"empObj"* autowire=*"constructor"*>

</bean>

</beans>

===========

**package** com.app;

**import** org.springframework.context.support.AbstractApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

AbstractApplicationContext context = **new** ClassPathXmlApplicationContext("config.xml");

Employee obj = (Employee)context.getBean("empObj");

System.*out*.println(obj);

}

}

======

Note: 1) If more than one matchi9ng found while doing constructor , then it compares names to inject the bean: for ex:

<?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: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.xsd*

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

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

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

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

*"*>

<bean class=*"com.app.Address"* name=*"addr"*>

<property name=*"addrId"* value=*"956"*/>

<property name=*"loc"* value=*"HYD"*/>

</bean>

<bean class=*"com.app.Address"* name=*"addr2"*>

<property name=*"addrId"* value=*"985"*/>

<property name=*"loc"* value=*"HYD1"*/>

</bean>

<bean class=*"com.app.Employee"* name=*"empObj"* autowire=*"constructor"*>

</bean>

</beans>

in the above code addr,addr2 are two objects then it compares the names to inject. So **addr** will be select to inject. If no name matched then no object will be injected.

=========

autodetect:

Java code:

**package** com.app;

**public** **class** Address {

**private** **int** addrId;

**private** String loc;

**public** **int** getAddrId() {

**return** addrId;

}

**public** **void** setAddrId(**int** addrId) {

**this**.addrId = addrId;

}

**public** String getLoc() {

**return** loc;

}

**public** **void** setLoc(String loc) {

**this**.loc = loc;

}

@Override

**public** String toString() {

**return** "Address [addrId=" + addrId + ", loc=" + loc + "]";

}

}

============

**package** com.app;

**public** **class** Employee{

**private** Address addr;

**public** Employee() {

**super**();

System.*out*.println("In default");

}

**public** Employee(Address addr) {

**super**();

**this**.addr = addr;

System.*out*.println("in Param");

}

**public** Address getAddr() {

**return** addr;

}

**public** **void** setAddr(Address addr) {

**this**.addr = addr;

System.*out*.println("in setter");

}

@Override

**public** String toString() {

**return** "Employee [addr=" + addr + "]";

}

}

===

<?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: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/util*

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

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

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

*"*>

<bean class=*"com.app.Address"* name=*"addr"*>

<property name=*"addrId"* value=*"9"*/>

<property name=*"loc"* value=*"HYD9"*/>

</bean>

<bean class=*"com.app.Employee"* name=*"empObj"* autowire=*"autodetect"*>

</bean>

</beans>

=====

**package** com.app;

**import** org.springframework.context.support.AbstractApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

AbstractApplicationContext context = **new** ClassPathXmlApplicationContext("config.xml");

Employee obj = (Employee)context.getBean("empObj");

System.*out*.println(obj);

}

}

Note: if you execute the above code it will check for default constructor and if found it performs byType Autowiring. If you comment default constructor then it uses parameterized constructor.

output:

In default

in setter

Employee [addr=Address [addrId=9, loc=HYD9]]

on comment of default constructor:

/\*public Employee() {

super();

System.*out*.println("In default");

}\*/

output:

in Param

Employee [addr=Address [addrId=9, loc=HYD9]]