Autowiring Modes

Spring supports the following autowiring modes:

**◦ no**: It’s the default autowiring mode. It means no autowiring.

**◦ byName**: The byName mode injects the object dependency according to name of the bean. In such case, property name and bean name should be same. It internally calls setter method.

**◦ byType**: The byType mode injects the object dependency according to type. So it can have different property name and bean name. It internally calls setter method.

**◦ constructor**: The constructor mode injects the dependency by calling the constructor of the class. It calls the constructor having large number of parameters.

**◦ autodetect**: In this mode, Spring first tries to autowire by constructor. If this fails, it tries to autowire by using byType.

1. Autowiring ‘no’:

This is a default autowiring mode. It means no autowiring.

1 <bean id="department" **class**="com.auto.wiring.Department">

2 <property name="deptName" value="Information Technology" />

3 </bean>

4 <bean id="employee" **class**="com.auto.wiring.Employee"></bean>

2. Autowiring ‘byName’:

This option enables autowire based on bean names. Spring looks up the configuration file for a matching bean name. If found, this bean is injected in the property. However, if no such bean is found, an error is raised.

In this case, the name of the department bean is same as the employee bean’s property (Department), so Spring will be autowired to it via the setter method – **setDepartment**(Department department).

1 <bean id="department" **class**="com.auto.wiring.Department">

2 <property name="deptName" value="Information Technology" />

3 </bean>

4 <bean id="employee" **class**="com.auto.wiring.Employee" autowire="byName"></bean>

3. Autowiring ‘byType’:

This option enables the autowire based on bean type. It searches property’s class type in configuration file. It injects the property, if such bean is found, otherwise an error is raised.

In this case, the data type of the department bean is same as the data type of the employee bean’s property (Department object), therefore Spring will autowire it via the setter method – **setDepartment**(Department department).

1 <bean id="department" **class**="om.auto.wiring.Department">

2 <property name="deptName" value="Information Technology" />

3 </bean>

4 <bean id="employee" **class**=“com.auto.wiring.Employee" autowire="byType"></bean>

4. Autowiring ‘constructor’

Autowiring by constructor is similar to byType but it applies to constructor arguments. It will look for the class type of constructor arguments, and then do an autowire byType on all constructor arguments. If exactly one bean of the constructor argument type is not present in the container, a fatal error will be raised.

The data type of department bean is same as the constructor argument data type in employee bean’s property (Department object). Therefore, Spring autowires it using the constructor method – **public** **Employee**(Department department).

1 <bean id="department" **class**="com.auto.wiring.Department">

2 <property name="deptName" value="Information Technology" />

3 </bean>

4 <bean id="employee" **class**="com.auto.wiring.Employee" autowire="constructor"></bean>

5. Autowiring ‘autodetect’

Autowiring by autodetect uses two modes i.e. constructor or byType modes. First, it will look for valid constructor with arguments. If it is found then the constructor mode is chosen. If there is no constructor defined in a bean, the autowire byType mode is chosen.

In following case, since there is a Department object in the Employee class, Spring autowires it using byType via the setter method – **setDepartment**(Department department).

1 <bean id="department" **class**="com.auto.wiring.Department">

2 <property name="deptName" value="Information Technology" />

3 </bean>

4 <bean id="employee" **class**="com.auto.wiring.Employee" autowire="autodetect"></bean>

**Note :** Autodetect functionality will work with the 2.5 and 2.0 schemas. It will not work from 3.0+.