### Need of dependency injection with example?

Earlier when one class is depend on another class .example: Person class is depend on Adhar class for getting Adhar card details.

Task: We want to call get getAdharDetails() method through person.

Earlier we used to create the Adhar card instance in Person class. In future if there is any modification in Adhar card like change of parameters. So constructor parameters also changes and we needed to modify the Adhar class as well as Person class.

To avoid this.

Dependency injection is used. We only create the reference of Adhar in Person class through interface. And we inject the dependency of Adhar in Person through Constructor or setter method. Now if there is change in Adhar card we don’t need to modify the Person class because we are not creating instance of Adhar class in Person class.

<bean id=*"adhar "* class=*"com.IOCConcepts.Adhar "*>

</bean>

<bean id=*"person "* class=*"com.IOCConcepts.Person "*>

<constructor-arg ref=*"adhar "*></constructor-arg>

</bean>

### Two ways to perform Dependency Injection in Spring framework

Spring framework provides two ways to inject dependency

* By Constructor
* By Setter method

# Constructor Injection

Steps:

1)Define dependency interface and class

2)Create constructor in your class for injection

3)Configure dependency injection in spring config file

<bean id=*"address"* class=*"com.constructorInjection.Address"*>

<constructor-arg value=*"Pune"*></constructor-arg>

<constructor-arg value=*"Maharashtra"*></constructor-arg>

</bean>

<bean id=*"person"* class=*"com.constructorInjection.Person"*>

<constructor-arg value=*"abc"* type=*"String"*></constructor-arg>

<constructor-arg >

<ref bean=*"address"* />

</constructor-arg>

</bean>

By default type of constructor-args is string

# Dependency Injection by setter method

We can inject the dependency by setter method also. The **<property>**subelement of **<bean>** is used for setter injection. Here we are going to inject

1)Define dependency interface and class

2)Create setter methods in your class for injection

3)Configure dependency injection in spring config file

Example 1

<bean id=*"adhar "* class=*"com.IOCConcepts.Adhar "*>

</bean>

<bean id=*"person "* class=*"com.IOCConcepts.Person "*>

<property name=”adharService ” ref=*"adhar "*></property >

</bean> setAdharService of Person class will get called

-------------------------------

Example 2

<bean id=*"add"* class=*"com.setterInjection.Address"*>

<property name=*"city"* value=*"Mumbai"*></property>

<property name=*"state"* value=*"MS"*></property>

</bean>

<bean id=*"student"* class=*"com.setterInjection.Student"*>

<property name=*"id"* value=*"1"*></property>

<property name=*"name"* value=*"shri"*></property>

<property name=*"address"* ref=*"add"*>

</property>

</bean>

Autowiring

## Autowiring Modes

There are many autowiring modes:

|  |  |  |
| --- | --- | --- |
| **No.** | **Mode** | **Description** |
| 1) | no | It is the default autowiring mode. It means no autowiring bydefault. |
| 2) | byName | The byName mode injects the object dependency according to name of the bean. In such case, property name and bean name must be same. It internally calls setter method. |
| 3) | byType | The byType mode injects the object dependency according to type. So property name and bean name can be different. It internally calls setter method. |
| 4) | constructor | The constructor mode injects the dependency by calling the constructor of the class. It calls the constructor having large number of parameters. |
| 5) | autodetect | It is deprecated since Spring 3. |

2)BYName

<bean id=*"emp"* class=*"com.autowiring.Employee"* autowire=*"byName"*>

<property name=*"name"* value=*"shri"*></property>

</bean>

ApplicationContext context=**new** ClassPathXmlApplicationContext("SpringConfig.xml");

Employee emp=(Employee) context.getBean("emp");;

System.***out***.println(emp);

If bean id and property id is not same then it will throw no such NoSuchBeanDefinitionException

2)ByType

In case of byType autowiring mode, bean id and reference name may be different. But there must be only one bean of a type otherwise it will throw

NoUniqueBeanDefinitionException

3)Field Dependency injection (It is by type)