Constructor Injection
==================

=> IOC container uses parameterized constructor to create target spring bean class obj and to assign/inject dependent spring bean class obj or values to target spring bean class obj

=> For this injection, we need to place <constructor-arg> tags for "n" times to use "n" param constructor for constructor injection while working with xml driven cfgs

=> For this injection, we need to place @Autowired on the top of parameterized constructor in annotation driven cfgs

=> In this injection, the spring bean class obj will always be created using parameterized constructor

=> This is called Fastest injection becoz the dependent values/objects will be injected to target spring bean class obj during the object creation process. (spring bean class obj)

=> In setter injection, the order of creating spring bean class obj are
a) target spring class class obj b) Dependent spring bean class obj
[IOC container calls target setXxx(-) having dependent obj as the arg value]

=> In constructor injection, the order of creating spring bean class objs are
a) Dependent spring bean class obj b) Target spring bean class obj
[IOC container uses parameterized constructor having dependent obj(s) as the arg values to create the target spring bean class obj]

Example app using constructor injection
====================================

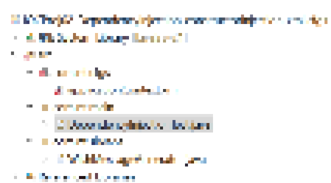
```
//WishMessageGenerator.java (Target spring bean class)
package com.nt.beans;

import java.time.LocalDate;
import java.time.LocalTime;

public class WishMessageGenerator {
    private LocalTime time; //HAS-A property
    private LocalDate date; //HAS-A property

    //parameterized constructor for constructor injection
    public WishMessageGenerator(LocalTime time, LocalDate date) {
        this.time=time;
        this.date=date;
        System.out.println("WishMessageGenerator: 2-param constructor");
    }

    //b.method
    public String showWishMessage(String user) {
        System.out.println("WishMessageGenerator.showWishMessage()" +time+"...."+date);
        //get current hour of the day
        int hour=time.getHour();
        //generate wish message based on current hour of the day
        if(hour<12)
            return "Good Morning:"+user;
        else if(hour<16)
            return "Good Afternoon:"+user;
        else if(hour<20)
            return "Good Evening:"+user;
        else
            return "Good night:"+user;
    }
}
```



applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

   <!-- Dependent spring bean cfgs -->
   <bean id="ltime" class="java.time.LocalTime" factory-method="now" />
   <bean id="ldate" class="java.time.LocalDate" factory-method="now"/>
   <!-- calls LocalTime.now() method to create LocalTime class obj (having system time) -->

   <!-- Target spring bean class cfg -->
   <bean id="wmg" class="com.nt.beans.WishMessageGenerator">
       <constructor-arg name="time" ref="ltime"/>
       <constructor-arg name="date" ref="ldate"/>    } constructor injection cfgs
   </bean>
                    constructor      Dependent
                    param name       spring bean id
</beans>
```

Dependency Injection Test.java (main class / Client App)

```
//DependencyInjectionTest.java
package com.nt.main;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.nt.beans.WishMessageGenerator;

public class DependencyInjectionTest {
```

Constructor Injection

========

=========

=> IOC container uses parameterized constructor to create target spring bean class obj and to assign/ inject dependent spring bean class objs or values to target spring bean class obj

=> For this injection, we need to place <constructor-arg> tags for "n" times to use "n" for constructor Injection while working with xml driven cfgs

param constructor

=> For this injection, we need to place @Autowired on the top of parameterized constructor in annotation driven cfgs

=> In this injection, the spring bean class obj will always be created using parameterized constructor

=> This is called Fastest injection becoz the dependent values/objects will be injected to target spring bean class obj during the object creation process.

(spring bean class obj)

=> In setter Injection, the order of creating spring bean class objs are

a) target spring class class obj b) Dependent spring bean class obj

calls

[IOC containerrget.setXxx(-) having dependent obj as the arg value]

=> In constructor Injection, the order of creating spring bean class objs are a) Dependent spring bean class obj b) Target spring bean class obj

[IOC container uses parameterized constructor having dependent obj(s) as the arg values to create the target spring bean class obj]

Example app using constructor Injection

=======:

===

//WishMessageGenerator.java (Target spring bean class) package com.nt.sbeans;

import java.time.LocalDate;

import java.time.LocalTime;

public class Wish MessageGenerator {

private LocalTime time; //HAS-A property

private LocalDate date; //HAS-A property

IOCProj02-DependencyInjection-constructorInjection-xml cfgs

// parameterized constructor for constructor Injection

public WishMessageGenerator(LocalTime time, LocalDate date) {

}

this.time-time;

this.date=date;

System.out.println("Wish MessageGenerator:: 2-param constructor");

> JRE System Library [JavaSE-21]

src

com.nt.cfgs

applicationContext.xml

✓com.nt.main

DependencyInjectionTest.java

com.nt.sbeans

> WishMessageGenerator.java Referenced Libraries

```
//b.method
public String showWishMessage(String user) {
System.out.println("Wish MessageGenerator.showWishMessage()::"+time+"....."+date);
// get current hour of the day
int hour-time.getHour();
//generate wish message based on current hour of the day
if(hour<12)
return "Good Morning::"+user;
else if(hour<16)
return "Good Afternoon:"+user;
else if(hour<20)
return "Good Evening :"+user;
else
return "Good Night:"+user;
}
}
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
<!-- Dependent spring bean cfg -->
<bean id="ltime" class="java.time.LocalTime" factory-method="now" />
<bean id="ldate" class="java.time.LocalDate" factory-method="now"/>
<!-- calls LocalTime.now() method to create LocalTime class obj having system time -->
<!-- Target spring bean class cfg -->
<bean id="wmg" class="com.nt.sbeans.WishMessageGenerator">
<constructor-arg name="time" ref="ltime"/>
```

<constructor-arg name="date" ref="ldate"/>

constructor Injection cfgs

</bean>

</beans>

constructor Dependent param name spring bean id

DependencyInjection Test.java (main class/ Client App)

==========

```
//DependencyInjectionTest.java
package com.nt.main;
import org.springframework.context.support.FileSystemXmlApplicationContext;
import com.nt.sbeans.WishMessageGenerator;
public class DependencyInjection Test {
public static void main(String[] args) {
System.out.println("DependencyInjection Test.main()");
//crete IOC Container
FileSystemXmlApplicationContext ctx=
new FileSystemXmlApplicationContext("src/com/nt/cfgs/applicationContext.xml");
//get Target Spring bean class obj ref
Object obj=ctx.getBean("wmg");
WishMessageGenerator generator=(WishMessageGenerator)obj;
//invoke the b.method
String result-generator.showWishMessage("raja");
System.out.println("Result is ::"+result);
//close the IOC cotnainer
ctx.close();
System.out.println("DependencyInjection Test.main()");
Spring Bean cfg file
}
```

=>For target spring bean, if want to inject another spring bean class obj using xml driven cfgs there we need to use "ref" attribute of <property> tag or <constructor-arg> tag.

=>For Dependent spring bean, if want to inject simple values/wrapper values/String values using xml driven cfgs there we need to use "value" attribute of <property> tag or <constructor-arg> tag.

=> A spring bean class can have the following types of bean properties

a) HAS -A properties (Object type/ Ref type properties)

b) Simple type properties (primitive datatype/wrapper data type/String type properties)

c) Array type properties

d) Collection type properties

Q) if we configure both setter Injection and constructor injection to the same property of spring bean class

can u tell me which injection value will be taken as the final value?

Ans) Since the setter Injection takes place after the constructor injection we can say

setter injection value overrides the constructor injection value

```
<!-- Target spring bean class cfg -->
<bean id="wmg" class="com.nt.sbeans.WishMessageGenerator"> <constructor-arg name="time" ref="ltime"/>
<constructor-arg name="date" ref="ldate"/> <property name="name" value="mukesh"/> <constructor-arg name="name" value="raja"/>
</bean>
//Injects mukesh as the final value(setter Injection)
to the "name" property
```

What Circular Injection or Cyclic Injection in spring programming?

Ans) if two spring beans are dependent to each other then we can say there in cyclic injection or Circular Injection (A is dependent to B and B is dependent to A) =>This is not possible with constructor Injection (applicaton goes to inifinite loop)

and possible with setter Injection

Sample code

```
==========
//A.java
package com.nt.sbeans;
public class A{
private B b;
public void setB(B b) {
System.out.println("A.setB()");
this.b=b;
}
/* public A(B b) {
System.out.println("A::1-param constructor");
this.b=b;
}*/
//toString()
public String toString() {
return "from A spring bean";
}
}
//B.java
package com.nt.sbeans;
public class B {
```

```
private A a;
public void setA(A a) {
System.out.println("B:: setA(-)");
this.a=a;
}
//toString()
public String toString() {
return "From B spring bean";
}
}
```

applicationContext.xml

```
<?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:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
<!-- spring beans cfg -->
<bean id="a1" class="com.nt.sbeans.A">
<property name="b" ref="b1"/>
</bean>
<bean id="b1" class="com.nt.sbeans.B">
<property name="a" ref="a1"/>
</bean>
</beans>
```

Client App/ Main class

==========

```
//CyclicInjection Test.java
package com.nt.main;
import org.springframework.context.support.FileSystemXmlApplicationContext;
import com.nt.sbeans.A;
public class CyclicInjection Test {
public static void main(String[] args) {
//create IOC container
FileSystemXmlApplicationContext ctx=
new FileSystemXmlApplicationContext("src/com/nt/cfgs/applicationContext.xml");
// get spring bean class obj ref
A a1 =(A)ctx.getBean("a1");
```

System.out.println(a1);

//close the container ctx.close();

}

}

Few Eclipse short cut keys

==========

ctrl+shift+o: To import the packages

ctrl+shift+c :: To enable or disable single line commenting (toggle key)

ctrl+shfit+ / To enable multiline commenting

alt+shift+s,r ---> For setters and getters

ctrl+shfit+:: To disable multiline commenting

systrace +ctrl+space :: System.out.println(-) with default message

alt+shift+s,o ---> for parameterized constructors alt+shift+s,s ----> for toString()

sysout +ctrl+space :: System.out.println(-) with no message

ctrl+space :: For getting possible suggestions

and etc..

How Disable block commenting problem (which disturbs the structure of the code) while working multi-line commenting?

Ans) window menu--->preferences ---> search for formatter ---> new formatter name :: f1 --->

go to comments section ---> enable ---> deselect the following checkbox

▾ Comments

Maximum width for comments:

✓Count width from comment's starting position

Enable Javadoc comment formatting

Enable block comment formatting