***OFFLINE DTD***

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN"

"http://www.springframework.org/dtd/spring-beans-2.0.dtd">

src/main/java/ ....................... (java classes)

src/main/resources/ .................. (some .properties files )

src/main/webapp/ ..................... (web content, web.xml )

src/main/webapp/spring/ .............. (other spring config files, app-confix.xml, etc.)

src/main/webapp/myApp-servlet.xml .... (servlet config)

src/test/java ........................ (java test classes)

src/test/resources ................... (some test .properties files)

***This Example Dependency in the form of Primitive and String based value***

**WelcomeBean.Java**

**Ex :: 1**

**package** p1;

**public** **class** WelcomeBean {

**private** String msg;

**public** **void** setMsg(String msg) {

**this**.msg = msg;

}

**public** **void** show(){

System.***out***.println("Message is " + msg);

}

}

**Spring.xml**

<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*

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

[*http://www.springframework.org/schema/context/spring-context.xsd*](http://www.springframework.org/schema/context/spring-context.xsd)*"*>

<bean id=*"one"* class=*"p1.WelcomeBean"*>

<property name=*"msg"* value=*"Welocme to Java"*>

</property>

</bean>

</beans>

**Mainc.java**

**public** **class** Mainc {

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

Resource res = **new** ClassPathResource("Spring.xml");

BeanFactory f = **new** XmlBeanFactory(res);

Object o = f.getBean("one");

WelcomeBean wb = (WelcomeBean)o;

wb.show(); }}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Dependency in the form of Objective***

**EX :: 2**

**BeanOne.java**

**package** p2;

**public** **class** BeanOne {

BeanTwo b;

**public** **void** setB(BeanTwo b) {

**this**.b = b;

}

**public** **void** show(){

b.speed();

}

}

**BeanTwo.java**

**public** **class** BeanTwo {

**public** **void** speed(){

System.***out***.println("In a BeanTwo");

}

}

**Spring.xml**

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"one"* class=*"p2.BeanOne"*>

<property name=*"b"*>

<bean class=*"p2.BeanTwo"*></bean>

</property>

</bean>

</beans>

**Client.java**

**public** **class** Client {

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

Resource res = **new** ClassPathResource("Spring.xml");

BeanFactory f = **new** XmlBeanFactory(res);

Object o = f.getBean("one");

BeanOne bt = (BeanOne)o;

bt.show();

}

}

**Spring.xml**

**Ex :: 3**

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"one"* class=*"p2.BeanOne"*>

<property name=*"b"*>

<bean class=*"p2.BeanTwo"*></bean>

</property>

</bean>

<bean id=*"two"* class=*"p2.BeanThree"*>

<property name=*"b"*>

<bean class=*"p2.BeanTwo"*></bean>

</property>

</bean>

</beans>

**BeanOne.java**

**public** **class** BeanOne {

BeanTwo b;

**public** **void** setB(BeanTwo b) {

**this**.b = b;

}

**public** **void** show(){

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

b.speed();

}

}

**BeanTwo.java**

**public** **class** BeanTwo {

**public** **void** speed(){

System.***out***.println("In a BeanTwo");

}

}

**BeanThree.java**

**public** **class** BeanThree {

BeanTwo b;

**public** **void** setB(BeanTwo b) {

**this**.b = b;

}

**public** **void** show1(){

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

b.speed();

}

}

**Client.java**

Resource res = **new** ClassPathResource("Spring.xml");

BeanFactory f = **new** XmlBeanFactory(res);

Object o = f.getBean("one");

BeanOne bt = (BeanOne)o;

bt.show();

Object o1 = f.getBean("two");

BeanThree bt1 = (BeanThree)o1;

bt1.show1();

**Spring.xml**

**Ex :: 4**

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"one"* class=*"p2.BeanOne"*>

<property name=*"b"*>

<ref local=*"three"*/>

<!-- <bean class="p2.BeanTwo"></bean> -->

</property>

</bean>

<bean id=*"two"* class=*"p2.BeanThree"*>

<property name=*"b"*>

<!-- <bean class="p2.BeanTwo"></bean> -->

<ref local=*"three"*/>

</property>

</bean>

<bean id=*"three"* class=*"p2.BeanTwo"*></bean>

</beans>

Like this we can place objective bean any number of time and it will not gonna create seperate for others

***Interface Journey***

**Ex :: 5**

**public** **interface** Journey {

**public** **void** startJourney();

//POJI

}

***Interface Vehicle***

**public** **interface** Vehicle {

**public** **void** move();

//POJI

}

***Class Car***

**public** **class** Car **implements** Vehicle{

**private** String fuelType;

**private** **int** maxSpeed;

**public** **void** setFuelType(String fuelType) {

**this**.fuelType = fuelType;

}

**public** **void** setMaxSpeed(**int** maxSpeed) {

**this**.maxSpeed = maxSpeed;

}

@Override

**public** **void** move() {

System.***out***.println("FuelType " + fuelType);

System.***out***.println("MaxSpeed " + maxSpeed);

}

//The Combination of POJI and POJO we called them as Spring Beans

//One Interface and One Class Combination we called them as Spring Beans

}

***Class MotorBike***

**public** **class** MotorBike **implements** Vehicle{

**private** **int** maxSpeed;

**public** **void** setMaxSpeed(**int** maxSpeed) {

**this**.maxSpeed = maxSpeed;

}

@Override

**public** **void** move() {

System.***out***.println("FuelType : Petrol");

System.***out***.println("MaxSpeed " + maxSpeed);

}

}

***Class Traveller***

**public** **class** Traveller **implements** Journey{

Vehicle v ;

**public** Vehicle getV() {

**return** v;

}

**public** **void** setV(Vehicle v) {

**this**.v = v;

}

@Override

**public** **void** startJourney() {

// **TODO** Auto-generated method stub

System.***out***.println("Journey is Started");

v.move();

***Spring.xml***

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"car"* class=*"p1.Car"*>

<property name=*"fuelType"* value=*"Diesel"*></property>

<property name=*"maxSpeed"* value=*"100"*></property>

<!-- Internally Spring will convert this 100 into int with Integer.parseInt(); -->

</bean>

<bean id=*"motor"* class=*"p1.MotorBike"*>

<property name=*"maxSpeed"* value=*"200"*></property>

</bean>

</beans>

***Spring2.xml***

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"tra"* class=*"p1.Traveller"*>

<property name=*"v"*>

<ref parent=*"car"*/>

</property>

</bean>

</beans>

***Client.Java***

Resource res = **new** ClassPathResource("Spring.xml");

BeanFactory ba = **new** XmlBeanFactory(res);//Parent Container

Resource res1 = **new** ClassPathResource("Spring2.xml");

BeanFactory ba1 = **new** XmlBeanFactory(res1,ba);//Child Container

/\*if we want the ID of parent container in child then we have

to pass new XmlBeanFactory(res1,ba);object of ba like this\*/

Object o = ba1.getBean("tra");

Traveller t = (Traveller)o;

t.startJourney();

/\*Internally first car class object will get created because(see Spring2.xml)

first inner bean will get chance and then outer bean

so first setter method of car class get called and all the instance variable of car class

get Initialied then Traveller class variable\*/

/\*Internally

Car c = new Car();

c.setFuelType("Diesel");

c.setMaxSpeed(100);

Traveller tt = new Traveller();

t.setV(c);\*/

}

}

**EX : 6**

***SampleBean.java***

**public** **class** SampleBean {

**private** Set data;

**public** Set getData() {

**return** data;

}

**public** **void** setData(Set data) {

**this**.data = data;

}

**public** **void** printData(){

Iterator it = data.iterator();

**while**(it.hasNext()){

System.***out***.println(it.next());

***Spring.xml***

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"one"* class=*"p1.SampleBean"*>

<property name=*"data"*>

<set>

<value>1001</value>

<value>Sam</value>

<value>HYD</value>

</set></property>

</bean>

</beans>

***Client.java***

Resource res = **new** ClassPathResource("Spring.xml");

BeanFactory b = **new** XmlBeanFactory(res);

Object o = b.getBean("one");

SampleBean s = (SampleBean)o;

s.printData();

***SampleBean.java***

**Ex : 7**

**public** **class** SampleBean {

**private** Map<String, Integer> data;

**public** **void** setData(Map<String, Integer> data) {

**this**.data = data;

}

**public** **void** printData(){

Set s = data.entrySet();

Iterator it = s.iterator();

**while**(it.hasNext()){

Map.Entry<String, Integer> m = (Map.Entry<String, Integer>)it.next();

System.***out***.println(m.getKey() + " " + m.getValue());

***Spring.xml***

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id = *"one"* class=*"p1.SampleBean"*>

<property name=*"data"*>

<map *key-type*=*"java.lang.Integer"* value-type=*"java.lang.String"*>

<entry key=*"1"* value=*"A"*></entry>

<entry key=*"2"* value=*"A"*></entry>

<entry key=*"3"* value=*"A"*></entry>

<entry key=*"4"* value=*"A"*></entry>

</map>

</property>

</bean>

</beans>

***Client.java***

**public** **class** Mainc {

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

Resource res = **new** ClassPathResource("Spring.xml");

BeanFactory b = **new** XmlBeanFactory(res);

Object o = (Object)b.getBean("one");

SampleBean s = (SampleBean)o;

s.printData();

}

}

***SampleBean.java***

**Ex : 8**

**public** **class** SampleBean {

**private** String name;

**public** SampleBean(String name) {

**this**.name = name;

}

**public** **void** display(){

System.***out***.println("Name is " + **this**.name);

}}

***Spring.xml***

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"one"* class=*"p1.SampleBean"*>

<constructor-arg>

<value>Welcome to my Const</value>

</constructor-arg>

</bean>

</beans>

***Client.java***

**public** **class** Mainc {

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

Resource res = **new** ClassPathResource("Spring.xml");

BeanFactory b = **new** XmlBeanFactory(res);

SampleBean s = (SampleBean)b.getBean("one");

s.display();

}

}

***SampleBean.java***

**Ex : 9**

**public** **class** SampleBean {

**private** **int** id;

**private** String name;

**public** SampleBean(String name) {

**this**.name = name;

}

**public** SampleBean(**int** id, String name) {

**this**.id = id;

**this**.name = name;

}

**public** **void** display(){

System.***out***.println("Name is " + **this**.name + "ID is "+ **this**.id);

}

}

***Spring.xml***

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"one"* class=*"p1.SampleBean"*>

<constructor-arg>

<value>Welcome to my Const</value>

</constructor-arg>

</bean>

<bean id=*"two"* class=*"p1.SampleBean"*>

<constructor-arg value=*"1"*></constructor-arg>

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

</bean>

</beans>

***Client.java***

**public** **class** Mainc {

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

Resource res = **new** ClassPathResource("Spring.xml");

BeanFactory b = **new** XmlBeanFactory(res);

SampleBean s = (SampleBean)b.getBean("two");

s.display();

}

}

**Ex:10**

***A.java***

**public** **class** A {

B b;

**public** A(B b) {

**this**.b = b;

}

}

***B.java***

**public** **class** B {

A a;

**public** B(A a) {

**this**.a = a;

}

}

***Spring.xml***

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"one"* class=*"p1.A"*>

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

</bean>

<bean id=*"two"* class=*"p1.B"*>

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

</bean>

<!-- This is called Circular dependencies problem because A class req B class obj

and B class req A class obj -->

</beans>

***Outer.java***

**Ex : 11**

**public** **class** Outer {

Inner i;

**public** **void** setI(Inner i) {

**this**.i = i;

}

}

***Inner.java***

**public** **class** Inner {

}

***Spring.xml***

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"one"* class=*"p1.Outer"* autowire=*"byName"*></bean>

<bean id=*"i"* class=*"p1.Inner"*></bean>

<!-- Inner bean id and ref in outer bean should be same -->

</beans>

***Client.java***

**public** **class** Mainc {

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

Resource res = **new** ClassPathResource("Spring.xml");

BeanFactory b = **new** XmlBeanFactory(res);

Outer o = (Outer)b.getBean("one");

}

}

**Spring Auto Wire by Constructor Start Here**

Outer.java

**public** **class** Outer {

Inner i;

**public** Outer(Inner i) {

**this**.i = i;

}

**public** **void** show(){

i.meth();

}

}

Inner.java

**public** **class** Inner {

**public** **void** meth(){

System.***out***.println("Hello Inner");

}

}

Spring.xml

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"one"* class=*"p1.Outer"* autowire=*"constructor"*></bean>

<bean id=*"i"* class=*"p1.Inner"*></bean>

</beans>

Client.java

**public** **class** Client {

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

Resource res = **new** ClassPathResource("Spring.xml");

BeanFactory bf = **new** XmlBeanFactory(res);

Outer o = (Outer)bf.getBean("one");

o.show();

}

}

***Student.java***

***Ex : 12***

**public** **class** Student {

**private** Integer age;

**private** String name;

**public** Integer getAge() {

**return** age;

}

**public** **void** setAge(Integer age) {

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

***Profile.java***

public class Profile {

@Autowired

@Qualifier("s1")

Student s;

public void printAge(){

System.out.println(s.getAge());

}

public void printName(){

System.out.println(s.getName());

**Spring.xml**

<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-3.0.xsd*

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

*http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>

<context:annotation-config/>

<bean id=*"one"* class=*"p1.Profile"*></bean>

<bean id=*"s1"* class=*"p1.Student"*>

<property name=*"age"* value=*"25"*></property>

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

</bean>

<bean id=*"s2"* class=*"p1.Student"*>

<property name=*"age"* value=*"45"*></property>

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

</bean>

</beans>

**Client.java**

**public** **class** Client {

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

ApplicationContext cont = **new** ClassPathXmlApplicationContext("Spring.xml");

Profile p = (Profile)cont.getBean("two");

p.printAge();

p.printName();

}

}

**Spring @Qualifier with constructor**

Student.java

**public** **class** Student {

Integer age;

String name;

**public** Integer getAge() {

**return** age;

}

**public** **void** setAge(Integer age) {

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

Profile.java

**public** **class** Profile {

Student s;

@Autowired

**public** Profile(@Qualifier("s1") Student s) {

**this**.s = s;

}

**public** Profile() {

System.***out***.println("Inside Profile");

}

**public** **void** printAge(){

System.***out***.println(s.age);

}

**public** **void** printName(){

System.***out***.println(s.name);

Spring.xml

<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-3.0.xsd*

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

*http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>

<context:annotation-config></context:annotation-config>

<bean id=*"s1"* class=*"p1.Student"*>

<property name=*"age"* value=*"25"*></property>

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

</bean>

<bean id=*"s2"* class=*"p1.Student"*>

<property name=*"age"* value=*"206"*></property>

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

</bean>

<bean id=*"p"* class=*"p1.Profile"*></bean>

</beans>

Client.java

ApplicationContext cont = **new** ClassPathXmlApplicationContext("Spring.xml");

Profile p = (Profile)cont.getBean("p");

p.printAge();

p.printName();

**@Primary example**

Car.java

@Component

**@Primary**

**public** **class** Car **implements** Vehicle{

@Override

**public** **void** move() {

System.***out***.println("From Car");

}

}

Motor.java

@Component

**public** **class** Motor **implements** Vehicle{

@Override

**public** **void** move() {

System.***out***.println("From Motor");

}

}

Traveller.java

@Component

**public** **class** Traveller {

@Autowired

Vehicle v;

**public** **void** startJouney() {

System.***out***.println("startJouney....");

**this**.v.move();

}

}

To Resolve Ambiguity issue we can use **@Primary** Annotation

**@Autowired at Collections** (Continuing Above Example)

Traveller.java

@Component

**public** **class** Traveller {

@Autowired

List<Vehicle> vehicles;

**public** **void** startJouney() {

**this**.vehicles.forEach(x -> {

x.move();

});

**@Resource(from JSR)**

Example

If we have defined 2 beans for same class in Spring.xml file so for to resolve ambiguity issue we use **@Qualifier** with **@Autowired**

Now instead to two annotation we can use single annotation called **@Resource** its internally works on byName concept.

EX...

**import** javax.annotation.Resource;

@Resource(name = "emp")

Employee e;

If we don’t define (name = "emp") default it will search byNAme

***Student.java***

**Ex:13**

**public** **class** Student {

**int** sid;

String sname;

**public** **void** setSid(**int** sid) {

**this**.sid = sid;

}

**public** **void** setSname(String sname) {

**this**.sname = sname;

***Spring.xml***

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN"

"http://www.springframework.org/dtd/spring-beans-2.0.dtd">

<beans>

<bean id=*"one"* class=*"p1.Student"* dependency-check=*"simple"*>

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

<property name=*"sname"* value=*"Sam"*></property>

</bean>

</beans>

***Client.java***

**public** **class** Mainc {

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

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

Resource res = **new** ClassPathResource("Spring.xml");

BeanFactory f = **new** XmlBeanFactory(res);

Student s = (Student)f.getBean("one");

System.***out***.println(s.sid);

System.***out***.println(s.sname);

}

}

***Student.java***

***Ex:14***

**public** **class** Student {

**private** Integer age;

**private** String name;

@Required

**public** **void** setAge(Integer age) {

**this**.age = age;

}

**public** Integer getAge() {

**return** age;

}

@Required

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getName() {

**return** name;

***Spring.xml***

<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-3.0.xsd*

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

*http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>

<context:annotation-config/>

<!-- Definition for student bean -->

<bean id = *"one"* class = *"p1.Student"*>

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

<!-- try without passing age and check the result -->

<!-- property name = "age" value = "11"-->

</bean>

</beans>

**@Required in Spring will not work in 5.1.4 to make it work add this line**

**<bean class=*"org.springframework.beans.factory.annotation.RequiredAnnotationBeanPostProcessor"*/>**

***------------------------------------------------------***

***CustomerService.java***

***Ex:15***

**public** **class** CustomerService **implements** InitializingBean,DisposableBean{

String msg;

**public** String getMsg() {

**return** msg;

}

**public** **void** setMsg(String msg) {

**this**.msg = msg;

}

@Override

**public** **void** destroy() **throws** Exception {

System.***out***.println("Destry");

}

@Override

**public** **void** afterPropertiesSet() **throws** Exception {

System.***out***.println("After");

***Spring.xml***

<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-2.5.xsd"*>

<bean name=*"one"* class=*"p2.CustomerService"*>

<property name=*"msg"* value=*"SAM"*></property>

</bean>

</beans>

***Client.java***

**public** **class** Client {

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

ApplicationContext app = **new** ClassPathXmlApplicationContext("Spring.xml");

CustomerService c = (CustomerService)app.getBean("one");

System.***out***.println(c.msg);

But after this it will not call destry method in order to do that we have to use one abstract class called ConfigurableApplicationContext.

ConfigurableApplicationContext app = **new** ClassPathXmlApplicationContext("Spring.xml");

CustomerService c = (CustomerService)app.getBean("one");

System.***out***.println(c.msg);

app.close();//this will shutdown container

}

}

***Ex:16***

***CustomerService.java***

**public** **class** CustomerService {

String msg;

**public** String getMsg() {

**return** msg;

}

**public** **void** setMsg(String msg) {

**this**.msg = msg;

}

**public** **void** destroy1() {

System.***out***.println("Destry");

}

**public** **void** init1() {

System.***out***.println("After");

***Spring.xml***

<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-2.5.xsd"*>

<bean name=*"one"* class=*"p2.CustomerService"* init-method=*"init1"* destroy-method=*"destroy1"*>

<property name=*"msg"* value=*"SAM"*></property>

</bean>

</beans>

***Client.java***

**public** **class** Mainc {

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

ConfigurableApplicationContext app = **new** ClassPathXmlApplicationContext("Spring.xml");

CustomerService c = (CustomerService)app.getBean("one");

System.***out***.println(c.msg);

app.close();

}

}

***Now if we don’t want to use that tag we can go with Annotations just make changes in CustomerService.java and remove that tag in Spring.Xml file***

***CustomerService.java***

**public** **class** CustomerService {

String msg;

**public** String getMsg() {

**return** msg;

}

**public** **void** setMsg(String msg) {

**this**.msg = msg;

}

@PreDestroy

**public** **void** destroy1() {

System.***out***.println("Destry");

}

@PostConstruct

**public** **void** init1() {

System.***out***.println("After");

}}