**Spring Bean**

In this chapter, we will discuss Spring Bean. While reading the previous chapter you have a basic idea what a Spring Bean is.

Spring bean is something which is managed by Spring container and it is nothing but a POJO.

But wait, many of the people using POJO and Spring bean interchangeably.

But my question is, are POJO and Spring bean same?

Let understand it by an Example.

**package** com.example.bean;

**public** **class** Car {

**private** String carName;

**private** Owner carOwner;

**public** String getCarName() {

**return** carName;

}

**public** **void** setCarName(String carName) {

**this**.carName = carName;

}

**public** Owner getCarOwner() {

**return** carOwner;

}

**public** **void** setCarOwner(Owner carOwner) {

**this**.carOwner = carOwner;

}

@Override

**public** String toString() {

**return** "Car [carName=" + carName + ", carOwner=" + carOwner + "]";

}

}

**package** com.example.bean;

**public** **class** Owner {

**private** String name;

**private** String state;

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

@Override

**public** String toString() {

**return** "Owner [name=" + name + ", state=" + state + "]";

}

}

**package** com.example.bean;

**import** org.springframework.context.ApplicationContext;

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

**import** com.example.hello.HelloWorld;

**public** **class** ApplicationMain {

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

ApplicationContext ctx = **new** ClassPathXmlApplicationContext("configFiles/SpringBean.xml");

Car car =(Car) ctx.getBean("car");

System.***out***.println("Before Setting Owner " + car);

// Try to inject Spring Bean Owner into Car

Owner owner =(Owner) ctx.getBean("owner");

car.setCarOwner(owner);

System.***out***.println("After Spring Bean Injection " + car);

car.setCarOwner(**new** Owner());// setting a Pojo in to car

System.***out***.println("After POJO Injection " + car);

}

}

SpringBean.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-3.0.xsd"*>

<bean id=*"car"* class=*"com.example.bean.Car"*>

<property name=*"carName"* value=*"Santro"*/>

</bean>

<bean id=*"owner"* class=*"com.example.bean.Owner"*>

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

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

</bean>

</beans>

Output :

Before Setting Owner Car [carName=Santro, carOwner=null]

After Spring Bean Injection Car [carName=Santro, carOwner=Owner [name=Shamik, state=WestBengal]]

After POJO Injection Car [carName=Santro, carOwner=Owner [name=null, state=null]]

Lot of things going in the application,

We try to create an Application which will print the Car owner name and state, so we will create two POJO class

1. Owner

2. Car

And did a composition (HAS-A) between car and Owner.

Now we register them in Spring Configuration xml (SpringBean.xml),

But intentionally we do not inject car’s carOwner property , as we try to inject them by own

So first we try to get Owner bean from Spring Container (ApplicationContext) the set it to car, when printing the car we can see

All information, car name and owner details provide in Spring configuration. So Spring Container creates Owner bean and injects name, state properties.

After that we create an Owner POJO using java new keyword, then set it to car and print the same

But here we observe Owner information is null Spring does not inject Owner name and state.

Why?

Because Spring container does not consider this POJO (created by java new keyword) as Spring bean.

SpringContainer only considers those POJO as spring bean which are register in the container. To get this Spring bean always we have to use

ctx.getBean(“id”) call.

Conclusion: A POJO is considered as Spring Bean if and only if it is registered in Spring Container.

So please never be so clever, inject a bean by new keyword. If you do so, you are not using Spring features then it is better not to use Spring.

As I said we intentionally skip the injection of Owner property.

Now we will modify the program so injection can be done at the first place.

Modify the SpringBean.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-3.0.xsd"*>

<bean id=*"car"* class=*"com.example.bean.Car"*>

<property name=*"carName"* value=*"Santro"*/>

<property name=*"carOwner"* ref=*"owner"*/>

</bean>

<bean id=*"owner"* class=*"com.example.bean.Owner"*>

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

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

</bean>

</beans>

**Java code:**

package com.example.bean;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.example.hello.HelloWorld;

public class ApplicationMain {

public static void main(String[] args) {

ApplicationContext ctx = new ClassPathXmlApplicationContext("configFiles/SpringBean.xml");

Car car =(Car) ctx.getBean("car");

System.out.println("Setting Owner " + car);

// Try to inject Spring Bean Owner into Car

/\* Owner owner =(Owner) ctx.getBean("owner");

car.setCarOwner(owner);

System.out.println("After Spring Bean Injection " + car);

car.setCarOwner(new Owner());// setting a Pojo in to car

System.out.println("After POJO Injection " + car);\*/

}

}

Output:

Before Setting Owner Car [carName=Santro, carOwner=Owner [name=Shamik, state=WestBengal]]

Here, we use **ref** attribute of property and provide Owner id to inject Owner bean into Car**.**

**Bean important properties**

|  |  |
| --- | --- |
| **Properties** | **Description** |
| class | This attribute is mandatory and It’s takes the FQDN (fully qualified domain name) of class. |
| name | This attribute specifies the bean name uniquely. |
| scope | This attribute specifies the scope of the it should be singleton or prototype etc. |
| constructor-arg | This is used to inject the constructor based dependencies. |
| properties | This is used to inject the dependencies into java attributes using Spring Configuration file. |
| autowiring mode | This is used to inject the dependencies by auto wiring. |
| lazy-initialization mode | A lazy-initialized bean tells the Spring container to create a bean instance when it is first requested, rather than at startup. |
| initialization method | A callback to be called just after all necessary properties on the bean have been set by the container. It will be used for custom operation after initialization. |
| destruction method | A callback to be used when the container containing the bean is destroyed. |