En en tutorial de [Introducción a Spring](http://acodigo.blogspot.com/2017/02/spring-tutorial-de-introduccion.html) vimos como crear el ApplicationContext a partir de una configuración XML, no entraremos en detalles, el código seria el siguiente:

ApplicationContext ctx = new ClassPathXmlApplicationContext("beans.xml");

### Creación de beans con XML

La etiqueta utilizada para crear beans es <bean  id="…" class="…" /> para cada uno de ellos debemos definir un nombre o identificador usando el atributo **id**, también requerimos indicar el nombre de clase completo que define el tipo del bean con el atributo **class**.gi

**<bean** id="mybean"

class="carmelo.spring.xml.MyBean"**/>**

Para este ejemplo el nombre del bean es mybean, el tipo está determinado por la clase MyBean que se define como sigue:

**package** **carmelo.spring.xml**;

**public** **class** **MyBean** {

**public** MyBean() {

}

**public** MyBean(Integer valor, String mensaje) {

**this**.valor = valor;

**this**.mensaje = mensaje;

}

**private** Integer valor;

**private** String mensaje;

**public** Integer getValor() {

**return** valor;

}

**public** void setValor(Integer valor) {

**this**.valor = valor;

}

**public** String getMensaje() {

**return** mensaje;

}

**public** void setMensaje(String mensaje) {

**this**.mensaje = mensaje;

}

}

Al crear un bean usando el código anterior estaremos usando el constructor por defecto, el vacío, si no estuviera disponible tendríamos un error, pero si deseamos instanciar el bean usando el segundo constructor, pues podemos usar <constructor-arg value="…" /> para indicar cada uno de los argumentos del constructor, por ejemplo:

**<bean** id="mybean" class="carmelo.spring.xml.MyBean"**>**

**<constructor-arg** value="150" **/>**

**<constructor-arg** value="Este es un mensaje..." **/>**

**</bean>**

Los argumentos deben estar en el orden adecuado y ser del tipo correcto.

No necesariamente necesitamos el constructor para inicializar los campos de la clase, podemos hacerlo también mediante <property name="…" value="…" /> donde **name** indica el nombre y **value** el valor que le asignaremos.

**<bean** id="mybean" class="carmelo.spring.xml.MyBean"**>**

**<property** name="valor" value="150"**/>**

**<property** name="mensaje" value="Este es un mensaje..."**/>**

**</bean>**

Es posible simplificar el uso de constructor-arg y property por medio de los esquemas **c** y **p** respectivamente, funciona igual solo que se escribe de una manera mas corta, por ejemplo: c:mensaje="…" para establecer el parámetro mensaje del constructor, recordemos que al usar **c** debemos indicar todos los parámetros requeridos por el constructor, si tenemos varios se utilizará el que se adapte a los datos establecidos.

**<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"

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

xmlns:c="http://www.springframework.org/schema/c"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd"**>**

**<bean** id="mybean"

class="carmelo.spring.xml.MyBean"

p:valor="25" p:mensaje="Este es un mensaje..." **/>**

**</beans>**

En algunas ocasiones será necesario hacer referencia a otros beans, para ejemplificarlo tenemos la clase OtherBean, la cual almacena un objeto MyBean, esta clase se define del siguiente modo:

**public** **class** **OtherBean** {

**private** MyBean myBean;

**public** OtherBean() {

}

**public** OtherBean(MyBean myBean) {

**this**.myBean = myBean;

}

**public** MyBean getMyBean() {

**return** myBean;

}

**public** void setMyBean(MyBean myBean) {

**this**.myBean = myBean;

}

}

Usando los namespaces **c** o **p** solo debemos agregar -ref después del nombre del campo e indicar cual es el nombre o id del bean al que hacemos referencia.

**<bean** id="otherbean"

class="carmelo.spring.xml.OtherBean"

p:myBean-ref="mybean" **/>**

Podemos hacerlo sin utilizar los namespaces solo debemos cambiar value por ref lo demás funciona igual, por ejemplo:

**<bean** id="otherbean" class="carmelo.spring.xml.OtherBean"**>**

*<!-- inyectar por constructor -->*

*<!--<constructor-arg ref="mybean" />-->*

*<!-- inyectar por setter -->*

**<property** name="myBean" ref="mybean"**/>**

**</bean>**

Lo que acabamos de ver se le conoce en la tecnología Spring como Inyección de Dependencias (DI) en sus dos modalidades, la inyección por constructor y la inyección por métodos setter.

Aun hay muchas cosas mas que ver sobre la configuración con XML, puedes ver la continuación directa de este tutorial en la siguiente página: [Spring Configuración XML parte II](http://acodigo.blogspot.com/2017/02/spring-configuracion-xml-parte-ii.html).

Código en GitHub: [Spring Configuración XML](https://github.com/TutorProgramacion/tutoriales-spring/tree/master/tutorial_config_xml)