Spring bean – XML config

Last Updated: October 1, 2022

By: [Lokesh Gupta](https://howtodoinjava.com/author/lokeshgupta1981/)

[Spring 5 Core](https://howtodoinjava.com/spring5/core/)

[Spring Basics](https://howtodoinjava.com/tag/spring-basics/), [Spring Beans](https://howtodoinjava.com/tag/spring-beans/), [Spring Context](https://howtodoinjava.com/tag/spring-context/)

In this **spring bean XML configuration example**, learn to create define and create spring beans and populate application context in any spring application. This example uses **xml config to define beans**. We will use maven to manage the *spring dependencies* and eclipse to build and run the code.

**1. Spring maven dependencies**

To create spring application context, which is capable of creating and managing beans, we need minimum three maven dependencies i.e. ***spring-core*, *spring-beans* and *spring-context***.

1. **Spring-core** module has most basic classes required to work with other spring modules.
2. **Spring-beans** module provides org.springframework.beans.factory.BeanFactory interface which is required to work with spring beans.
3. **Spring-context** module provides org.springframework.context.ApplicationContext interface which enables additional features such as [message resources](https://howtodoinjava.com/spring-mvc/spring-mvc-internationalization-i18n-and-localization-i10n-example/), [AOP](https://howtodoinjava.com/spring-aop-tutorial/) capabilities, specific types of application contexts and [bean lifecycle](https://howtodoinjava.com/spring-core/spring-bean-life-cycle/) event listeners.

|  |
| --- |
| <**project** xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd;    <modelVersion>4.0.0</**modelVersion**>      <**groupId**>com.howtodoinjava.spring.webmvc</**groupId**>    <**artifactId**>SpringDemos</**artifactId**>    <**version**>0.0.1-SNAPSHOT</**version**>    <**packaging**>jar</**packaging**>      <**name**>SpringDemos</**name**>    <**url**>http://maven.apache.org</**url**>      <**properties**>      <**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>      <**spring.version**>5.2.0.RELEASE</**spring.version**>    </**properties**>      <**dependencies**>      <!-- Spring Dependencies -->      <**dependency**>        <**groupId**>org.springframework</**groupId**>        <**artifactId**>spring-core</**artifactId**>        <**version**>${spring.version}</**version**>      </**dependency**>      <**dependency**>        <**groupId**>org.springframework</**groupId**>        <**artifactId**>spring-beans</**artifactId**>        <**version**>${spring.version}</**version**>      </**dependency**>      <**dependency**>        <**groupId**>org.springframework</**groupId**>        <**artifactId**>spring-context</**artifactId**>        <**version**>${spring.version}</**version**>      </**dependency**>    </**dependencies**>  </**project**> |

**2. Spring beans definition in xml config**

**2.1. Single configuration file with bean definitions**

You can define all **spring beans** and their *transitive dependencies* in single xml file. This xml file can be used to **create application context**.

AD

|  |
| --- |
| <?**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    http://www.springframework.org/schema/context/    http://www.springframework.org/schema/context/spring-context.xsd">      <**bean** id="operations"  class="com.howtodoinjava.spring.beans.Operations"></**bean**>    <**bean** id="employee"  class="com.howtodoinjava.spring.beans.Employee"></**bean**>    <**bean** id="department"  class="com.howtodoinjava.spring.beans.Department"></**bean**>    </**beans**> |

**2.2. Define beans in multiple configuration files and import into main file**

This method is more useful in writing **modular code**. You can define beans in separate xml files and import the files into main xml file.

|  |
| --- |
| <**beans**>      <**bean** id="employee" class="com.howtodoinjava.spring.beans.Employee"></**bean**>    </**beans**> |
| <**beans**>      <**bean** id="department" class="com.howtodoinjava.spring.beans.Department"></**bean**>    </**beans**> |

|  |
| --- |
| <?**xml** version="1.0" encoding="UTF-8"?>  <**beans**>      <**import** resource="employee.xml"/>    <**import** resource="department.xml"/>      <**bean** id="operations" class="com.howtodoinjava.spring.beans.Operations"></**bean**>    </**beans**> |

**3. Spring bean example**

To **create**[**ApplicationContext**](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/ApplicationContext.html), we can use it’s one of specific implementation from a list of available implementations e.g. ClassPathXmlApplicationContext, FileSystemXmlApplicationContext, StaticApplicationContext, XmlWebApplicationContext etc.

We will need to pass the **bean configuration file name as constructor parameter** of the used class. Do not forget to file in classpath or [resources folder](https://howtodoinjava.com/java/io/read-file-from-resources-folder/).

|  |
| --- |
| **import** org.springframework.context.ApplicationContext;  **import** org.springframework.context.support.ClassPathXmlApplicationContext;    **public** **class** XmlConfigExample  {  **public** **static** **void** main( String[] args )      {    @SuppressWarnings("resource")    ApplicationContext ctx = **new**                    ClassPathXmlApplicationContext( "com/howtodoinjava/core/demo/beans/beans.xml" );            Employee employee = ctx.getBean(Employee.**class**);            Department department = ctx.getBean(Department.**class**);            Operations operations = ctx.getBean(Operations.**class**);            System.out.println(department);          System.out.println(employee);            operations.helloWorld();      }  } |

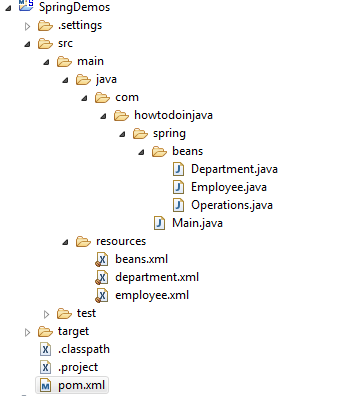
Program output:

AD

|  |
| --- |
| Jan 02, 2018 3:10:27 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions  INFO: Loading XML bean definitions from **class** path resource [beans.xml]    Jan 02, 2018 3:10:27 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions  INFO: Loading XML bean definitions from **class** path resource [employee.xml]    Jan 02, 2018 3:10:27 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions  INFO: Loading XML bean definitions from **class** path resource [department.xml]    Employee [id=0, name=**null**]  Department [id=0, name=**null**]  Hello World !! |

**4. Project structure and other classes used in example**

**4.1. Project Structure**

Spring XML Config Project Structure

**4.2. Bean Classes**

|  |
| --- |
| @Getter  @Setter  @ToString  **public** **class** Employee  {  **private** **long** id;  **private** String name;  } |
| @Getter  @Setter  @ToString  **public** **class** Department  {  **private** **long** id;  **private** String name;  } |

|  |
| --- |
| **public** **class** Operations  {  **public** **void** helloWorld(){      System.out.println("Hello World !!");    }  } |

Drop me your questions related to **Spring XML configuration** based **spring containers** in comments section.