(/)

# Guide to the Spring BeanFactory

Last modified: April 18, 2019

by baeldung (https://www.baeldung.com/author/baeldung/)

**Spring (https://www.baeldung.com/category/spring/) +**



If you have a few years of experience in the Java ecosystem, and you're interested in sharing that experience with the community (and getting paid for your work of course), have a look at the "Write for Us" page (/contribution-guidelines). Cheers. Eugen

## 1. Introduction

This article will focus on **exploring the Spring BeanFactory API**.

BeanFactory (http://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/beans/factory/BeanFactory.html) interface provides a simple, yet flexible configuration mechanism to manage objects of any nature via the Spring IoC container. Let's have a look at some basics before diving deep into this central Spring API.

## 2. Basics – Beans and Containers

Simply put, beans are the java objects which form the backbone of a Spring application and are managed by Spring IoC container. Other than being managed by the container, there is nothing special about a bean (in all other respects it's one of many objects in the application).

The Spring container is responsible for instantiating, configuring, and assembling the beans. The container gets its information on what objects to instantiate, configure, and manage by reading configuration metadata we define for the application.

## 3. Maven Dependencies

Let's add the required Maven dependency (https://search.maven.org/classic/#search%7Cga%7C1%7Cg%3A%22org.springframework%22%20AND%20a%3A%22spring-beans%22) to the *pom.xml* file. We will be using Spring Beans dependency to set up the BeanFactory:

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-beans</artifactId>
    <version>5.1.4.RELEASE</version>
</dependency>
```

# 4. The *BeanFactory* Interface

It's interesting to start by having a look at the interface definition (https://github.com/spring-projects/spring-framework/blob/master/spring-beans/src/main/java/org/springframework/beans/factory/BeanFactory.java) in *org.springframework.beans.factory* package and discuss some of its important APIs here.

## 4.1. The *getBean()* APIs

Various versions of ***`getBean()`*** *(https://docs.spring.io/spring/docs/5.0.x/javadoc-api/org/springframework/beans/factory/BeanFactory.html#getBean-java.lang.String-)* method return an instance of the specified bean, which may be shared or independent across the application.

## 4.2. The *containsBean()* API

This method confirms if this bean factory contains a bean with the given name. More specifically, it confirms if the *getBean(java.lang.String)* (https://docs.spring.io/spring/docs/5.0.x/javadoc-api/org/springframework/beans/factory/BeanFactory.html#getBean-java.lang.String-) able to obtain a bean instance with the given name.

## 4.3. The *isSingleton()* API

The *isSingleton* API can be used to query if this bean is a shared singleton. That is if *getBean(java.lang.String)* (https://docs.spring.io/spring/docs/5.0.x/javadoc-api/org/springframework/beans/factory/BeanFactory.html#getBean-java.lang.String-) will always return the same instance.

## 4.4. The *isPrototype()* API

This API will confirm if *getBean(java.lang.String)* (https://docs.spring.io/spring/docs/5.0.x/javadoc-api/org/springframework/beans/factory/BeanFactory.html#getBean-java.lang.String-) returns independent instances – meaning a bean configured with the prototype scope, or not.

The important thing to note is this method returning *false* does not clearly indicate a singleton object. It indicates non-independent instances, which may correspond to other scopes as well.

We need to use the *isSingleton(java.lang.String)* (https://docs.spring.io/spring/docs/5.0.x/javadoc-api/org/springframework/beans/factory/BeanFactory.html#isSingleton-java.lang.String-) operation to explicitly check for a shared singleton instance.

## 4.5. Other APIs

While the *isTypeMatch(String name, Class targetType)* method checks whether the bean with the given name matches the specified type, *getType(String name)* is useful in identifying the type of the bean with the given name.

Finally, *getAliases(String name)* return the aliases for the given bean name, if any.

# 5. *BeanFactory* API

*BeanFactory* holds bean definitions and instantiates them whenever asked for by the client application – which means:

- It takes care of the lifecycle of a bean by instantiating it and calling appropriate destruction methods
- It is capable of creating associations between dependent object while instantiating them
- It is important to point that *BeanFactory* does not support the Annotation-based dependency Injection whereas *ApplicationContext*, a superset of BeanFactory does

Do have a read on understanding Application Context (https://spring.io/understanding/application-context) to find out what Application Context can do extra.

# 6. Defining the Bean

Let's define a simple bean:

```
1  public class Employee {
2      private String name;
3      private int age;
4
5      // standard constructors, getters and setters
6  }
```

# 7. Configuring the *BeanFactory* with XML

We can configure the *BeanFactory* with XML. Let's create a file *bean factory-example.xml:*

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans (http://www.springframework.org/schema
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance (http://www.w3.org/2001/XMLSchema-instanc
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

  <bean id="employee" class="com.baeldung.beanfactory.Employee">
    <constructor-arg name="name" value="Hello! My name is Java"/>
    <constructor-arg name="age" value="18"/>
  </bean>
  <alias name="employee" alias="empalias"/>
</beans>
```

Note that, we have created an alias for the *employee* bean.

## 8. *BeanFactory* with *ClassPathResource*

*ClassPathResource* belongs to the *org.springframework.core.io* package. Let's run a quick test and initialize *XmlBeanFactory* using *ClassPathResource* as shown below:

```
public class BeanFactoryWithClassPathResourceTest {

    @Test
    public void createBeanFactoryAndCheckEmployeeBean() {
        Resource res = new ClassPathResource("beanfactory-example.xml");
        BeanFactory factory = new XmlBeanFactory(res);
        Employee emp = (Employee) factory.getBean("employee");

        assertTrue(factory.isSingleton("employee"));
        assertTrue(factory.getBean("employee") instanceof Employee);
        assertTrue(factory.isTypeMatch("employee", Employee.class));
        assertTrue(factory.getAliases("employee").length > 0);
    }
}
```

# 9. Conclusion

In this quick article, we learned about the main methods Spring *BeanFactory* API offers and an example to illustrate the configuration and its usage.

The code backing these examples is all available over on the GitHub project (https://github.com/eugenp/tutorials/tree/master/spring-core/src/test/java/com/baeldung/beanfactory).


Learning to build your API

**with Spring**?

## CATEGORIES

SPRING (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
REST (HTTPS://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SECURITY (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCE (HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSON/)
HTTP CLIENT (HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTTP/)
KOTLIN (HTTPS://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

## SERIES

JAVA “BACK TO BASICS” TUTORIAL (/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (/JACKSON)

HTTPCLIENT 4 TUTORIAL (/HTTPCLIENT-GUIDE)
REST WITH SPRING TUTORIAL (/REST-WITH-SPRING-SERIES)
SPRING PERSISTENCE TUTORIAL (/PERSISTENCE-WITH-SPRING-SERIES)
SECURITY WITH SPRING (/SECURITY-SPRING)

## ABOUT

ABOUT BAELDUNG (/ABOUT)
THE COURSES (HTTPS://COURSES.BAELDUNG.COM)
CONSULTING WORK (/CONSULTING)
META BAELDUNG (HTTP://META.BAELDUNG.COM/)
THE FULL ARCHIVE (/FULL_ARCHIVE)
WRITE FOR BAELDUNG (/CONTRIBUTION-GUIDELINES)
EDITORS (/EDITORS)
OUR PARTNERS (/PARTNERS)
ADVERTISE ON BAELDUNG (/ADVERTISE)

TERMS OF SERVICE (/TERMS-OF-SERVICE)
PRIVACY POLICY (/PRIVACY-POLICY)
COMPANY INFO (/BAELDUNG-COMPANY-INFO)
CONTACT (/CONTACT)