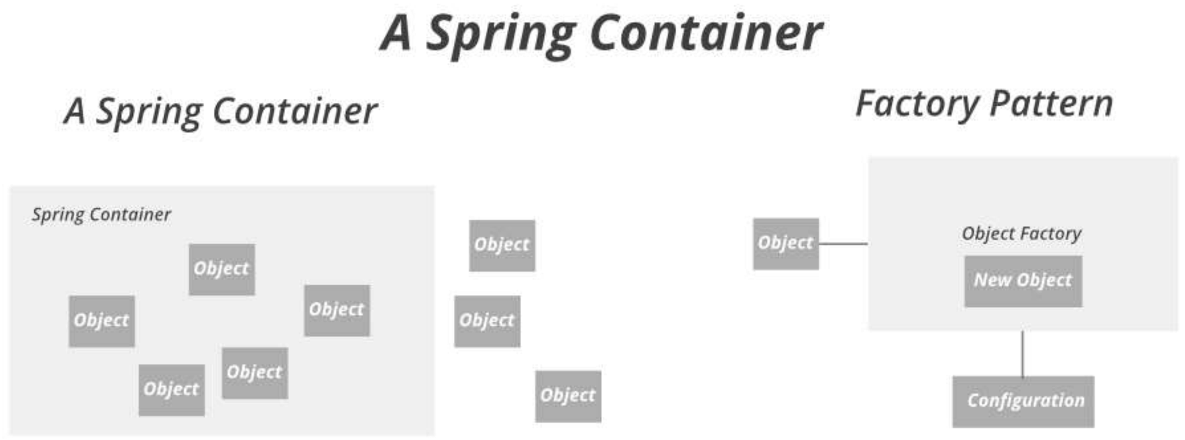
Spring – BeanFactory

G Ganeshchowdharysadanala

The first foremost thing when we talk about spring is dependency injection which is possible because spring is actually a container and behaves as a factory of Beans. Just like the BeanFactory interface is the simplest container providing an advanced configuration mechanism to instantiate, configure and manage the life cycle of beans. Beans are java objects that are configured at run-time by Spring IoC Container. BeanFactory represents a basic **IoC container** which is a parent interface of **ApplicationContext. BeanFactory** uses Beans and their dependencies metadata to create and configure them at run-time. BeanFactory loads the bean definitions and dependency amongst the beans based on a configuration file(XML) or the beans can be directly returned when required using Java Configuration. There are other types of configuration files like LDAP, RDMS, properties file, etc. BeanFactory does not support Annotation-based configuration whereas ApplicationContext does.



Let us do first go through some of the methods of Bean factory before landing up on implementation which are shown below in tabular format below as follows:

Method	Description
containsBean(String name)	Does this bean factory contain a bean definition or externally registered singleton instance with the given name?
getAliases(String name)	Return the aliases for the given bean name, if any.
getBean(Class<T> requiredType)	Return the bean instance that uniquely matches the given object type, if any.
getBean(Class<T> requiredType, Object… args)	Return an instance, which may be shared or independent, of the specified bean.
getBean(String name)	Return an instance, which may be shared or independent, of the specified bean.
getBean(String name, Class<T> requiredType)	Return an instance, which may be shared or independent, of the specified bean.
getBean(String name, Object… args)	Return an instance, which may be shared or independent, of the specified bean.
getBeanProvider(Class<T> requiredType)	Return a provider for the specified bean, allowing for lazy on-demand retrieval of instances, including availability and uniqueness options.
getBeanProvider(ResolvableType requiredType)	Return a provider for the specified bean, allowing for lazy on-demand retrieval of instances, including availability and uniqueness options.
getType(String name)	Determine the type of the bean with the given name.
getType(String name, boolean	Determine the type of the bean with the given

Method	Description
isPrototype(String name)	Is this bean a prototype? That is, will getBean(java.lang.String) always return independent instances?
isSingleton(String name)	Is this bean a shared singleton? That is, will getBean(java.lang.String) always return the same instance?
isTypeMatch(String name, Class<?> typeToMatch)	Check whether the bean with the given name matches the specified type.
isTypeMatch(String name, ResolvableType typeToMatch)	Check whether the bean with the given name matches the specified type.

Procedure:

1. Creating a Spring project using start.spring.io.
2. Creating a POJO class.
3. Configure the Student bean in the **bean-factory-demo.xml** file.
4. Writing it to application class.

Implementation:

Step 1: Bean Definition: Create a Student POJO class.

```
// Java Program where we are creating a POJO class
```

```
public class Student {

  // Member variables
  private String name;
  private String age;

  // Constructor 1
  public Student() {
  }

  // Constructor 2
  public Student(String name, String age) {
    this.name = name;
    this.age = age;
  }

  // Method inside POJO class
  @Override
  public String toString() {

    // Print student class attributes
    return "Student{" + "name='" + name + '\'' + ", age='" + age + '\'' +
'}';
  }
}
```

Step 2: XML Bean Configuration: Configure the Student bean in the ***bean-factory-demo.xml*** file.

```
<?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
            https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="student" class = "com.gfg.demo.domain.Student">
                <constructor-arg name="name" value="Tina"/>
                <constructor-arg name="age" value="21"/>
        </bean>
```

Step 3: Main Class

```
// Application class
@SpringBootApplication

// Main class
public class DemoApplication {

  // Main driver method
  public static void main(String[] args) {

    // Creating object in a spring container (Beans)
    BeanFactory factory = new ClassPathXmlApplicationContext("bean-factory-demo.xml");
    Student student = (Student) factory.getBean("student");

    System.out.println(student);
  }
}
```

Output:

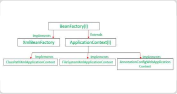
```
Student{name='Tina', age='21'}
```

> ***Note: XmlBeanFactory class*** *is deprecated.*

Last Updated : 02 Sep, 2022

7

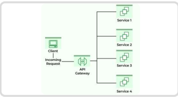
Similar Reads



Spring - Difference Between BeanFactory and ApplicationContext

Spring Boot - Spring JDBC vs Spring Data JDBC

Difference Between Spring DAO vs Spring ORM vs Spring JDBC

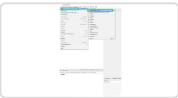


Java Spring Boot Microservices - Develop API Gateway Using Spring Cloud Gateway

Spring Boot | How to access database using Spring Data JPA

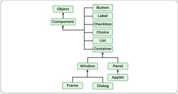
Difference between Spring MVC and Spring Boot

Spring - Add User Name and Password in Spring Security



How to Create and Setup Spring Boot Project in Spring Tool Suite?

Related Tutorials



Java AWT Tutorial

Spring MVC Tutorial

Spring Tutorial

Spring Boot Tutorial

Java 8 Features - Complete Tutorial

Previous Next

Article Contributed By :

G

Ganeshchowdharysadanala

Ganeshchowdharysadanala

Vote for difficulty

Current difficulty : Basic

Easy Normal Medium Hard Expert

Improved By : surindertarika1234, mitalibhola94

Article Tags : Java-Spring, Java

Practice Tags : Java

Improve Article Report Issue

GeeksforGeeks

A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

About Us

Legal

Careers

In Media

Contact Us

Advertise with us

Explore

Job-A-Thon Hiring Challenge

Hack-A-Thon

GfG Weekly Contest

Offline Classes (Delhi/NCR)

DSA in JAVA/C++

Master System Design

Master CP

Languages

Python

Java

C++

PHP

GoLang

DSA Concepts

Data Structures

Arrays

Strings

Linked List

Algorithms

Top 50 Graph Problems

Top 50 Array Problems

Top 50 String Problems

Top 50 DP Problems

Top 15 Websites for CP

Interview Corner

Company Wise Preparation

Preparation for SDE

Experienced Interviews

Internship Interviews

Competitive Programming

Aptitude Preparation

Commerce

Accountancy

Business Studies

Economics

Management

Income Tax

Finance

Statistics for Economics

SSC/ BANKING

SSC CGL Syllabus

SBI PO Syllabus

SBI Clerk Syllabus

IBPS PO Syllabus

IBPS Clerk Syllabus

Aptitude Questions

SSC CGL Practice Papers

Scalability in SD

Databases in SD

High Level Design or HLD

Low Level Design or LLD

Top SD Interview Questions

GfG School

CBSE Notes for Class 8

CBSE Notes for Class 9

CBSE Notes for Class 10

CBSE Notes for Class 11

CBSE Notes for Class 12

English Grammar

UPSC

Polity Notes

Geography Notes

History Notes

Science and Technology Notes

Economics Notes

Important Topics in Ethics

UPSC Previous Year Papers

Write & Earn

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships