Spring Bean

Spring Bean

Bean is an object in Spring that is managed by the Spring IoC Container. Spring creates bean with the configuration metadata that we have supplied in the <bean> tag of the XML file. We provide metadata to the IoC container either by using the XML file or by Java annotations.

During metadata configuration, we provide bean definitions with some optional attributes such as:

- The fully qualified name of Bean class name. such as com.tonight.community.Reader.
- Bean behavior such as Bean scope, lifecycle callback, etc.
- Bean dependencies (references to other beans) that are needed for the bean.

The <bean> Tag Structure

```
<bean id="demoBean" class="com.examples.DemoBean" />
```

The id attribute sets a unique id for the class specified by the class attribute.

Bean Definition Properties

Spring Bean

The following table contains the <bean> tag properties that are used to configure the bean in the configuration file.

Bean Property	Description
Class	This property is used to specify the class for which an object is created.
Name	This property is used to specify an identifier for a bean that is unique.
Scope	It specifies the scope of beans such as singleton or prototype.
Constructor arguments	It is used for constructor-based dependency injection.
Properties	This property is used for property-based dependency injection.
Autowiring mode	It is used to set bean auto wiring.

collaborators and lazy initialization mode	It is used to set lazy bean initialization.
Initialization method	It is used to set the initialization method to execute at bean initialization.
Destruction method	It is used to set destructive methods that execute before destroying of bean object.

1. Naming Bean

To set the name of a bean in XML-based configuration, we use the id, name attributes, or both. The id attribute lets us specify exactly one id.

Spring Bean

While setting bean names we are required to follow the naming conventions it means bean names start with a lowercase letter and are camel-cased.

```
<bean id="..." class="...">
   <!-- Configuration for this bean go here -->
</bean>
```

Lazy Initialization

By default, ApplicationContext implementations eagerly create and configure all singleton beans as part of the initialization process.

A lazy-initialized bean tells the IoC container to create a bean instance when it is first requested, rather than at startup.

```
<bean id = "..." class = "..." lazy-init = "true">

    <!-- Configuration for this bean go here -->

</bean>
```

Init Method

This attribute is used to specify the method that executes at bean initialization time. Syntax of the bean tag and attribute is given below.

Spring Bean

```
<bean id = "..." class = "..." init-method = "...">

    <!-- Configuration for this bean go here -->

</bean>
```

Destruction Method

This attribute is used to specify the method that executes at bean destroy time. The syntax of the bean tag and attribute is given below.

```
<bean id = "..." class = "..." destroy-method = "...">

    <!-- Configuration for this bean go here -->

</bean>
```

Simple Bean Example

Let's create an example to create a bean and access its properties from the bean and property tags. The following are Java and XML files that we created in our project.

// Employee.java

Spring Bean

This is a simple Java POJO bean class that has setter and getter methods to handle data.

```
package com.studytonight.community;

public class Employee

{

        int id;

        String name;

        public int getId()

         {

                return id;

        }

        public void setId(int id) {

                this.id = id;

        }



        public String getName() {

                return name;

        }
        public void setName(String name) {

                this.name = name;

        }

}
```

// SpringApp.java

This file reads the XML file and gets bean data set in the XML file. We used getter methods to get bean data.

```
import org.springframework.context.ApplicationContext;

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

Spring Bean

```
public class SpringApp

{

        private static ApplicationContext context;

        public static void main(String[] args)

        {

                context = new ClassPathXmlApplicationContext("applicationContext.xml");

                Employee e = (Employee) context.getBean("employee");

                System.out.println("Id: "+e.getId());

                System.out.println("Name: "+e.getName());

        }

}
```

// applicationContext.xml

This is a context file that configures the bean tag and its properties by using the property tag.

```
<?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"
```

Spring Bean

```
        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="employee" class="com.studytonight.community.Employee">

                <property name="id" value="0501"></property>

                <property name="name" value="Rahul"></property>

        </bean>

</beans>
```

// pom.xml

This is a maven dependency file that is used to configure all the project settings and dependencies.

```
<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.studytonight</groupId>
        <artifactId>SpringApp</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <dependencies>
```

```
		<!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-core</artifactId>
			<version>${spring.version}</version>
		</dependency>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-context</artifactId>
			<version>${spring.version}</version>
		</dependency>
	</dependencies>
	<properties>
		<spring.version>5.2.8.RELEASE</spring.version>
	</properties>
	<build>
		<sourceDirectory>src</sourceDirectory>
		<plugins>
			<plugin>
				<artifactId>maven-compiler-plugin</artifactId>
				<version>3.8.1</version>
				<configuration>
					<source>1.8</source>
					<target>1.8</target>
				</configuration>
			</plugin>
		</plugins>
	</build>
</project>
```

Run the Application

After successfully completing the project and adding the dependencies run the application and you will get the output as below.

```
Id: 501

Name: Rahul
```

Spring Bean