Spring @Required Annotation with Example

Last Updated : 23 Jul, 2025

Spring Annotations provide a powerful way to configure dependencies and implement dependency injection in Java applications. These annotations act as metadata, offering additional information about the program.

The **@Required annotation in Spring** is a method-level annotation used in the setter method of a bean property. It ensures that the specified bean property must be injected with a value at configuration time. If the property is not injected, Spring throws a BeanInitializationException.

In this article, we will demonstrate the use of the **@Required annotation** with a practical example.

> ***Note:*** *The @Required annotation was deprecated in Spring 5 and completely removed in Spring 6. It is no longer recommended for enforcing mandatory properties. Instead, you should use constructor injection or @Autowired with required = true (default behavior) to ensure that required dependencies are injected properly.*

Step-by-Step Implementation

Step 1: Create a Student Class

Create a simple Student class with three attributes: rollNo, name, and age. Define setter methods for these attributes and a method to display the student details.

```
// Java Program to Illustrate Student Class
public class Student {

    // Class data members
    private int rollNo;
```

```
    private String name;
    private int age;

    // Setter
    public void setRollNo(int rollNo)
    {
        this.rollNo = rollNo;
    }

    // Setter
    public void setName(String name) {
        this.name = name;

    }

    // Setter
    public void setAge(int age) {
        this.age = age;

    }

    // Method
    public void display()
    {

        // Printing attributes of student
        System.out.println("Roll No: " + rollNo);
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
```

Step 2: Create a Properties File

Create a properties file named student-info.properties in the classpath. This file will store the values for the Student bean properties.

```
student.rollNo = 201
student.name = Asish
student.age = 30
```

Step 3: Inject Values Using @Value Annotation

Modify the Student class to inject values from the properties file using the @Value annotation.

```
import org.springframework.beans.factory.annotation.Value;

public class Student {
    private int rollNo;
    private String name;
    private int age;

    @Value("${student.rollNo}")
    public void setRollNo(int rollNo) {
        this.rollNo = rollNo;
    }

    @Value("${student.name}")
    public void setName(String name) {
        this.name = name;
    }

    @Value("${student.age}")
    public void setAge(int age) {
        this.age = age;
    }

    public void display(){
        System.out.println("Roll No: " + rollNo);
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
```

Step 4: Configure the Spring Bean in beans.xml

Define the Student bean in the beans.xml configuration file. Enable annotation-based configuration and load the properties file.

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

    <context:annotation-config/>
      <context:property-placeholder location="classpath:student-
info.properties"/>

    <bean id="student" class="Student">

    </bean>
```

```
</beans>
```

Step 5: Create the Main Class

Create a Main class with the main method to load the Spring context and retrieve the Student bean.

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

public class Main {
    public static void main(String[] args) {
        // Using ApplicationContext tom implement Spring IoC
        ApplicationContext context = new
ClassPathXmlApplicationContext("beans.xml");

        // Get the bean student
        Student student = context.getBean("student", Student.class);

        // Calling the methods
        student.display();
    }
}
```

Now run your main() method and the output will be like this.

Output:

```
Roll No: 201
Name: Asish
Age: 30
```

Using @Required Annotation

The @Required annotation ensures that a specific property must be injected with a value. If the property is not injected, Spring throws a BeanInitializationException.

Step 6: Modify the Student Class to Use @Required

Add the @Required annotation to the setRollNo method to enforce that the rollNo property must be injected.

```
// Student.java
import org.springframework.beans.factory.annotation.Required;
import org.springframework.beans.factory.annotation.Value;

public class Student {

    private int rollNo;
    private String name;
    private int age;

    @Required
    @Value("${student.rollNo}")
    public void setRollNo(int rollNo) {
        this.rollNo = rollNo;
    }

    @Value("${student.name}")
    public void setName(String name) {
        this.name = name;
    }

    @Value("${student.age}")
    public void setAge(int age) {
        this.age = age;
    }

    public void display() {
        System.out.println("Roll No: " + rollNo);
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
```

Step 7: Test the @Required Annotation

If the rollNo property is not injected (e.g., by removing the @Value annotation), Spring will throw a BeanInitializationException.

```
// Student.java (Modified to remove @Value for rollNo)
import org.springframework.beans.factory.annotation.Required;

public class Student {

    private int rollNo;
    private String name;
    private int age;

    @Required
```

```
    public void setRollNo(int rollNo) {
        this.rollNo = rollNo;
    }

    @Value("${student.name}")
    public void setName(String name) {
        this.name = name;
    }

    @Value("${student.age}")
    public void setAge(int age) {
        this.age = age;
    }

    public void display() {
        System.out.println("Roll No: " + rollNo);
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
```

Exception:

```
Exception in thread "main"
org.springframework.beans.factory.BeanCreationException: Error
creating bean with name 'student' defined in class path resource
[beans.xml]: Initialization of bean failed; nested exception is
org.springframework.beans.factory.BeanInitializationException:
Property 'rollNo' is required for bean 'student'
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBe
anFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:610)
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBe
anFactory.createBean(AbstractAutowireCapableBeanFactory.java:524)
    at
org.springframework.beans.factory.support.AbstractBeanFactory.lambd
a$doGetBean$0(AbstractBeanFactory.java:335)
    at
org.springframework.beans.factory.support.DefaultSingletonBeanRegis
try.getSingleton(DefaultSingletonBeanRegistry.java:234)
    at
org.springframework.beans.factory.support.AbstractBeanFactory.doGet
Bean(AbstractBeanFactory.java:333)
    at
```

```
org.springframework.beans.factory.support.AbstractBeanFactory.getBe
an(AbstractBeanFactory.java:208)
    at
org.springframework.beans.factory.support.DefaultListableBeanFactor
y.preInstantiateSingletons(DefaultListableBeanFactory.java:944)
    at
org.springframework.context.support.AbstractApplicationContext.fini
shBeanFactoryInitialization(AbstractApplicationContext.java:918)
    at
org.springframework.context.support.AbstractApplicationContext.refr
esh(AbstractApplicationContext.java:583)
    at
org.springframework.context.support.ClassPathXmlApplicationContext.
<init>(ClassPathXmlApplicationContext.java:144)
    at
org.springframework.context.support.ClassPathXmlApplicationContext.
<init>(ClassPathXmlApplicationContext.java:85)
    at Main.main(Main.java:7)
Caused by:
org.springframework.beans.factory.BeanInitializationException:
Property 'rollNo' is required for bean 'student'
    at
org.springframework.beans.factory.annotation.RequiredAnnotationBean
PostProcessor.postProcessPropertyValues(RequiredAnnotationBeanPostP
rocessor.java:158)
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBe
anFactory.populateBean(AbstractAutowireCapableBeanFactory.java:1418
)
    at
org.springframework.beans.factory.support.AbstractAutowireCapableBe
anFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:601)
    ... 11 more
```

Key Points:

- The @Required annotation ensures that a specific property must be injected with a value.

- If the property is not injected, Spring throws a BeanInitializationException.
- The @Required annotation is used in conjunction with setter methods.
- Always use the @Value annotation or other dependency injection mechanisms to provide values for @Required properties.