Understanding Scope Annotations in Spring

Spring **scope annotations** define how a bean should be created and managed within the Spring container. The most commonly used scopes are:

1️⃣ `@Singleton` **(Default)** → Only **one** instance per Spring container.

2️⃣ `@Prototype` → A **new** instance every time the bean is requested.

3️⃣ `@Request` (for web apps) → A new instance per HTTP request.

4️⃣ `@Session` (for web apps) → A new instance per HTTP session.

💡 Simple Explanation with Example (Using Interfaces)

Let's modify our **interface-based implementation** to include **scope annotations**.

🔷 Step 1: Interface (Address.java)

```
java
CopyEdit
package in.sp.beans;

public interface Address {
    String getAddressDetails();
}
```

🔷 Step 2: Implementing Address (Two Beans)

🏠 `HomeAddress.java` (Singleton)

```
java
CopyEdit
package in.sp.beans;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

@Component("homeAddress")
@Scope("singleton")  // Only ONE instance shared across the container
public class HomeAddress implements Address {
    @Override
    public String getAddressDetails() {
        return "Home Address: #101, Bangalore -560001";
    }
}
```

🏢 OfficeAddress.java (Prototype)

```
java
CopyEdit
package in.sp.beans;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

@Component("officeAddress")
@Scope("prototype")  // A NEW instance created every time
public class OfficeAddress implements Address {
    @Override
    public String getAddressDetails() {
        return "Office Address: #202, Mumbai -400001";
    }
```

```
}
```

🔷 Step 3: `Student.java` (Injecting Address)

```
java
CopyEdit
package in.sp.beans;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;

@Component
public class Student {
    private Address address;

    @Autowired
    public Student(@Qualifier("officeAddress") Address address) {
        this.address = address;
    }

    public void display() {
        System.out.println("Student Address: " + address.getAddressDetails());
    }
}
```

🔷 Step 4: `MainApp.java` (Running the Code)

```
java
CopyEdit
package in.sp.main;
```

```
import in.sp.beans.Student;
import in.sp.beans.OfficeAddress;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationC
ontext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new AnnotationConfigApplicationContext("i
n.sp.beans");

        Student student1 = context.getBean(Student.class);
        student1.display();

        // Checking Scope of OfficeAddress
        OfficeAddress address1 = context.getBean(OfficeAddress.class);
        OfficeAddress address2 = context.getBean(OfficeAddress.class);

        System.out.println("Are OfficeAddress objects same? " + (address1 == ad
dress2)); // Should print false
    }
}
```

📝 Expected Output

```
less
CopyEdit
Student Address: Office Address: #202, Mumbai -400001
Are OfficeAddress objects same? false
```

Spring **scope annotations** define how a bean should be created and managed within the Spring container. The most commonly used scopes are:

1️⃣ `@Singleton` **(Default)** → Only **one** instance per Spring container.

2️⃣ `@Prototype` → A **new** instance every time the bean is requested.

3️⃣ `@Request` (for web apps) → A new instance per HTTP request.

4️⃣ `@Session` (for web apps) → A new instance per HTTP session.

💡 Simple Explanation with Example (Using Interfaces)

Let's modify our **interface-based implementation** to include **scope annotations**.

🔷 Step 1: Interface (Address.java)

```
java
CopyEdit
package in.sp.beans;

public interface Address {
    String getAddressDetails();
}
```

🔷 Step 2: Implementing Address (Two Beans)

🏠 `HomeAddress.java` (Singleton)

```
java
CopyEdit
package in.sp.beans;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

@Component("homeAddress")
```

```
@Scope("singleton")  // Only ONE instance shared across the container
public class HomeAddress implements Address {
    @Override
    public String getAddressDetails() {
        return "Home Address: #101, Bangalore -560001";
    }
}
```

🏢 `OfficeAddress.java` (Prototype)

```
java
CopyEdit
package in.sp.beans;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

@Component("officeAddress")
@Scope("prototype")  // A NEW instance created every time
public class OfficeAddress implements Address {
    @Override
    public String getAddressDetails() {
        return "Office Address: #202, Mumbai -400001";
    }
}
```

🔷 Step 3: `Student.java` (Injecting Address)

```
java
CopyEdit
package in.sp.beans;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;

@Component
public class Student {
    private Address address;

    @Autowired
    public Student(@Qualifier("officeAddress") Address address) {
        this.address = address;
    }

    public void display() {
        System.out.println("Student Address: " + address.getAddressDetails());
    }
}
```

🔷 Step 4: `MainApp.java` (Running the Code)

```
java
CopyEdit
package in.sp.main;

import in.sp.beans.Student;
import in.sp.beans.OfficeAddress;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationC
ontext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new AnnotationConfigApplicationContext("i
n.sp.beans");
```

```
        Student student1 = context.getBean(Student.class);
        student1.display();

        // Checking Scope of OfficeAddress
        OfficeAddress address1 = context.getBean(OfficeAddress.class);
        OfficeAddress address2 = context.getBean(OfficeAddress.class);

        System.out.println("Are OfficeAddress objects same? " + (address1 == ad
dress2)); // Should print false
    }
}
```

📝 Expected Output

```
less
CopyEdit
Student Address: Office Address: #202, Mumbai -400001
Are OfficeAddress objects same? false
```