

[Spring] 애노테이션을 이용한 빈 설정 방법 정리

빈 설정을 하기 위해 사용되는 스프링 애노테이션 **@Required**, **@Autowired**, **@Qualifier**, **@Value**과 JSR-250 애노테이션 **@PostConstruct**, **@PreDestroy**, **@Resource**에 대해 알아보자

이 애노테이션들을 사용해 기존에 XML 빈 설정 파일에 모두 작성하던 빈 설정을 빈 클래스에 직접 설정할 수 있다.

XML 설정에 애노테이션 빈 설정을 사용하기 위한 코드 추가


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

    <context:annotation-config/>
</beans>
```

context 네임 스페이스와 <context:annotation-config/> 코드를 추가한다.

이렇게 설정하면 빈 설정을 XML 파일이 아닌 빈 클래스의 애노테이션을 검색해 반영한다.

@Required

- setter에 붙여 반드시 주입해야하는 프로퍼티로 설정하는 애노테이션
- Spring 5.1 버전 부터 Deprecated 되었다.  반드시 주입해야 할 프로퍼티는 생성자 주입을 이용한다.
- 스프링 5.1이상을 사용하거나 자바 파일로 bean을 등록했을 경우 무시된다.

예제 코드

```
public class TestBean1 {

    private int data1;

    public int getData1() {
        return data1;
    }

    // 필수 주입 프로퍼티
    // 스프링 5.1이상을 사용하거나 자바 파일로 bean을 등록했을 경우 무시된다
    @Required
    public void setData1(int data1) {
```

```

        this.data1 = data1;
    }
}

@Configuration
public class BeanConfigClass {

    @Bean
    public TestBean1 testBean1() {
        return new TestBean1();
    }
}

```

@Autowired

- 객체 타입을 통해 빈 객체를 자동으로 주입한다.
- 필드, 생성자, setter에 붙일 수 있다.
- 필드, setter에 붙여서 사용할 경우 반드시 기본 생성자가 정의되어 있어야 한다.
- 필드에 붙이면 setter를 통해 주입되며 setter가 없을 경우 컴파일 과정에서 자동으로 추가된다.

예제 자바 코드

```

public class TestBean1 {

    // 필드 자동 주입
    // 자동으로 setter 메서드가 추가되어 setter 메서드를 통해 주입된다.
    @Autowired
    private DataBean1 data3;
}

public class DataBean1 { }

```

설정 - 자바

```

@Configuration
public class BeanConfigClass {

    @Bean
    public TestBean1 testBean1() {
        return new TestBean1();
    }

    @Bean
    public DataBean1 dataBean1() {
        return new DataBean1();
    }
}

```

설정 - XML

```
<?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"
       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">

    <context:annotation-config/>

    <bean id='testBean1' class='com.atoz_develop.beans.TestBean1' />
    <bean class='com.atoz_develop.beans.DataBean1' />

</beans>
```

@Qualifier

- @Autowired와 함께 사용한다.
- @Autowired를 통한 자동 주입 시 같은 타입의 빈이 여러 개 정의되어 있으면 @Qualifier에 설정되어 있는 빈을 찾아 주입한다.

예제 자바 코드

```
public class TestBean1 {

    @Autowired
    @Qualifier("obj4")
    private DataBean2 data4;

    @Autowired
    @Qualifier("obj5")
    private DataBean2 data5;
}

public class DataBean2 { }
```

설정 - 자바

```
@Configuration
public class BeanConfigClass {

    @Bean
    public TestBean1 testBean1() {
        return new TestBean1();
    }

    @Bean
```

```

    public DataBean2 obj4() {
        return new DataBean2();
    }

    @Bean
    public DataBean2 obj5() {
        return new DataBean2();
    }
}

```

설정 - XML

```

<?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"
       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">

    <context:annotation-config/>

    <bean id='testBean1' class='com.atoz_develop.beans.TestBean1' />
    <bean id='obj4' class='com.atoz_develop.beans.DataBean2' />
    <bean id='obj5' class='com.atoz_develop.beans.DataBean2' />

</beans>

```

@Value

- 생성자 주입 시 자동으로 주입되지 않는 기본 자료형과 문자열의 값을 설정한다.

예제 자바 코드

```

public class TestBean2 {

    private int data1;
    private String data2;
    private DataBean3 data3;
    private DataBean4 data4;

    public TestBean2(@Value("100") int data1, @Value("문자열") String data2, DataBean3 data3, DataBean4 data4) {
        this.data1 = data1;
        this.data2 = data2;
        this.data3 = data3;
        this.data4 = data4;
    }

}

public class DataBean3 { }

public class DataBean4 { }

```

설정 - XML

```
<?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"
       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">

    <context:annotation-config/>

    <bean id='testBean2' class='com.atoz_develop.beans.TestBean2' />

    <bean class='com.atoz_develop.beans.DataBean3' />
    <bean class='com.atoz_develop.beans.DataBean4' />

</beans>
```

XML 설정에서 <context:annotation-config/>를 추가하면 빈 클래스에 주입 설정을 따로 하지 않아도 생성자 주입이 자동으로 이루어진다.

이때 기본 타입과 문자열 타입의 필드는 자동으로 주입되지 않으며 @Value를 사용해서 값을 설정할 수 있다.

자바 애노테이션 설정(@Configuration)을 사용할때는 이 방법이 적용되지 않는다.

JSR-250 애노테이션 의존성 추가

```
<!-- https://mvnrepository.com/artifact/javax.annotation/javax.annotation-api -->
<dependency>
    <groupId>javax.annotation</groupId>
    <artifactId>javax.annotation-api</artifactId>
    <version>1.3.2</version>
</dependency>
```

@PostConstruct

- @Bean(initMethod="...") 대신 사용
- 생성자 호출 후 자동으로 호출될 메소드에 붙여서 사용

@Configuration 클래스에서 @Bean의 initMethod 속성에 빈이 생성될때 호출될 메소드를 지정하는 대신 빈의 클래스의 메소드에 @PostConstruct를 붙여 직접 설정할 수 있다.

예제 코드

```
public class TestBean2 {  
    // 생성자 호출 이후 자동으로 호출  
    @PostConstruct  
    public void init2() {  
        System.out.println("TestBean2의 init 메서드");  
    }  
}
```

설정

```
@Configuration  
public class BeanConfigClass {  
    @Bean  
    public TestBean2 obj2() {  
        return new TestBean2();  
    }  
}
```

@PreDestroy

- @Bean(destroyMethod="...") 대신 사용
- 객체 소멸 전 자동으로 호출될 메소드에 붙여서 사용

@Configuration 클래스에서 @Bean의 destroyMethod 속성에 빈이 생성될때 호출될 메소드를 지정하는 대신 빈의 클래스의 메소드에 @PreDestroy를 붙여 직접 설정할 수 있다.

예제 코드

```
public class TestBean2 {  
    // 객체가 소멸되기 전에 자동으로 호출  
    @PreDestroy  
    public void destroy2() {  
        System.out.println("TestBean2의 destroy 메서드");  
    }  
}
```

```
}  
}
```

설정

```
@Configuration  
public class BeanConfigClass {  
  
    @Bean  
    public TestBean2 obj2() {  
        return new TestBean2();  
    }  
}
```

@Resource

- 빈 이름(id)을 통해 자동 주입
- @Autowired + @Qualifier와 유사
- 필드명과 동일한 이름의 빈을 주입
- 필드명과 빈 이름이 다르면 @Resource(name="...")

👉 필드명과 빈 이름(id)가 동일한 경우

예제 코드

```
public class TestBean5 {  
  
    // 변수의 이름과 동일한 이름의 Bean이 주입된다.  
    @Resource  
    private DataBean1 data1;  
  
    @Resource  
    private DataBean2 data2;  
}
```

설정

```
@Configuration  
public class BeanConfigClass {  
  
    @Bean  
    public DataBean1 data1() {
```

```

        return new DataBean1();
    }

    @Bean
    public DataBean2 data2() {
        return new DataBean2();
    }

    @Bean
    public TestBean5 obj5() {
        return new TestBean5();
    }
}

```

👉 필드명과 빈 이름(id)가 다른 경우 - name 속성 사용

예제 코드

```

public class TestBean6 {

    @Resource(name = "data1")
    private DataBean1 data100;

    @Resource(name = "data2")
    private DataBean2 data200;
}

```

설정

```

@Configuration
public class BeanConfigClass {

    @Bean
    public DataBean1 data1() {
        return new DataBean1();
    }

    @Bean
    public DataBean2 data2() {
        return new DataBean2();
    }

    @Bean
    public TestBean6 obj6() {
        return new TestBean6();
    }
}

```

TestBean6의 DataBean1타입 필드명이 각각 data100, data200이고 빈 등록은 data1, data2로 되어있으므로 그냥 @Resource를 붙이면 자동 주입이 이루어지지 않는다.

이런 경우 @Resource의 name 속성을 사용해서 빈 이름을 직접 지정한다.