Spring Framework

8. DI 애플리케이션 작성(3)

Tacademy

CONTENTS

학습 목표

- Bean 의존관계 설정 방법에 대하여 이해할 수 있습니다.
- 프로퍼티(Property)값 설정 방법에 대하여 이해할 수 있습니다.
- 프로퍼티(Property) 파일을 이용한 설정 방법에 대하여 이해할 수 있습니다.

1. Bean 의존관계 설정 방법

Setter Injection : <property> 태그

Setter 메서드를 통해 의존관계가 있는 Bean을 주입하려면 <property> 태그를 사용할 수 있다.

- ref 속성은 사용하면 Bean 이름을 이용해 주입할 Bean을 찾는다.
- value 속성은 단순 값 또는 Bean이 아닌 객체를 주입할 때 사용한다.

Setter Injection : <property> 태그

```java
public class Hello {
    String name;
    Printer printer;

    public Hello() { }

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

    public void setPrinter(Printer printer) {
        this.printer = printer;
    }
```

```xml
<bean id="hello" class="myspring.di.xml.Hello">
      <property name="name" value="Spring" />
      <property name="printer" ref="printer" />
</bean>

<bean id="printer"
     class="myspring.di.xml.StringPrinter" />
```

Constructor Injection : <constructor-arg> 태그

Constructor를 통해 의존관계가 있는 Bean을 주입하려면 <constructor-arg> 태그를 사용할 수 있다.

Constructor 주입방식은 생성자의 파라미터를 이용하기 때문에 한번에 여러 개의 객체를 주입할 수 있다.

```
public class Hello {
    String name;
    Printer printer;

    public Hello() {}

    public Hello(String name, Printer printer) {
        this.name = name;
        this.printer = printer;
    }
```

Constructor Injection : <constructor-arg> 태그

Constructor를 통해 의존관계가 있는 Bean을 주입하려면 <constructor-arg> 태그를 사용할 수 있다.

Constructor 주입방식은 생성자의 파라미터를 이용하기 때문에 한번에 여러 개의 객체를 주입할 수 있다.

❖ 생성자 주입을 위한 설정 : index 지정

```
<bean id="hello" class="myspring.di.xml.Hello">
        <constructor-arg index="0" value="Spring"/>
        <constructor-arg index="1" ref="printer"/>
</bean>
```

Constructor Injection : <constructor-arg> 태그

Constructor를 통해 의존관계가 있는 Bean을 주입하려면 <constructor-arg> 태그를 사용할 수 있다.

Constructor 주입방식은 생성자의 파라미터를 이용하기 때문에 한번에 여러 개의 객체를 주입할 수 있다.

❖ **생성자 주입을 위한 설정 : 파라미터 이름 지정**

```
<bean id="hello" class="myspring.di.xml.Hello">
        <constructor-arg name="name" value="Spring"/>
        <constructor-arg name="printer" ref="printer"/>
</bean>
```

POJO 클래스 수정 및 Bean 설정 파일 수정

*Hello.java

```java
package myspring.di.xml;

public class Hello {
    String name;
    Printer printer;

    public Hello() {}

    public Hello(String name, Printer printer) {
        this.name = name;
        this.printer = printer;
    }
```

beans.xml

```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"
    xsi:schemaLocation="http://www.springframework.org/sche

    <bean id="hello2" class="myspring.di.xml.Hello">
        <constructor-arg index="0" value="Spring" />
        <constructor-arg index="1" ref="printer" />
    </bean>

    <bean id="printer"
            class="myspring.di.xml.StringPrinter" />
```

2. 프로퍼티(Property) 값 설정 방법

2. 프로퍼티(Property) 값 설정 방법

단순 값(문자열이나 숫자)의 주입(Injection)

Setter 메서드를 통해 Bean의 레퍼런스가 아니라 단순 값을 주입하려고 할 때는 <property> 태그의 value 속성을 사용한다.

```
public class Hello {
    String name;
    Printer printer;

    public Hello() {}

    public void setName(String name) {
        this.name = name;
    }
```

```
<bean id="hello3" class="myspring.di.xml.Hello">
    <property name="name" value="스프링" />
</bean>
```

컬렉션(Collection) 타입의 값 주입(Injection) (1)

Spring은 List, Set, Map, Properties와 같은 컬렉션 타입을
XML로 작성해서 프로퍼티에 주입하는 방법을 제공한다.

❖ List와 Set 타입 : <list>와 <value> 태그를 이용

◉ 프로퍼티가 Set 타입 이면 <list> 대신에 <set>을 사용하면 된다.

```
public class Hello {
    List<String> names;

    public void setNames(List<String> list) {
        this.names = list;
    }
```

```
<bean id="hello" class="myspring.di.xml.Hello">
    <property name="names">
        <list>
            <value>Spring</value>
            <value>IoC</value>
            <value>DI</value>
        </list>
    </property>
</bean>
```

컬렉션(Collection) 타입의 값 주입(Injection) (2)

Spring은 List, Set, Map, Properties와 같은 컬렉션 타입을
XML로 작성해서 프로퍼티에 주입하는 방법을 제공한다.

❖ **Map 타입 :** <map>과 <entry> 태그를 이용

```java
public class Hello {
    Map<String,Integer> ages;

    public void setAges(Map<String, Integer> ages) {
        this.ages = ages;
    }
```

```xml
<bean id="hello" class="myspring.di.xml.Hello">
    <property name="ages">
        <map>
            <entry key="Kim" value="30" />
            <entry key="Lee" value="35" />
            <entry key="Ahn" value="40" />
        </map>
    </property>
</bean>
```

2. 프로퍼티(Property) 값 설정 방법

POJO 클래스 수정 및 Bean 설정 파일 수정

Hello.java

```
3 import java.util.List;
5
6 public class Hello {
7     String name;
8     Printer printer;
9     List<String> names;
10
11    public Hello() {
12    }
13
14    public void setNames(List<String> list) {
15        this.names = list;
16    }
```

*beans.xml

```
5
6     <bean id="hello" class="myspring.di.xml.Hello">
7         <property name="name" value="Spring" />
8         <property name="printer" ref="printer" />
9         <property name="names">
10            <list>
11                <value>Spring</value>
12                <value>IoC</value>
13                <value>DI</value>
14            </list>
15        </property>
16    </bean>
```

2. 프로퍼티(Property) 값 설정 방법

DI 테스트 클래스 수정

HelloBeanSpringTest.java

```
16 @RunWith(SpringJUnit4ClassRunner.class)
17 @ContextConfiguration(locations = "classpath:config/beans.xml")
18 public class HelloBeanSpringTest {
19     @Autowired
20     private ApplicationContext context;
21
22     @Test
23     public void bean1() {
24         Hello hello = (Hello) context.getBean("hello");
25         assertEquals("Hello Spring", hello.sayHello());
26         hello.print();
27
28         assertEquals(3, hello.getNames().size());
29         List<String> list = hello.getNames();
30         for (String value : list) {
31             System.out.println(value);
32         }
33     }
```

3. 프로퍼티(Property) 파일을 이용한 설정 방법

환경에 따라 자주 변경되는 내용의 분리

- XML의 Bean 설정 메타정보는 애플리케이션 구조가 바뀌지 않으면 자주 변경되지 않는다.
- 반면에 프로퍼티 값으로 제공되는 일부 설정정보 (예-DataSource Bean이 사용하는 DB 연결정보)는 애플리케이션이 동작하는 환경(개발, 테스트, 스테이징, 운영)에 따라서 자주 바뀔 수 있다.
- 변경되는 이유와 시점이 다르다면 분리하는 것이 객체지향 설계의 기본 원칙이다. 설정에도 동일한 원칙을 적용할 수 있다.
- 환경에 따라 자주 변경될 수 있는 내용은 properties 파일로 분리하는 것이 가장 깔끔하다 XML 처럼 복잡한 구성이 필요 없고 키와 값의 쌍(key=value)으로 구성하면 된다.

환경에 따라 자주 변경되는 내용의 분리의 예시(1)

- value 속성에 설정된 값들은 환경에 따라 변경될 수 있는 내용이다.
- 자주 변경되는 값들은 properties 파일에 넣어 분리하는 것이 좋다.

*beans.xml

```
6  <bean id="dataSource"
7      class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
8      <property name="driverClass" value="com.sql.jdbc.Driver" />
9      <property name="url" value="jdbc:mysql://localhost/testdb" />
10     <property name="username" value="spring" />
11     <property name="password" value="book" />
12 </bean>
```

환경에 따라 자주 변경되는 내용의 분리의 예시(2)

- 프로퍼티 파일로 분리한 정보는 ${ } (프로퍼티 치환자)을 이용하여 설정한다.
- ${ } 값을 치환해주는 기능은 <context:property-placeholder> 태그에 의해 자동으로 등록되는 PropertyPlaceHolderConfigurer Bean이 담당한다.

database.properties

```
1 db.driverClass=com.sql.jdbc.Driver
2 db.url=jdbc:mysql://l
3 db.username=spring
4 db.password=book
```

beans.xml

```
 8 <context:property-placeholder
 9         location="classpath:config/database.properties" />
10
11 <bean id="dataSource"
12     class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
13     <property name="driverClass" value="${db.driverClass}" />
14     <property name="url" value="${db.url}" />
15     <property name="username" value="${db.username}" />
16     <property name="password" value="${db.password}" />
17 </bean>
```

Bean 설정 파일 수정

value.properties

```
myname=Spring
myprinter=printer
value1=JUnit
value2=AOP
value3=DI
```

beans.xml

```
<context:property-placeholder
        location="classpath:config/value.properties" />

<bean id="hello" class="myspring.di.xml.Hello">
    <property name="name" value="${myname}" />
    <property name="printer" ref="${myprinter}" />
    <property name="names">
        <list>
            <value>${value1}</value>
            <value>${value2}</value>
            <value>${value3}</value>
        </list>
    </property>
</bean>
```

학습정리

지금까지 [DI 애플리케이션 작성(3)]에 대해서 살펴보았습니다.

Bean 의존관계 설정 방법

<property >, <constructor-arg> 태그

프로퍼티(property) 값 설정 방법

- <property> 태그의 value 속성
- <list>, <set>, <map> 태그

프로퍼티(property) 파일을 이용한 값 설정방법

Properties 파일 작성, ${ } 치환자 사용, <context:property-placeholder>