Spring Framework

6. DI 애플리케이션 작성(1)

Tacademy

CONTENTS

학습 목표

- Spring DI 용어에 대하여 이해할 수 있습니다.
- POJO 클래스를 작성할 수 있습니다.
- 설정 메타정보 XML을 작성할 수 있습니다.
- DI 테스트 클래스를 작성할 수 있습니다.

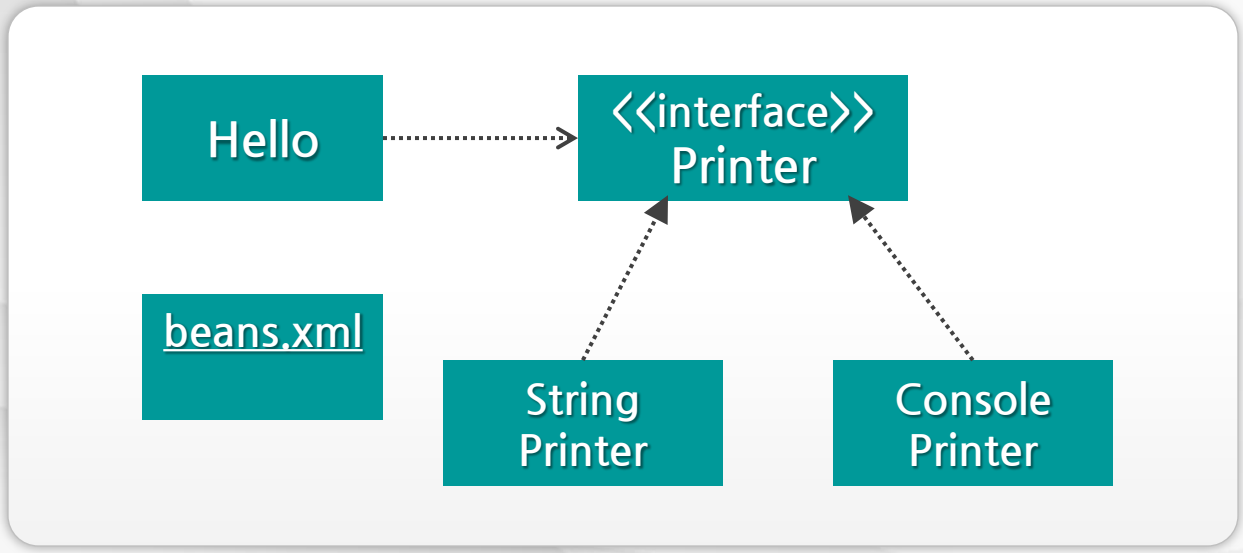
1. Spring DI 용어

1. Spring DI(Dependency Injection) 용어

빈 (Bean)	▪ 스프링이 IoC 방식으로 관리하는 객체라는 뜻 ▪ 스프링이 직접 생성과 제어를 담당하는 객체를 Bean이라고 부름
빈 팩토리 (BeanFactory)	▪ 스프링의 IoC를 담당하는 핵심 컨테이너를 가리킴 ▪ Bean을 등록, 생성, 조회, 반환하는 기능을 담당함 ▪ 이 BeanFactory를 바로 사용하지 않고 이를 확장한 ApplicationContext를 주로 이용함
애플리케이션 컨텍스트 (Application Context)	▪ BeanFactory를 확장한 IoC 컨테이너 ▪ Bean을 등록하고 관리하는 기능은 BeanFactory와 동일하지만 스프링이 제공하는 각종 부가 서비스를 추가로 제공함 ▪ 스프링에서는 ApplicationContext를 BeanFactory 보다 더 많이 사용함
설정 메타정보 (Configuration metadata)	▪ ApplicationContext 또는 BeanFactory가 IoC를 적용하기 위해 사용하는 메타정보를 말함 ▪ 설정 메타정보는 IoC컨테이너에 의해 관리되는 Bean 객체를 생성하고 구성할 때 사용됨

2. POJO 클래스 작성

POJO 클래스 다이어그램



Hello.java

*Hello.java

```java
package myspring.di.xml;

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;
    }

    public String sayHello() {
        return "Hello " + name;
    }

    public void print() {
        this.printer.print(sayHello());
    }
}
```

Printer.java

Printer.java

```
1  package myspring.di.xml;
2
3  public interface Printer {
4      public void print(String message);
5  }
6
```

StringPrinter.java

StringPrinter.java

```
1  package myspring.di.xml;
2
3  public class StringPrinter implements Printer {
4      private StringBuffer buffer = new StringBuffer();
5
6      public void print(String message) {
7          this.buffer.append(message);
8      }
9
10     public String toString() {
11         return this.buffer.toString();
12     }
13 }
14
```

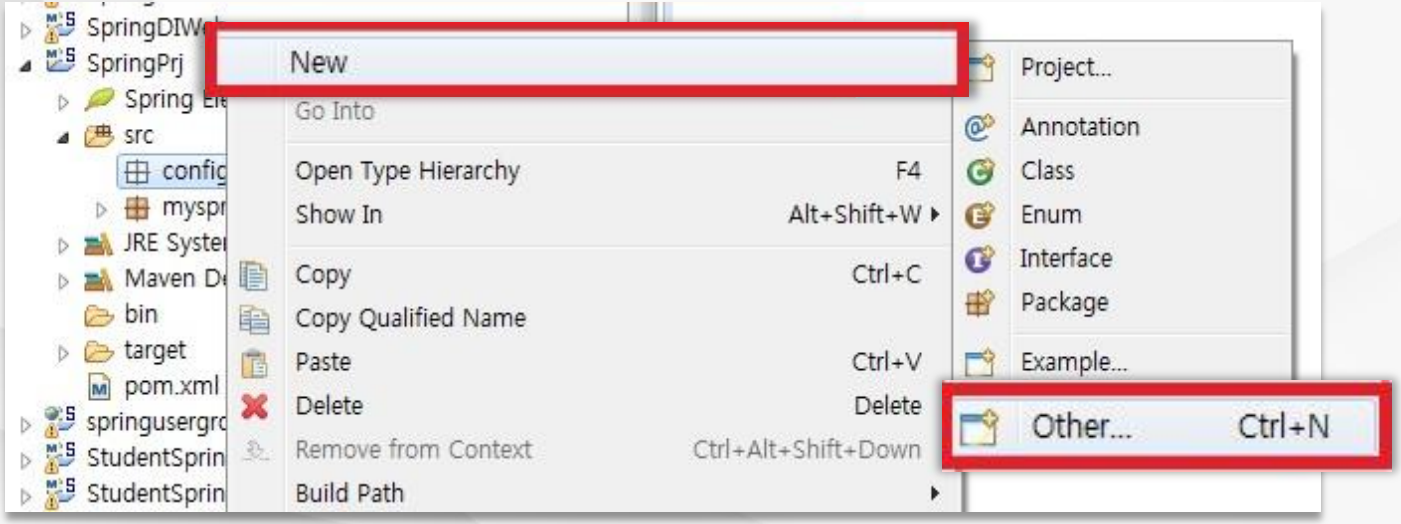
ConsolePrinter.java

ConsolePrinter.java

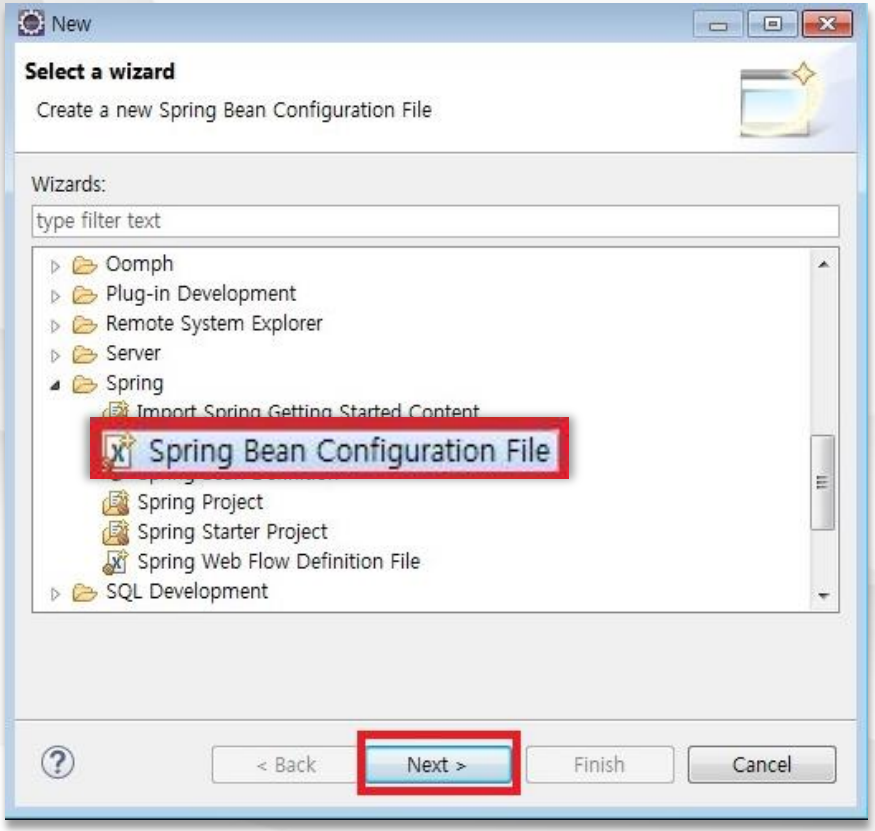
```
1  package myspring.di.xml;
2
3  public class ConsolePrinter implements Printer {
4      public void print(String message) {
5          System.out.println(message);
6      }
7  }
8
```

3. 설정 메타정보 XML 작성

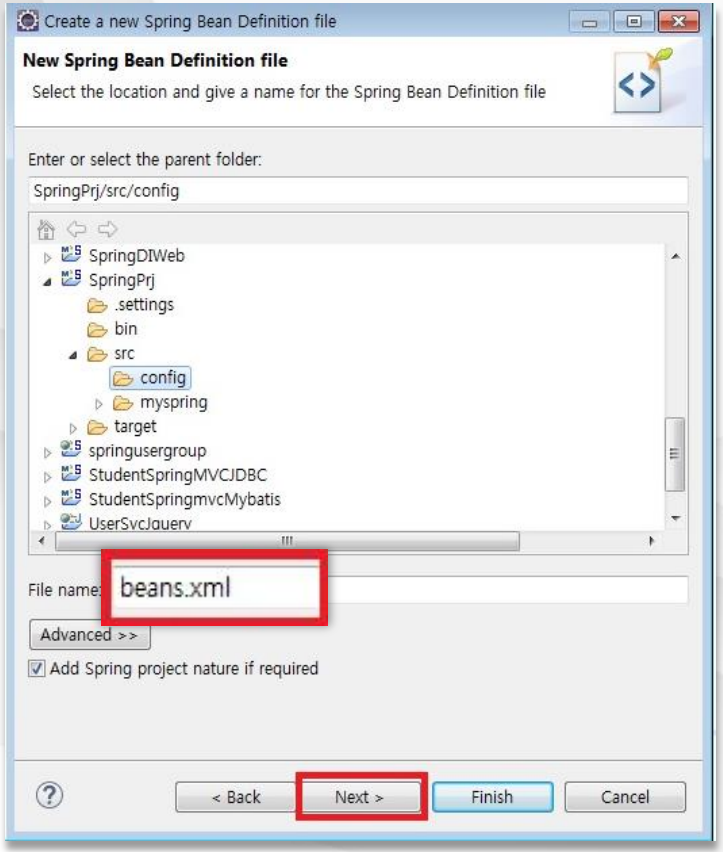
Bean Configuration(빈 설정) XML 작성 (1)



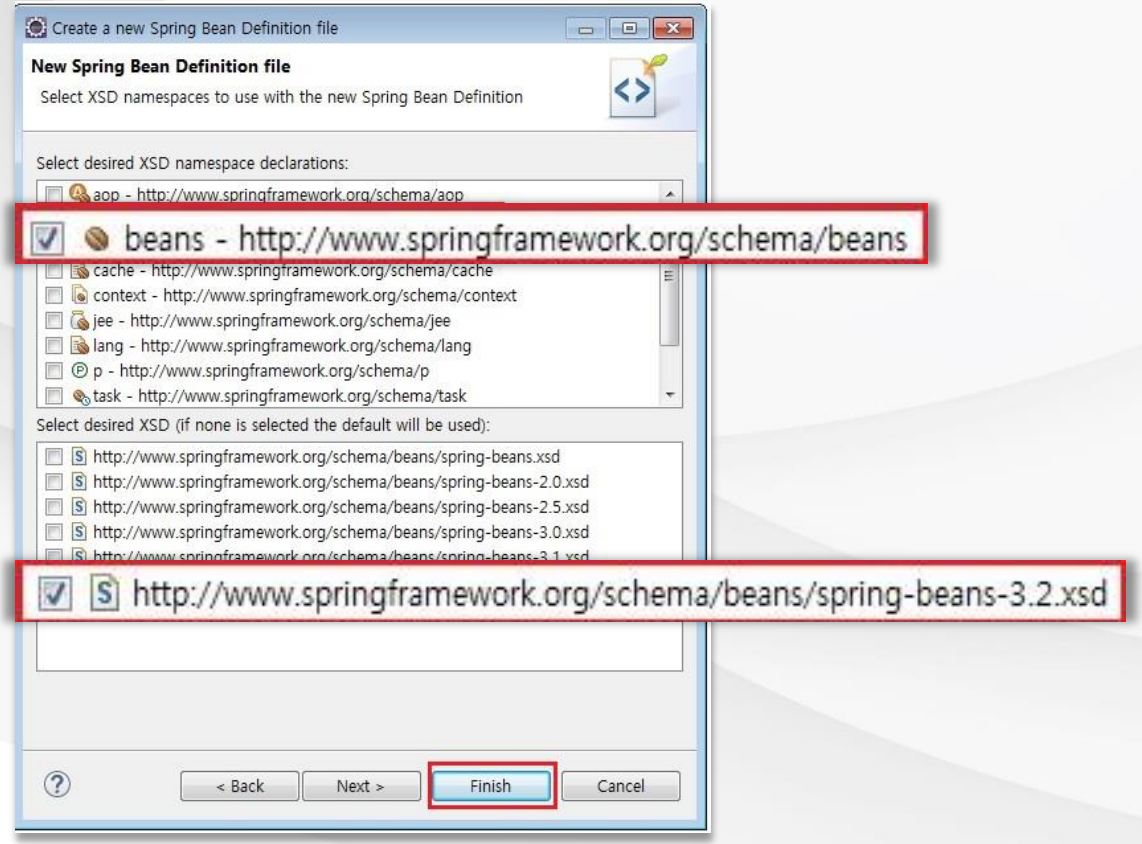
Bean Configuration(빈 설정) XML 작성 (2)



Bean Configuration(빈 설정) XML 작성 (3)



Bean Configuration(빈 설정) XML 작성 (4)



Bean Configuration(빈 설정) XML 작성 (5)

beans.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://www.springframework.org/schema/beans http.
5
6      <bean id="hello" class="myspring.di.xml.Hello">
7          <property name="name" value="Spring" />
8          <property name="printer" ref="printer" />
9      </bean>
10
11     <bean id="printer" class="myspring.di.xml.StringPrinter" />
12     <bean id="consolePrinter" class="myspring.di.xml.ConsolePrinter" />
13
14 </beans>
```

4. DI 테스트 클래스 작성

HelloBeanTest.java

*HelloBeanTest.java

```
1  package myspring.di.xml.test;
2
3  import org.springframework.context.ApplicationContext;
8
9  public class HelloBeanTest {
10     public static void main(String[] args) {
11         ApplicationContext context =
12                 new GenericXmlApplicationContext("config/beans.xml");
13         Hello hello = (Hello) context.getBean("hello");
14         System.out.println(hello.sayHello());
15         hello.print();
16         Printer printer = (Printer) context.getBean("printer");
17         System.out.println(printer.toString());
18
19         Hello hello2 = context.getBean("hello", Hello.class);
20         hello2.print();
21
22         System.out.println(hello == hello2);
23     }
24 }
```

학습정리

지금까지 [DI 애플리케이션 작성(1)]에 대해서 살펴보았습니다.

Spring DI 용어

빈(Bean), 빈 팩토리(BeanFactory), 애플리케이션 컨텍스트(ApplicationContext),
설정 메타정보(Configuration Metadata)

POJO 클래스 작성

의존관계가 있는 Java 클래스 작성
Hello.java, Printer.java, StringPrinter.java, ConsolePrinter.java

설정 메타정보 XML 작성

빈 설정(Bean Configuration) XML 파일작성 - beans.xml

DI 테스트 클래스 작성

DI 컨테이너 (ApplicationContext)를 사용한 테스트 클래스 작성