KH정보교육원

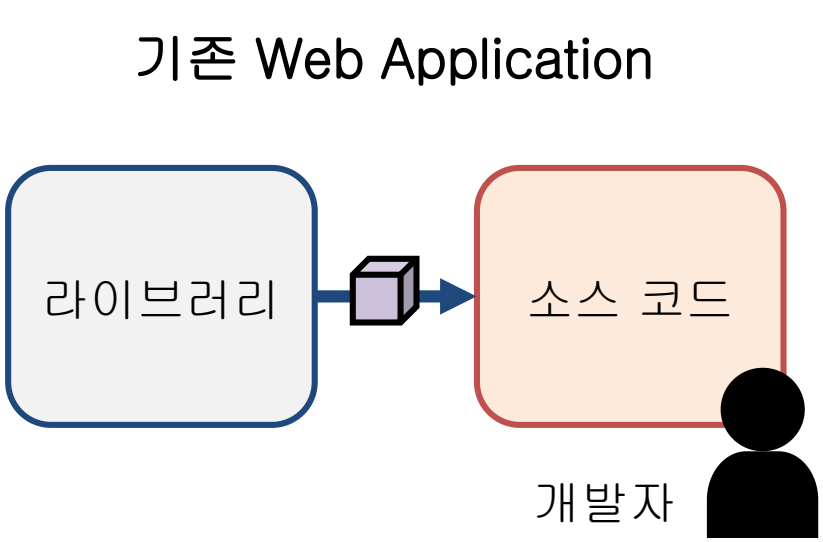
Spring IOC

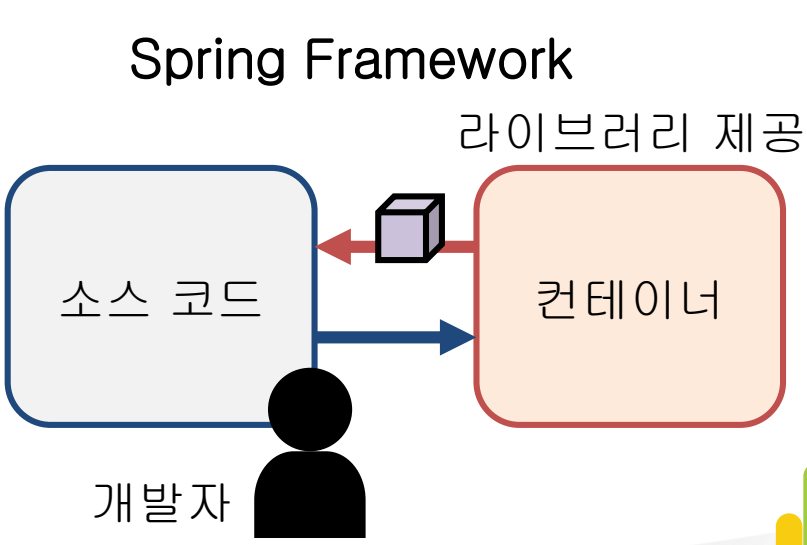
Spring IoC

KH정보교육원

IoC (제어의 역행)

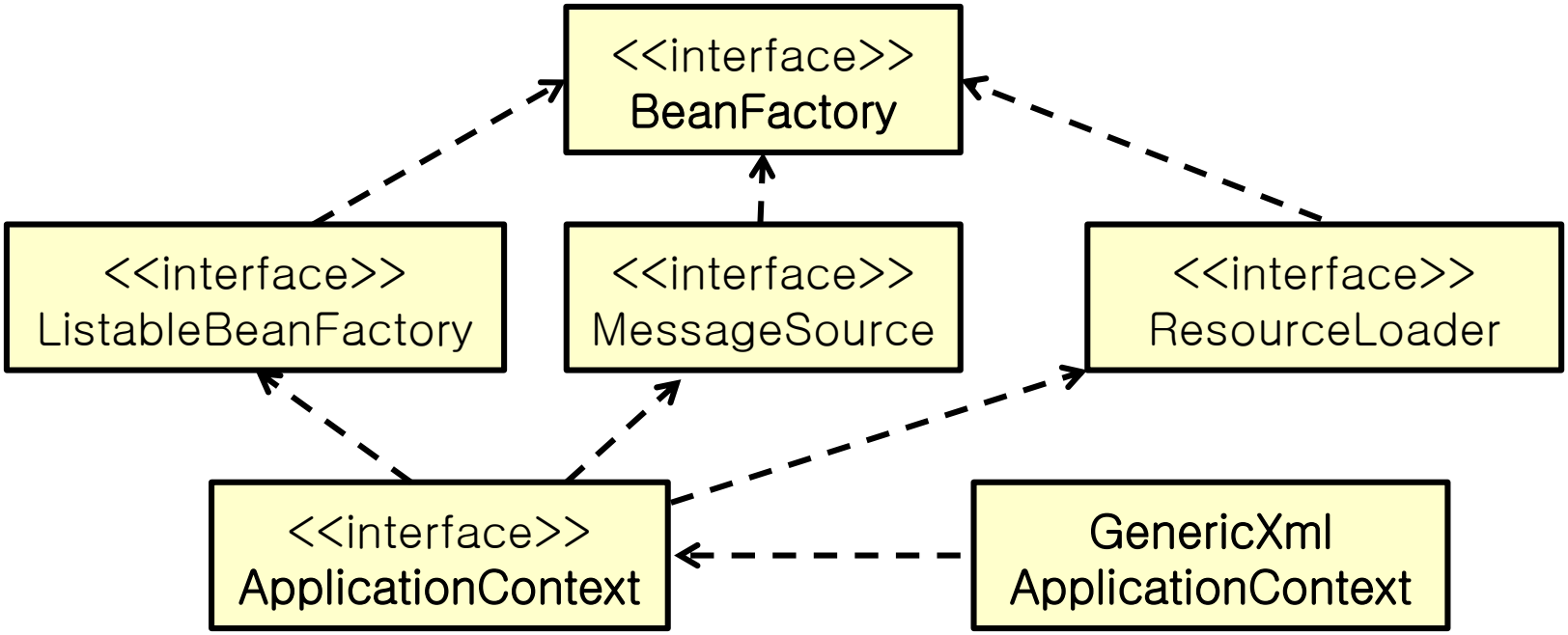
IoC(Inversion of Control)란, 프로그램을 구동하는데 필요한 객체에 대한 생성, 변경 등의 관리를 프로그램을 **개발하는 사람이 아닌** 프로그램을 구동하는 **컨테이너에서 직접 관리**하는 것을 말한다.
스프링은 IoC 구조를 통해 구동 시 필요한 객체의 생성부터 생명 주기까지 해당 객체에 대한 관리를 직접 수행한다.





IoC 컨테이너

스프링에서는 관리하는 객체를 'Bean(빈)'이라고 하고, 해당 빈들을 관리한다는 의미로 컨테이너를 'Bean Factory'라고 한다.



Spring IoC

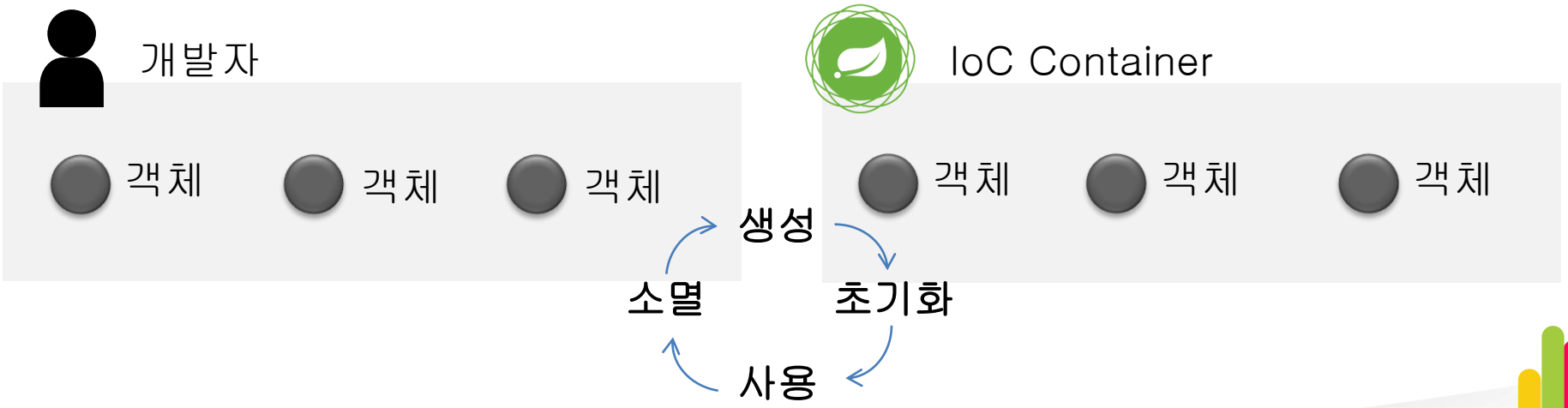
KH정보교육원

IoC 컨테이너의 역할

1. 객체의 생명주기와 의존성을 관리한다.

2. VO(DTO/POJO) 객체의 생성, 초기화, 소멸 등의 처리를 담당한다.

3. 개발자가 직접 객체를 생성할 수 있지만 해당 권한을 컨테이너에 맡김으로써 소스 코드 구현의 시간을 단축할 수 있다.



IoC 컨테이너와 Bean 객체

빈 (Bean)	- 스프링이 IoC 방식으로 관리하는 Class - **스프링이 직접 생성과 제어를 담당**하는 객체
빈 팩토리 (BeanFactory)	- 스프링의 IoC를 담당하는 핵심 컨테이너 - **Bean을 등록, 생성, 조회, 반환하는 기능을 담당**
애플리케이션 컨텍스트 (Application Context)	- **BeanFactory를 확장한 IoC 컨테이너** - Bean을 등록하고 관리하는 기능은 BeanFactory와 동일하지만 스프링이 제공하는 각종 부가 서비스를 추가로 제공함
설정 메타정보 (Configuration metadata)	- ApplicationContext 또는 BeanFactory가 **IoC를 적용하기 위해 사용하는 설정 정보** - 설정 메타정보는 IoC컨테이너에 의해 관리되는 Bean 객체를 생성하고 구성할 때 사용됨

주요 컨테이너 종류

BeanFactory	- 자바 빈 객체를 등록하고 이를 관리한다. - getbean() 메소드가 정의되어 있다.
Application Context	- BeanFactory의 확장 개념이다. - Spring의 각종 부가 서비스를 제공한다. - 일반적인 IoC 컨테이너를 말한다.
GenericXml Application Context	- ApplicationContext 를 구현한 클래스 - 일반적인 XML 형태의 문서를 읽어 컨테이너역할을 수행한다.

KH정보교육원

Spring DI

DI (의존성 주입)

DI(Dependency Injection)란 IoC 구현의 핵심 기술로, 사용하는 객체를 직접 생성하여 만드는 것이 아니라 컨테이너가 빈의 설정 정보를 읽어와 자동으로 해당 객체에 연결하는 것을 말한다.

이렇게 의존성을 주입 받게 되면 이후 해당 객체를 수정해야 할 상황이 발생했을 때 소스 코드의 수정을 최소화 할 수 있다.

DI의 장점

1. 개발자가 작성해야 할 코드가 단순해진다.

2. 각 객체 간의 종속 관계(결합도)를 해소할 수 있다.

객체간의 종속관계(결합도)

한 클래스에서 필드 객체를 생성 할 때 발생하는 **두 객체 간의 관계**를 말하며, **각 객체간의 내용이 수정될 경우 영향을 미치는 정도**를 나타낸다.

예를 들어 A Class에서 B Class를 생성할 경우, 만약 B Class의 생성자의 매개변수가 변경되거나 제공하는 메소드가 변경될 경우 이를 사용하는 A Class의 일부 정보도 필히 수정해야 하는 상황이 발생하는데 이를 '두 객체간 종속관계(결합도)가 강하다' 라고 표현한다.

DI 의 종류

◆ Setter 메소드를 통한 의존성 주입

의존성을 주입받는 Setter 메소드를 만들고, 이를 통해 의존성을 주입

◆ 생성자를 통한 의존성 주입

필요한 의존성을 포함하는 클래스에 생성자를 만들고, 이를 통해 의존성을 주입

◆ 메소드를 통한 의존성 주입

의존성을 입력 받는 일반 메소드를 만들고 이를 통해 의존성을 주입

Setter 메소드를 통한 의존성 주입

Setter 메소드를 통해 의존관계가 있는 Bean을 주입하려면
<property> 태그를 사용한다.

XML 선언 방법

```
<bean  id="객체의 이름"  class="클래스 풀네임">
    <property name="name" value="OOO" />
    <property name="name" ref="OOO" />
</bean>
```

- **name 속성**은 Class에서 선언한 필드 변수의 이름을 사용한다.
- **value 속성**은 단순 값 또는 Bean이 아닌 객체를 주입할 때 사용한다.
- **ref 속성**은 사용하면 Bean 이름을 이용해 주입할 Bean을 찾는다.

KH정보교육원

Setter 메소드를 통한 의존성 주입 예시

```
<bean id="student" class="com.kh.firstSpring.person.model.vo.Student">
    <property name="name" value="홍길동" />
    <property name="wallet"  ref="money" />
</bean>

<bean id="money" class="com.kh.firstSpring.wallet.model.vo.Wallet" />
```

생성자를 통한 의존성 주입

Constructor를 통해 의존관계가 있는 Bean을 주입하려면
<constructor-arg> 태그를 사용한다.

XML 선언 방법

```
<bean id="불러 올 객체"  class="클래스 풀네임">
        <constructor-arg  index="0"  value="OOO"/>
        <constructor-arg  name="OOO"  ref="OOO"/>
</bean>
```

- Constructor 주입방식은 생성자의 파라미터를 이용하기 때문에 **한번에 여러 개의 객체를 주입**할 수 있다.
- 필드 선언 순서에 따라 **index 속성**을 통해서도 접근이 가능하다.

생성자를 통한 의존성 주입 예시

```
<bean id="student" class="com.kh.firstSpring.person.model.vo.Student">
        <constructor-arg  index="0"  value="홍길동"/>
        <constructor-arg  index="1"  ref="money"/>
</bean>

<bean id="money" class="com.kh.firstSpring.wallet.model.vo.Wallet" />
```

KH정보교육원

Spring DI 실습

Spring DI 실습

KH정보교육원

Person Class

Spring DI 실습을 위한 Person VO 객체를 만든다.

```
package com.kh.testSpring.person.model.vo;

public class Person {
	private String name;	// 이름
	private Job myJob;	// 직업

	public Person() { }	// 기본 생성자

	// Getter & Setter
}
```

Spring DI 실습

KH정보교육원

Job Interface

Spring DI 실습을 위한 Job Interface 를 만든다.

```
package com.kh.testSpring.job.model.vo;

public interface Job {

    // 직업 정보를 출력하기 위한 추상 메소드
    public void printInfo(String compName);

}
```

Spring DI 실습

KH정보교육원

Developer Class

Job 인터페이스를 상속 받는 Developer 객체를 만든다.

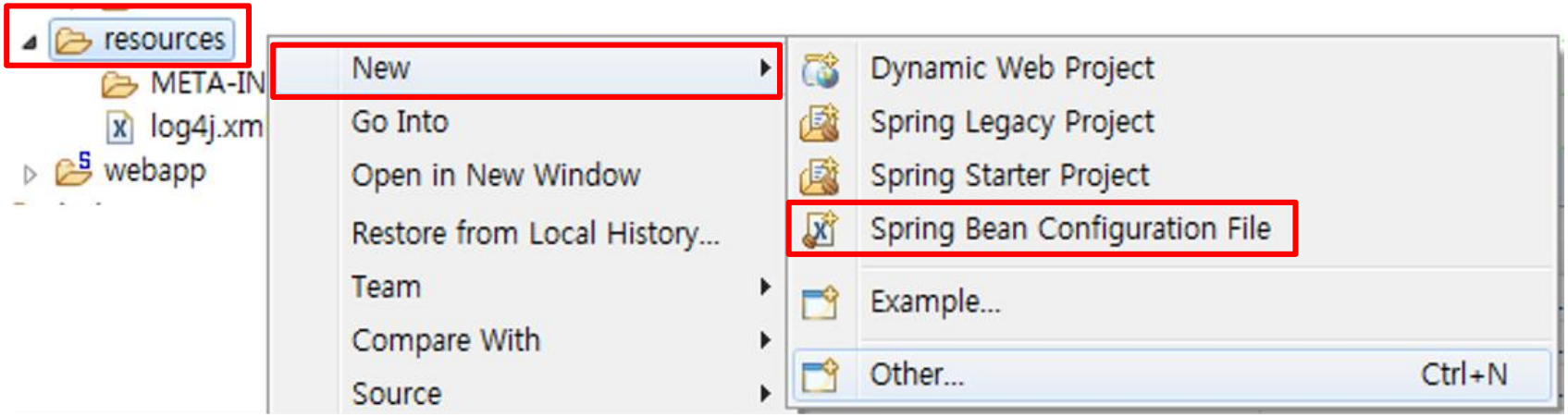
```
package com.kh.testSpring.job.model.vo;

public class Developer implements Job {

    @Override
    public void printInfo(String compName) {

        System.out.println("직장 명 : "+compName);

    }
}
```

Spring DI 실습

KH정보교육원

GenericXmlApplicationContext 생성

resources 폴더를 우클릭하여 Spring Bean Configuration File 을 클릭한다.



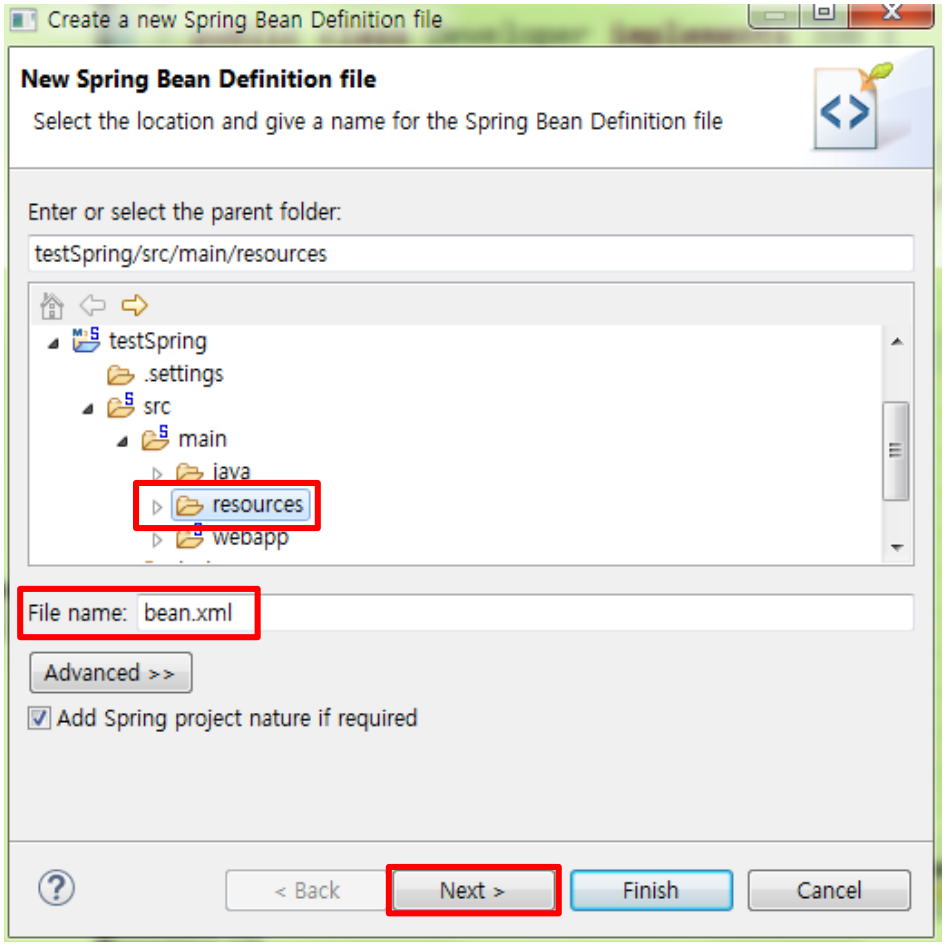
※ 만약 해당 목록이 보이지 않는다면 Other... 를 선택하여 spring tab에 있는 목록을 통해 진행하도록 하자.

Spring DI 실습

KH정보교육원

GenericXmlApplicationContext 생성

resources 폴더 경로를 확인하고, 파일의 이름을 bean.xml로 지은 뒤 Next



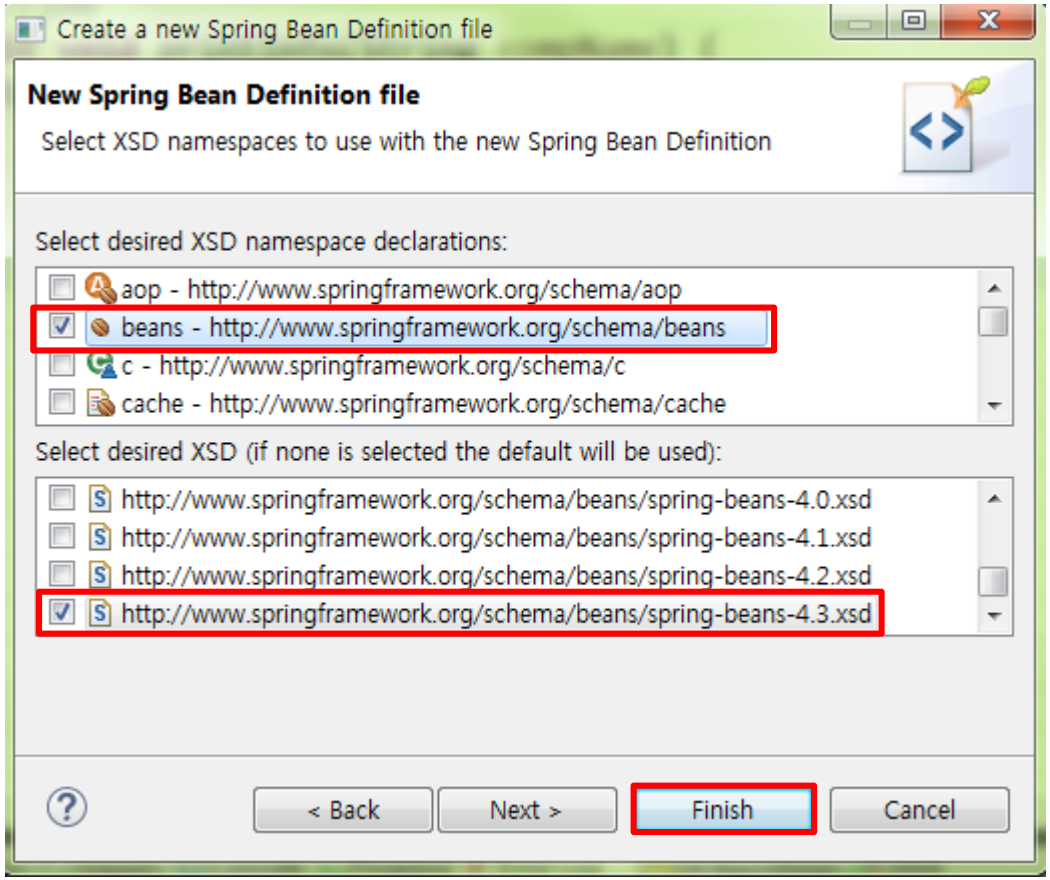
※ 사용할 xml 파일의 이름은 반드시 같지 않아도 된다.

Spring DI 실습

KH정보교육원

GenericXmlApplicationContext 생성

spring에서 제공하는 네임스페이스 중 Beans 를 선택하고, 버전은 최신버전을 선택한 뒤 Finish



Spring DI 실습

KH정보교육원

GenericXmlApplicationContext 설정

생성된 xml 을 확인 한 뒤, 다음과 같은 내용을 추가한다.

```
<!-- 생성자를 사용하여 DI를 적용할 경우 -->
<bean id="person" class="com.kh.testSpring.person.model.vo.Person">
   <constructor-arg index="0" value="홍길동"/>
   <constructor-arg index="1" ref="job"/> <!-- name 속성도 가능 -->
</bean>

<!-- Setter를 사용하여 DI를 적용할 경우 -->
<bean id="person2" class="com.kh.testSpring.person.model.vo.Person">
    <!-- setName() -->
    <property name="name" value="이순신" />
    <!-- setmyJob() -->
    <property name="myJob" ref="job" />
</bean>

<bean id="job" class="com.kh.testSpring.job.model.vo.Job" />
```

Spring DI 실습

KH정보교육원

MyPersonTest 생성

생성된 xml 을 확인 한 뒤, 다음과 같은 내용을 추가한다.

```
public class MyPersonTest {
public static void main(String[] args) {
// 1. IoC Container 생성
ApplicationContext context = new
GenericXmlApplicationContext("config/beans.xml");

// 2. Person 객체 생성
Person p1 = (Person) context.getBean("person1");

// 3. Job객체 가져오기
Job myJob = context.getBean("job", Job.class);

// 4. 생성한 객체를 싱글톤 패턴으로 가져옴을 확인
Person p2= (Person) context.getBean("person2", Person.class);
}
}
```