

The Spring logo is a red arrow pointing to the right, with the word "Spring" written in white inside it.

Spring

DI & IoC 개요

Decorative graphic element consisting of several thin, curved lines in dark blue and light grey, originating from the bottom left corner and extending upwards and to the right.

권기웅

1.OOP 개요

1.1 정보처리기사

12.3, 11.8, 11.6, 10.3, 09.8, 09.3, 08.9, 08.5, 08.3, 04.5, 03.3, 02.9, 00.7, 00.3

4

바람직한 설계의 특징

- 설계는 소프트웨어 구조*를 나타내야 한다.
- 설계는 독립적인 기능적 특성을 가진 요소(모듈)로 구성되어야 한다.
- 설계는 모듈 구조, 즉 특정 기능 또는 부기능을 수행하는 논리적 요소들로 분리되는 구조를 가져야 한다.
- 소프트웨어 요소(모듈) 간의 효과적인 제어를 위해 설계에서 계층적 자료 조직이 제시되어야 한다.
- 설계는 자료와 프로시저에 대한 분명하고 분리된 표현을 포함해야 한다.
- 설계는 모듈 간과 외부 개체 간의 연결 복잡성을 줄이는 인터페이스를 가져야 한다.
- 설계는 요구사항 분석에서 얻어진 정보를 이용하여 반복적인 방법으로 이루어져야 한다.
- 요구사항을 모두 구현해야 하고, 유지보수가 용이해야 한다.
- 적당한 모듈의 크기를 유지하고, 모듈 간의 상관성(결합도)은 낮추고, 응집도는 높인다.
- 전체적이고 포괄적인 개념을 설계한 후 차례대로 세분화하여 구체화시켜 나간다.

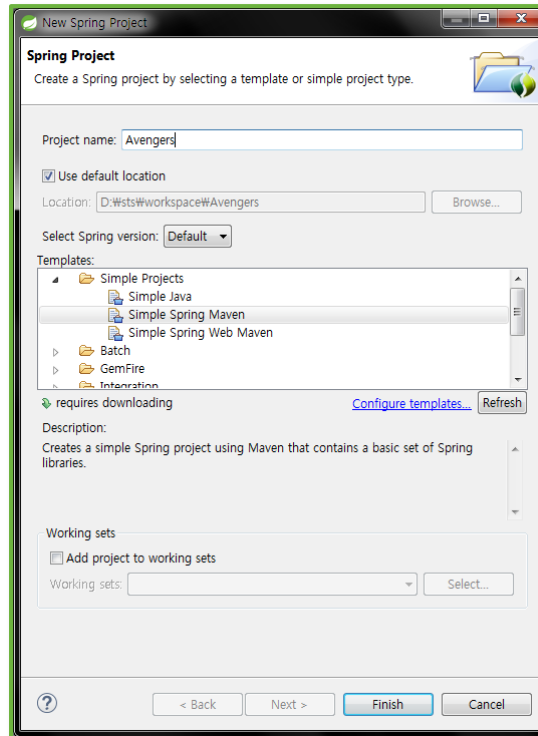
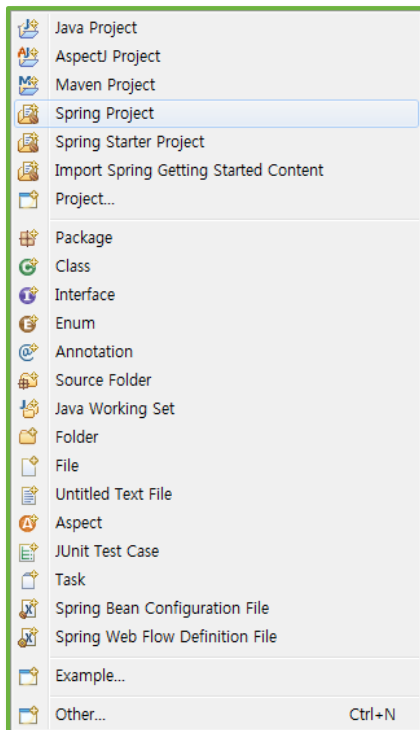
1.2 설계사항

- 어벤져스 : 아이언맨, 스파이더맨
- 어벤져스는 공격
- 아이언맨은 빔, 스파이더맨은 거미줄
- 아이언맨이 공격하는 프로젝트 작성
- 근데 스파이더맨이 공격하는걸로 수정

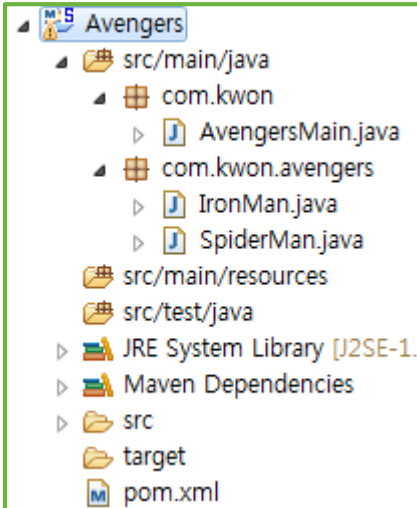


2. 저질 버전

2.1 프로젝트 작성



2.2 클래스 작성



```
public class IronMan {  
    public void attackBeam() {  
        System.out.println("빔 발사");  
    }  
}
```

```
public class SpiderMan {  
    public void attackWeb() {  
        System.out.println("거미줄 발사");  
    }  
}
```

```
public class AvengersMain {  
    public static void main(String[] args) {  
        IronMan i = new IronMan();  
        i.attackBeam();  
    }  
}
```



2.3 수정하면

```
public class AvengersMain {  
    public static void main(String[] args) {  
        SpiderMan s = new SpiderMan();  
        s.attackWeb();  
    }  
}
```

2.4 문제점

- AvengersMain 에서 IronMan, SpiderMan 의 구현에 강하게 의존 – 모듈간의 상관성(결합도) 높음
- 결합도가 높다는 것은 의존하고 있는 클래스(IronMan or SpiderMan)에 변경이 생기면 코드 수정 범위가 넓어질 가능성 높아진다는 얘기
- 공격 메소드가 구현되고 있다는 보장이 없음
- IronMan=>SpiderMan 으로 바꾸면 객체 생성하는 코드 수정해야
- IronMan=>SpiderMan 으로 바꾸면 공격 메소드 호출하는 코드 수정해야

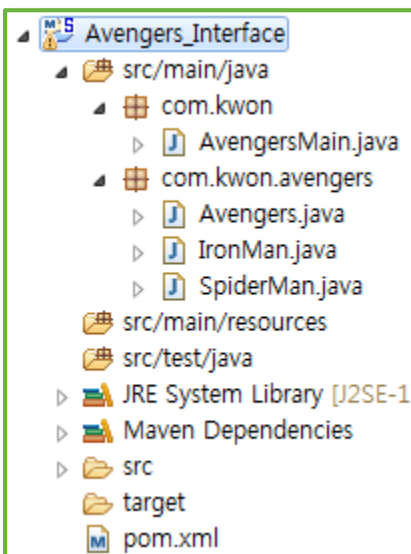
유지보수 불편



3. 개선된 저질 버전

3.1 프로젝트 작성

3.2 클래스 작성



```
public interface Avengers {  
    public abstract void attack();  
}
```

```
public class IronMan implements Avengers {  
    @Override  
    public void attack() {  
        // TODO Auto-generated method stub  
        System.out.println("빔 발사");  
    }  
}
```



```
public class SpiderMan implements Avengers{
    @Override
    public void attack() {
        // TODO Auto-generated method stub
        System.out.println("거미줄 발사");
    }
}
```

```
public class AvengersMain {
    public static void main(String[] args) {
        Avengers a = new IronMan();
        a.attack();
    }
}
```

3.3 수정하면

```
public class AvengersMain {
    public static void main(String[] args) {
        Avengers a = new SpiderMan();
        a.attack();
    }
}
```

3.4 문제점

- interface 를 사용하여, 공격 메소드 구현을 보장할 수 있음
- IronMan=>SpiderMan 으로 바뀌어도 메소드 호출하는 코드는 수정할 필요 없음

결합도 낮아짐

남아있는 문제점 - 객체 생성시 클래스명은 바꿔줘야 함



4.Spring 으로 개선

4.1 DI

Dependency Injection(의존성 주입)

4.1.1 Dependency

- 어떤 클래스가 자신의 임무를 다하기 위해 필요한 값(멤버 변수 등)
- 다른 클래스와의 관계(is a, has a)

4.1.2 Injection

주입

4.2 IoC

Inversion of Control(제어의 역전)

- 프로그램으로 프로그램을 제어하는 게 아닌 XML 파일 등 다른 것으로 프로그램을 제어
- 스프링은 멤버변수/관계를 자바코드를 통하지 않고 xml 파일에 넣은 값을 통해 주입 가능

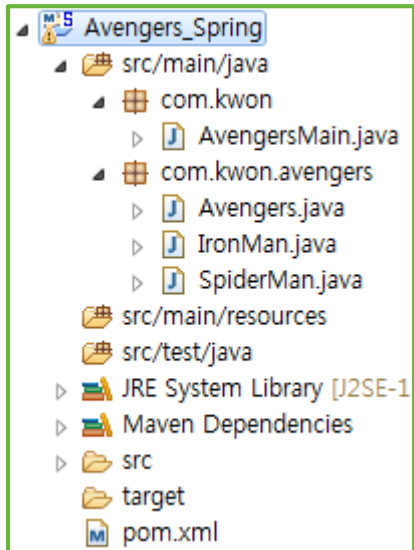
코드를 수정하지 않아도 값을 바꿀 수 있음

컴파일을 다시 하지 않아도 됨

유지보수 용이



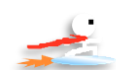
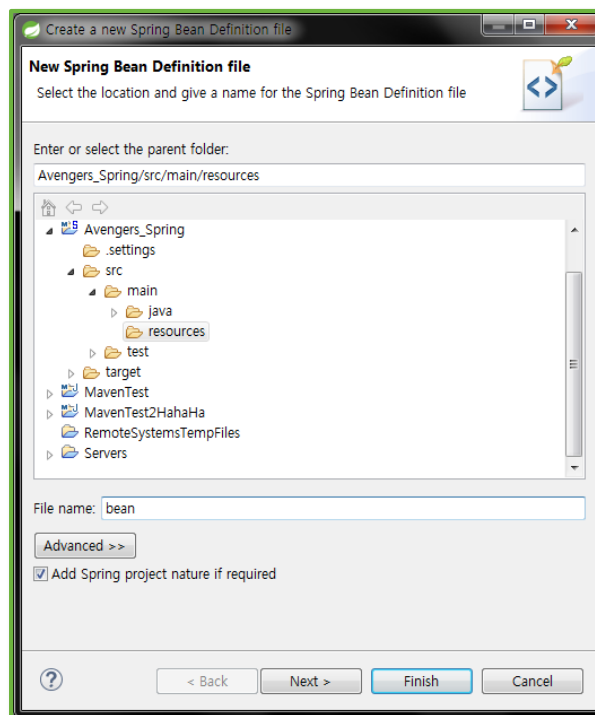
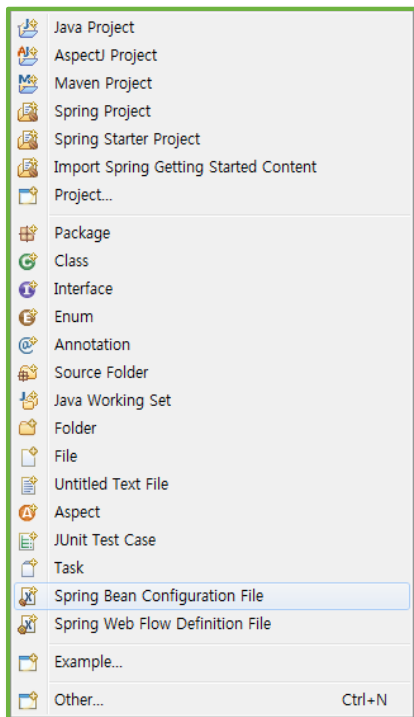
4.3 프로젝트 작성 & 클래스는 그대로 복사



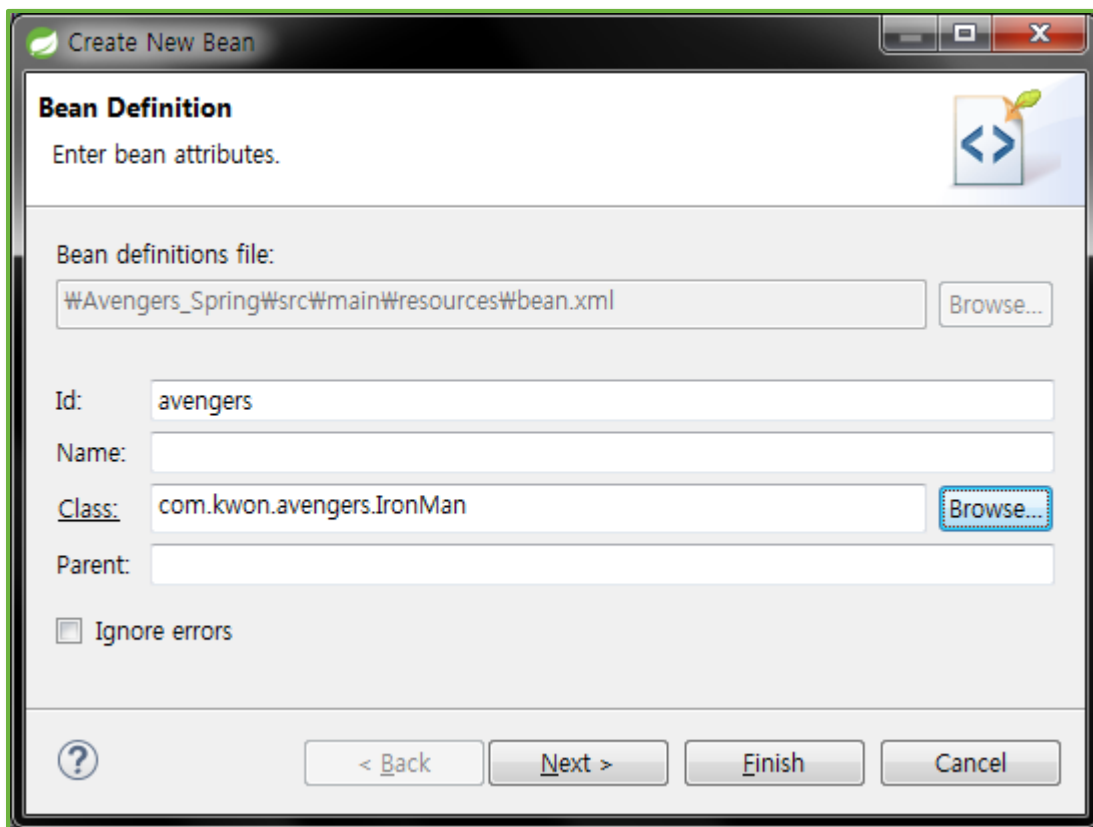
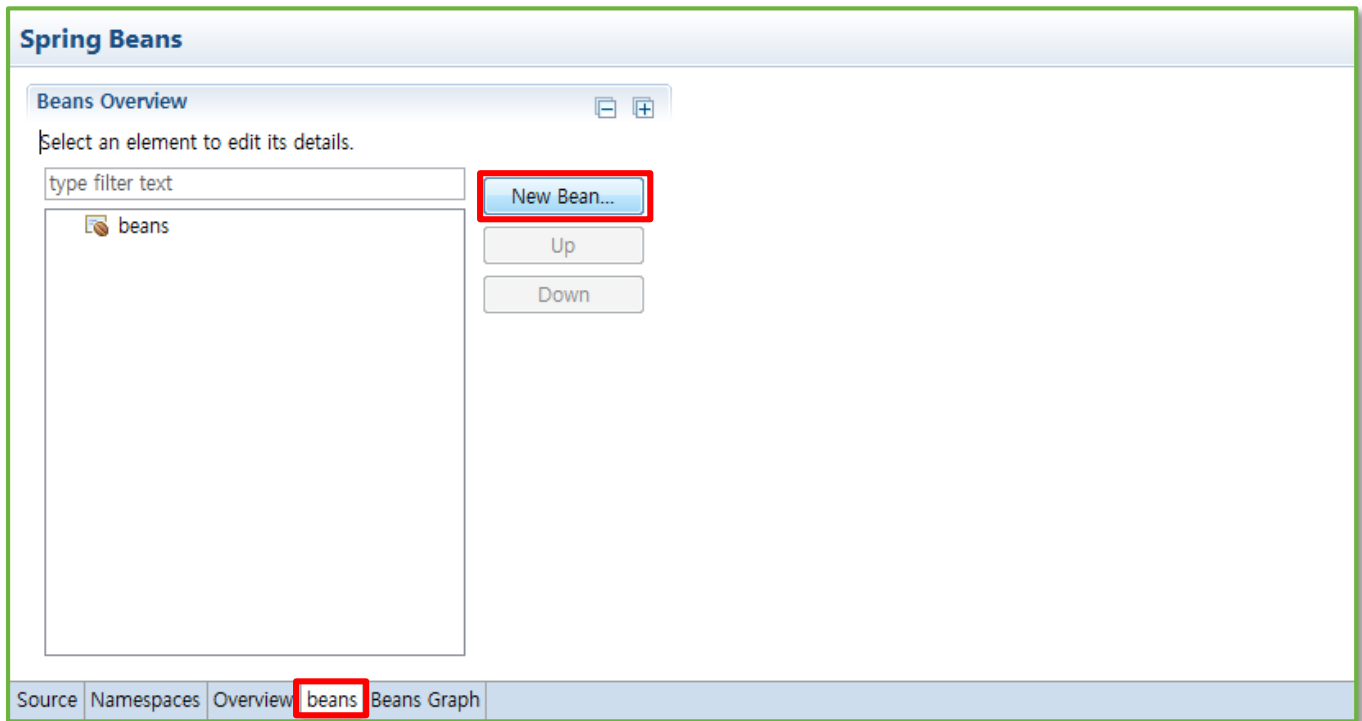
4.4 Spring Bean Configuration File

4.4.1 Spring Bean Configuration File 추가

src/main/resources 에||



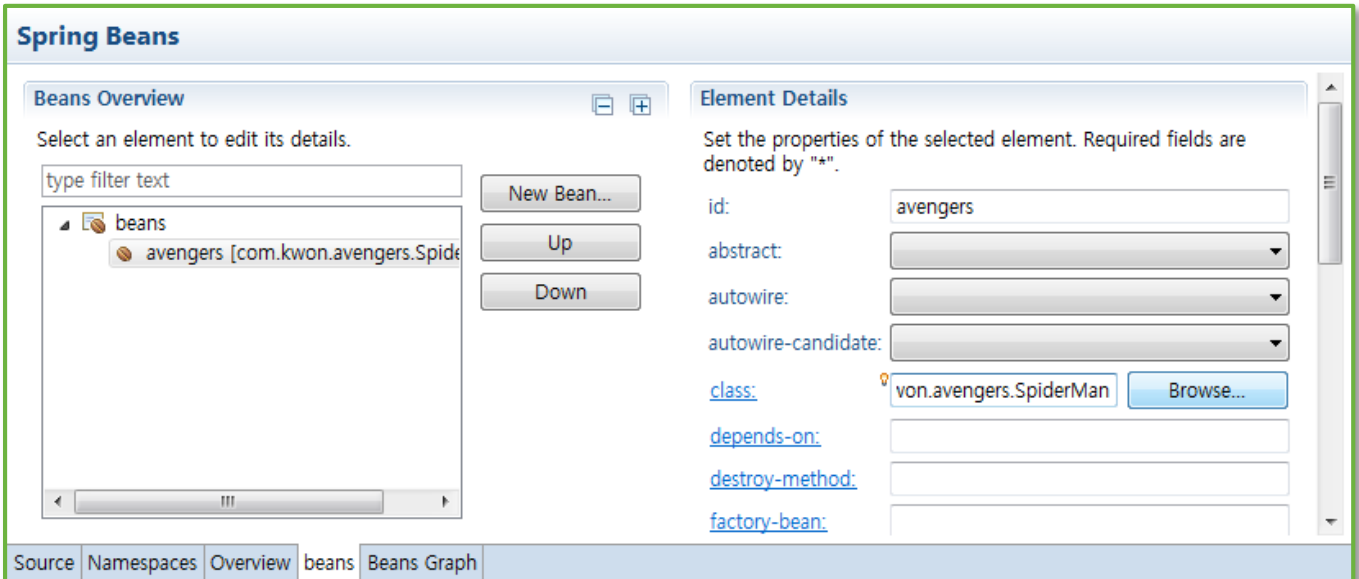
4.4.2 Bean 만들기



4.5 AvengersMain 클래스

```
public class AvengersMain {  
    public static void main(String[] args) {  
        DefaultListableBeanFactory dlbf = new DefaultListableBeanFactory();  
        XmlBeanDefinitionReader xbdr = new XmlBeanDefinitionReader(dlbf);  
        xbdr.loadBeanDefinitions(new ClassPathResource("bean.xml"));  
  
        // bean id  
        Avengers a = dlbf.getBean("avengers", Avengers.class);  
        a.attack();  
    }  
}
```

4.6 수정하면



The screenshot shows the Spring Beans IDE interface. On the left, the 'Beans Overview' pane displays a tree structure with 'beans' expanded, showing 'avengers [com.kwon.avengers.Spide]'. Below this is a search filter 'type filter text'. To the right of the tree are buttons for 'New Bean...', 'Up', and 'Down'. The 'Element Details' pane on the right shows the configuration for the selected 'avengers' bean. It includes fields for 'id' (set to 'avengers'), 'abstract' (dropdown), 'autowire' (dropdown), 'autowire-candidate' (dropdown), 'class' (set to 'von.avengers.SpiderMan' with a 'Browse...' button), 'depends-on' (text field), 'destroy-method' (text field), and 'factory-bean' (text field). At the bottom, there are tabs for 'Source', 'Namespaces', 'Overview', 'beans' (selected), and 'Beans Graph'.



4.7 해결점

- 자바코드에 IronMan, SpiderMan 클래스명 사용하지 않음
- IronMan -> SpiderMan 전환 때 자바코드 수정 없이 XML 파일 수정으로 전환

결합도 더 낮아짐

컴파일을 다시 하지 않아도 됨

유지보수 용이

