

# 1장. 스프링(Spring) 프레임 워크 개발 환경구축



*스프링(Spring)*



# 스프링 프레임워크

- 프레임워크란?

소프트웨어 관점에서 접근하면 “아키텍처에 해당하는 골격 코드”이다.  
사용 방법을 미리 정해놓은 라이브러리 라고도 할 수 있다.

시스템을 개발하는 과정에서 개발자에게 모든 것을 위임하는 것이 아니라,  
애플리케이션의 기본 아키텍처는 프레임워크가 제공하고, 그 뼈대와 살을 붙이는 작업  
만 개발자가 하는 것임.

- 스프링 프레임워크의 특징

- ① 경량(LightWeight)

스프링은 여러 개의 모듈로 구성되어 있으며, 각 모듈은 하나 이상의 jar파일로 구성  
되어 있다. 이 몇 개의 jar 파일만 있으면 개발과 실행이 모두 가능하다.  
애플리케이션 배포 역시 매우 빠르고 쉽다.



# 스프링 프레임워크

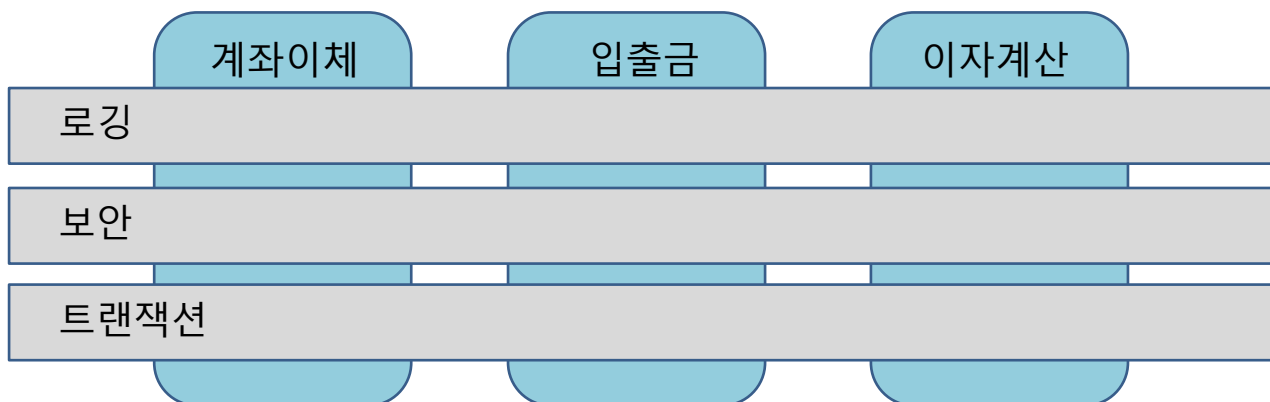
## ② 제어의 역행(Inversion of Control)

비즈니스 컴포넌트를 개발할 때 중요한 점이 낮은 결합도와 높은 응집도이다.

기존의 Control은 객체와 객체 사이의 의존 관계를 개발자가 직접 자바 코드로 처리했으나 IoC가 적용되면 객체 생성을 컨테이너가 대신 처리한다.

## ③ 관점지향 프로그래밍(Asspect Oriented Programming, AOP)

비즈니스 메소드를 개발할 때, 핵심 비즈니스 로직과 각 비즈니스 메소드마다 반복해서 등장하는 공통 로직을 분리함으로써 응집도가 높게 개발하는 것을 말한다.






# 개발 환경 구축

## ◆ JDK 1.8 설치

### Java SE Development Kit 8u202

This software is licensed under the [Oracle Binary Code License Agreement for Java SE Platform Products](#)

Solaris x64	85.38 MB	 <a href="#">jdk-8u202-solaris-x64.tar.gz</a>
Windows x86	201.64 MB	 <a href="#">jdk-8u202-windows-i586.exe</a>
Windows x64	211.58 MB	 <a href="#">jdk-8u202-windows-x64.exe</a>



# 톰캣(Tomcat) 설치

## ◆ 톰캣 설치

8.5.81

Please see the [README](#) file for packaging information. It explains what

**Binary Distributions**

- Core:
  - [zip](#) ([pgp](#), [sha512](#))
  - [tar.gz](#) ([pgp](#), [sha512](#))
  - [32-bit Windows zip](#) ([pgp](#), [sha512](#))
  - [64-bit Windows zip](#) ([pgp](#), [sha512](#))
  - [32-bit/64-bit Windows Service Installer](#) ([pgp](#), [sha512](#))
- Full documentation:
  - [tar.gz](#) ([pgp](#), [sha512](#))

64-bit Windows.zip 다운로드

로컬 디스크 (C:) > Dev > apache-tomcat-8.5.78

이름	수정한 날짜
bin	2022-04-29 오전 4:44
conf	2022-04-29 오전 6:41
lib	2022-04-29 오전 4:44
logs	2022-04-29 오전 6:41
temp	2022-04-29 오전 4:44
webapps	2022-04-29 오전 7:08
work	2022-04-29 오전 6:41
BUILDING.txt	2022-04-29 오전 4:43
CONTRIBUTING.md	2022-04-29 오전 4:43
LICENSE	2022-04-29 오전 4:43

C드라이브에 압축 풀기



# 이클립스(Eclipse) IDE 설치

## ◆ 이클립스(Eclipse) 설치

2021년 3월 버전 -> Workspace(작업공간) 설정 : C:\SpringWorks

### MORE DOWNLOADS

- Other builds
- Eclipse 2022-06 (4.24)
- Eclipse 2022-03 (4.23)
- Eclipse 2021-12 (4.22)
- Eclipse 2021-09 (4.21)
- Eclipse 2021-06 (4.20)
- Eclipse 2021-03 (4.19)
- Eclipse 2020-12 (4.18)
- Eclipse 2020-09 (4.17)
- Older Versions

### Eclipse IDE 2021-03 R Packages



#### Eclipse IDE for Java Developers

328 MB 399,895 DOWNLOADS

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration



Windows x86\_64  
macOS x86\_64  
Linux x86\_64 | AArch64



#### Eclipse IDE for Enterprise Java and Web Developers

519 MB 397,458 DOWNLOADS

Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data Tools, Maven and Gradle, Git, and more.



Windows x86\_64  
macOS x86\_64  
Linux x86\_64 | AArch64

[Click here](#) to file a bug against Eclipse Web Tools Platform.  
[Click here](#) to file a bug against Eclipse Platform.  
[Click here](#) to file a bug against Maven integration for web projects.  
[Click here](#) to report an issue against Eclipse Wild Web Developer (incubating).

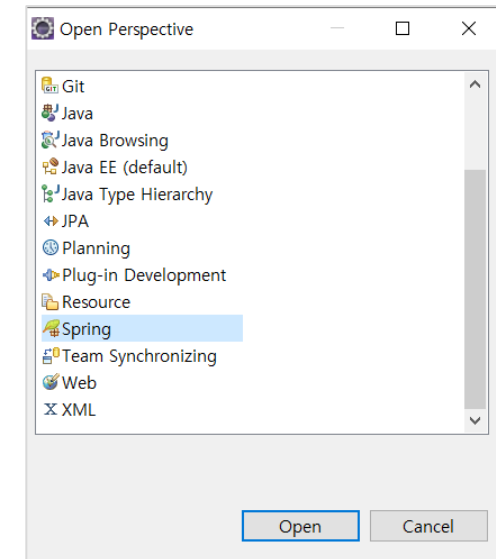
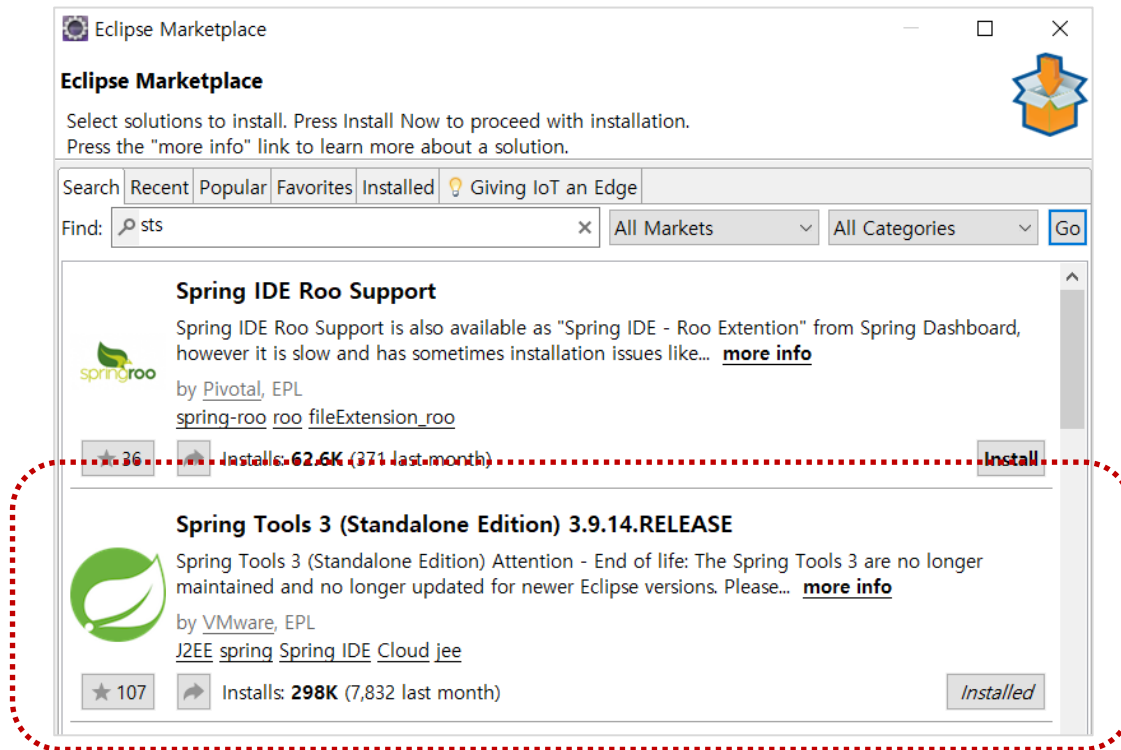


# 스프링 프레임워크

- STS(Spring Tool Suite) 플러그인 설치

이클립스 –[Help] – [Eclipse Marketplace] – sts 검색

## Spring Tools 3(Standalone Edition) 3.9.14 RELEASE - install



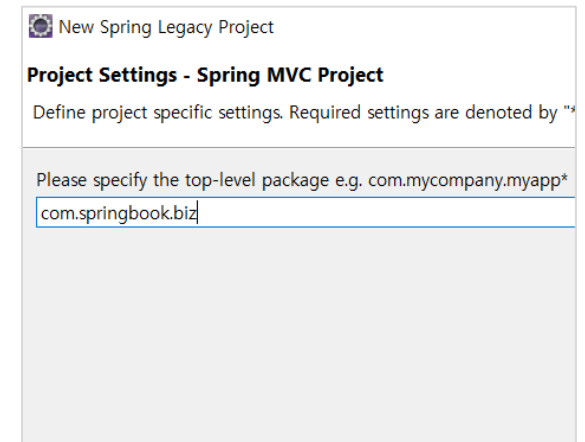
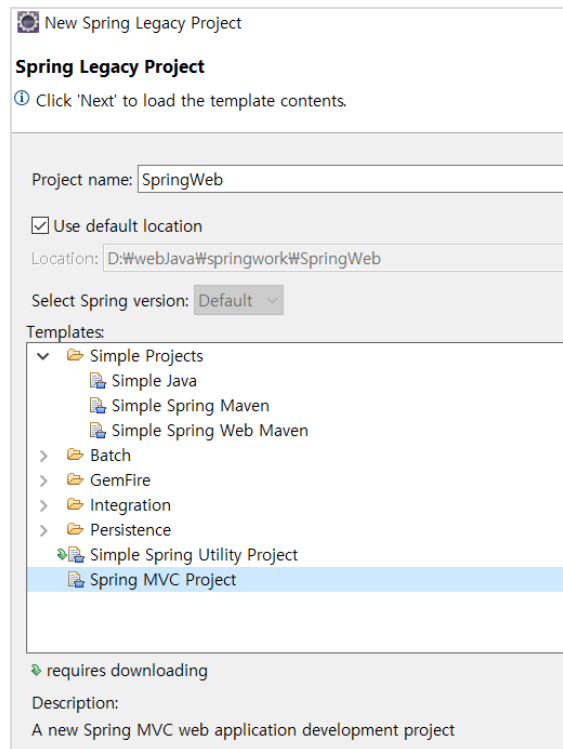
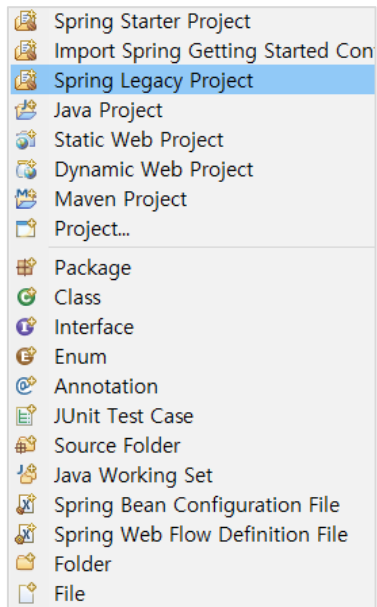
# 스프링 프레임워크

- 실습 프로젝트 생성

[File] -> [New] -> [Spring Legacy Project]

Project name – SpringWeb, Templates – Spring MVC Project

최상위 패키지 – com.springbook.biz



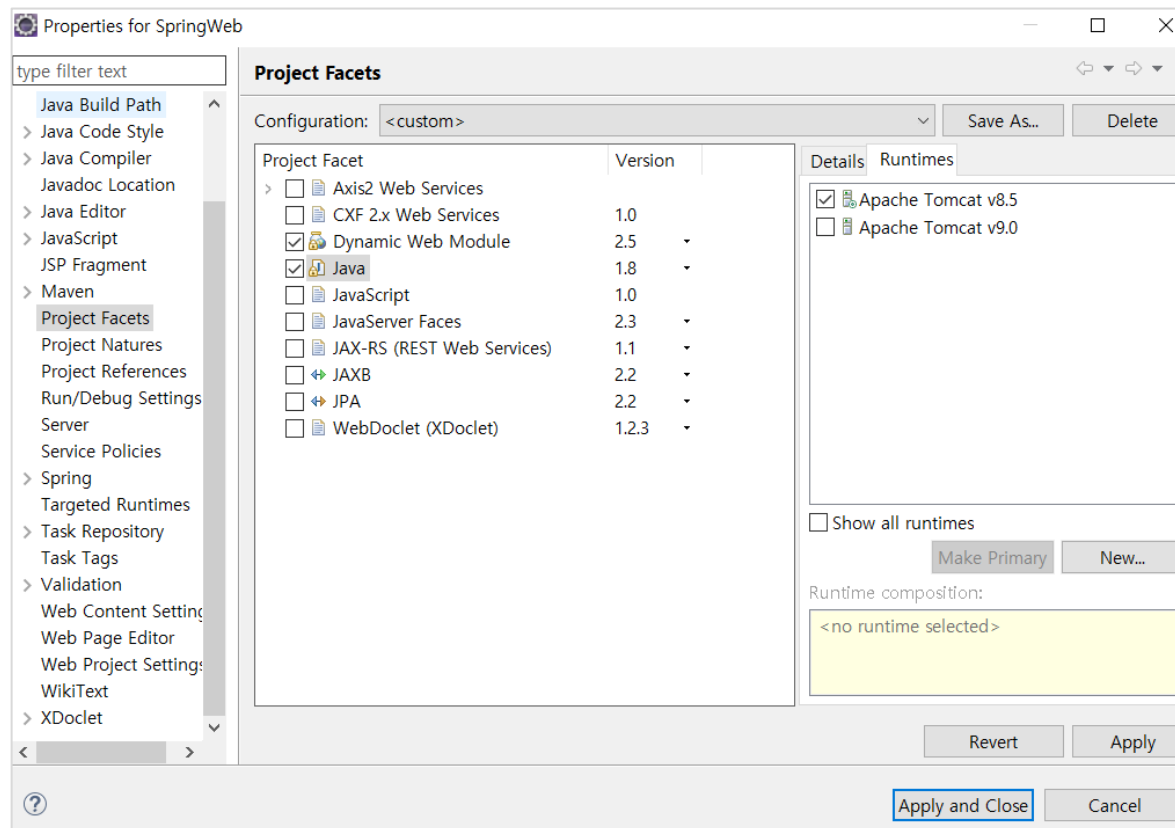


# 스프링 프레임워크

- 프로젝트 설정 변경

SpringWeb -> 우측메뉴 -> properties -> Project Facet -> java 1.8로 변경

Runtimes : Apache Tomcat v8.5 체크 > Apply



# 스프링 프레임워크

- Spring 버전 변경

pom.xml -> 5.2.7 RELEASE로 변경

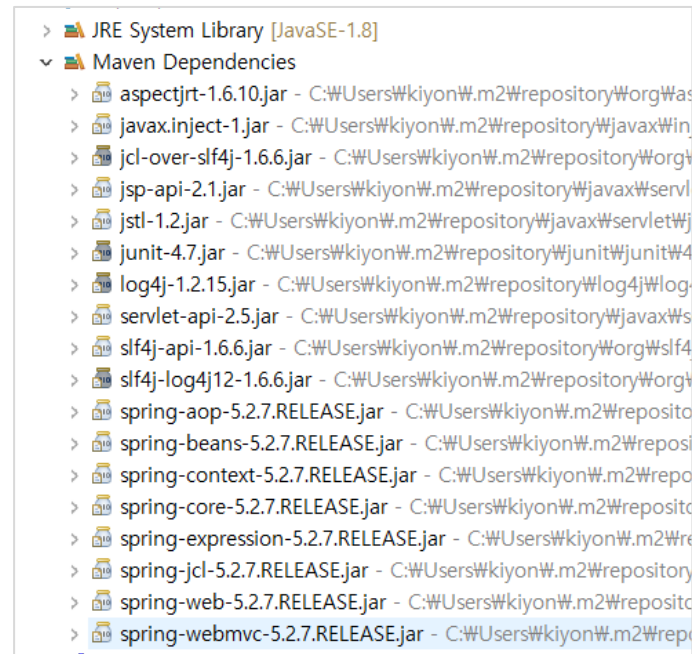
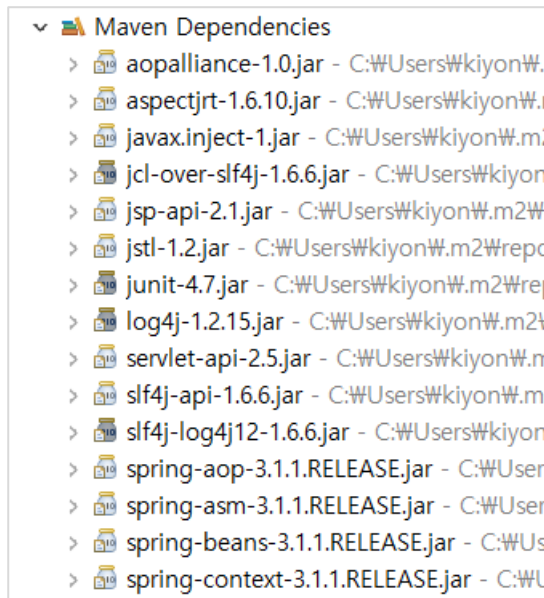
```
<properties>
  <java-version>1.6</java-version>
  <org.springframework-version>5.2.7.RELEASE</org.springframework-version>
  <org.aspectj-version>1.6.10</org.aspectj-version>
  <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
```



# 스프링 프레임워크

- Spring 버전 변경

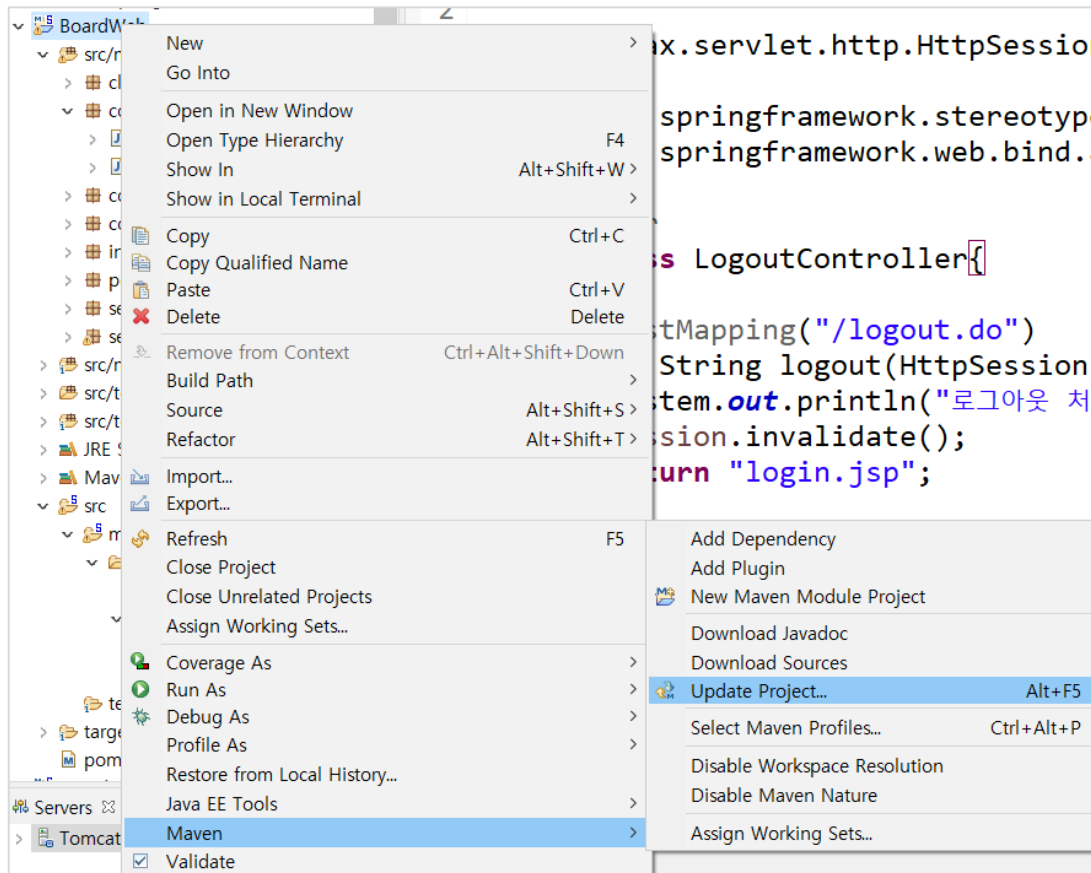
Maven Dependencies 변경전 -> 변경후



# 스프링 프레임워크

- Maven Update

Pom.xml 변경후 > Maven > Update Project



# 스프링 프레임워크

- 결합도가 높은 프로그램 – 예제 : **TV 교체**

결합도란 하나의 클래스가 다른 클래스와 얼마나 많이 연결되어 있는 지를 나타내는 표현이며 결합도가 높은 프로그램은 유지 보수가 어렵다.

```
package product;

public class SamsungTV {

    public void powerOn() {
        System.out.println("SamsungTV--전원 켜다");
    }

    public void powerOff() {
        System.out.println("SamsungTV--전원 끈다");
    }

    public void volumeUp() {
        System.out.println("SamsungTV--소리 올린다");
    }

    public void volumeDown() {
        System.out.println("SamsungTV--소리 내린다");
    }

}
```

```
public class LgTV {

    public void turnOn() {
        System.out.println("LgTV--전원 켜다");
    }

    public void turnOff() {
        System.out.println("LgTV--전원 끈다");
    }

    public void soundUp() {
        System.out.println("LgTV--소리 올린다");
    }

    public void soundDown() {
        System.out.println("LgTV--소리 내린다");
    }

}
```



# 스프링 프레임워크

- 결합도가 높은 프로그램 – 예제 : **TV 교체**

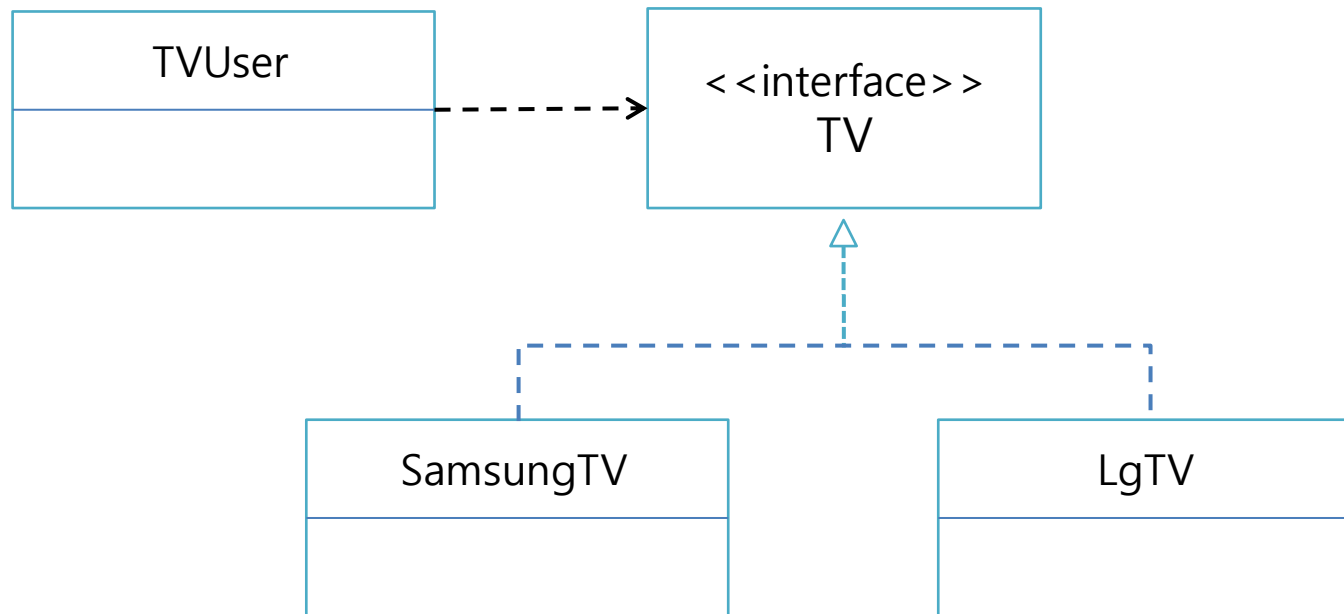
```
public class TVUser {  
  
    public static void main(String[] args) {  
        /*SamsungTV tv = new SamsungTV();  
  
        tv.powerOn();  
        tv.powerOff();  
        tv.volumeUp();  
        tv.volumeDown();*/  
  
        LgTV tv = new LgTV();  
        tv.turnOn();  
        tv.soundUp();  
        tv.soundDown();  
        tv.turnOff();  
        /*  
            SamsungTV와 LgTV는 메소드가 다르므로 TVUser 코드 대부분을 수정해야  
            TV를 교체할 수 있다. TVUser와 같은 프로그램이 여러 개라면 유지보수는  
            더욱 힘들 것이며, TV 교체를 결정하기가 쉽지 않을 것이다.  
        */  
    }  
}
```



# 스프링 프레임워크

- 다형성(Polymorphism) 이용하기 – 예제 : **TV 교체**

결합도를 낮추기 위한 방법 - 다형성을 이용하려면 상속과 메소드 재정의, 형변환이 필요하다.



# 스프링 프레임워크

- 다형성(Polymorphism) 이용하기 – 예제 : **TV 교체**

결합도를 낮추기 위한 방법 - 다형성을 이용하려면 상속과 메소드 재정의, 형변환이 필요하다.

```
package com.spring.polymorphism;

public interface TV {

    public void powerOn();

    public void powerOff();

    public void volumeUp();

    public void volumeDown();

    /*
     인터페이스를 이용하여 모든 TV 클래스가 같은 메소드들을 가질 수밖에 없도록
     강제할 수 있게 되었다.
    */
}
```





# 스프링 프레임워크

- 다형성(Polymorphism) 이용하기 – 예제 : TV 교체

```
public class SamsungTV implements TV{

    @Override
    public void powerOn() {
        System.out.println("SamsungTV__전원 켜다");
    }

    @Override
    public void powerOff() {
        System.out.println("SamsungTV__전원 끈다");
    }

    @Override
    public void volumeUp() {
        System.out.println("SamsungTV--소리 올린다");
    }

    @Override
    public void volumeDown() {
        System.out.println("SamsungTV--소리 내린다");
    }
}
```



# 스프링 프레임워크

- 다형성(Polymorphism) 이용하기 – 예제 : TV 교체

```
public class LgTV implements TV{

    @Override
    public void powerOn() {
        System.out.println("LgTV__전원 켜다");
    }

    @Override
    public void powerOff() {
        System.out.println("LgTV__전원 끈다");
    }

    @Override
    public void volumeUp() {
        System.out.println("LgTV--소리 올린다");
    }

    @Override
    public void volumeDown() {
        System.out.println("LgTV--소리 내린다");
    }
}
```



# 스프링 프레임워크

- 다형성(Polymorphism) 이용하기 – 예제 : **TV 교체**

```
public class TVUser {  
  
    public static void main(String[] args) {  
  
        TV tv = new LgTV(); //SamsungTV()로 교체  
        tv.powerOn();  
        tv.volumeUp();  
        tv.volumeDown();  
        tv.powerOff();  
  
        /*  
        TVUser 클래스는 TV 인터페이스 타입의 변수로 LgTV 객체를 참조하고 있다.  
        이렇게 묵시적 형변환을 이용하여 객체를 참조하면 쉽게 교체할 수 있다.  
        */  
    }  
}
```



# 스프링 프레임워크

- 디자인 패턴 이용하기 – 예제 : **TV 교체**

인터페이스 역시 TV 클래스 소스 수정을 해야만 한다.

소스 수정을 하지 않고 TV를 교체할 수 있다면 유지보수는 더욱 편리해질 것이다.

**Factory 패턴**은 클라이언트에서 사용할 객체 생성을 캡슐화하여 TVUser와 TV 사이를 느슨한 결합 상태로 만들어 준다.

```
public class BeanFactory {  
  
    public Object getBean(String beanName) {  
        if(beanName.equals("samsung")) {  
            return new SamsungTV();  
        }else if(beanName.equals("lg")) {  
            return new LgTV();  
        }  
        return null;  
    }  
    /*  
    getBean() 메소드는 매개변수로 받은 beanName에 해당하는  
    객체를 생성하여 리턴한다.  
    */  
}
```



# 스프링 프레임워크

- 디자인 패턴 이용하기 – 예제 : **TV 교체**

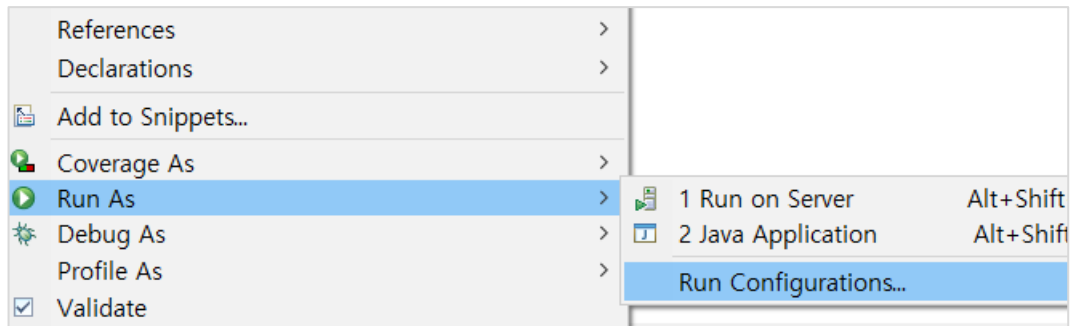
```
public class TVUserGetBean {  
    public static void main(String[] args) {  
        BeanFactory factory = new BeanFactory();  
        TV tv = (TV) factory.getBean(args[0]);  
        tv.powerOn();  
        tv.volumeUp();  
        tv.volumeDown();  
        tv.powerOff();  
    }  
}
```

```
LgTV__전원 켜다  
LgTV--소리 올린다  
LgTV--소리 내린다  
LgTV__전원 끈다
```

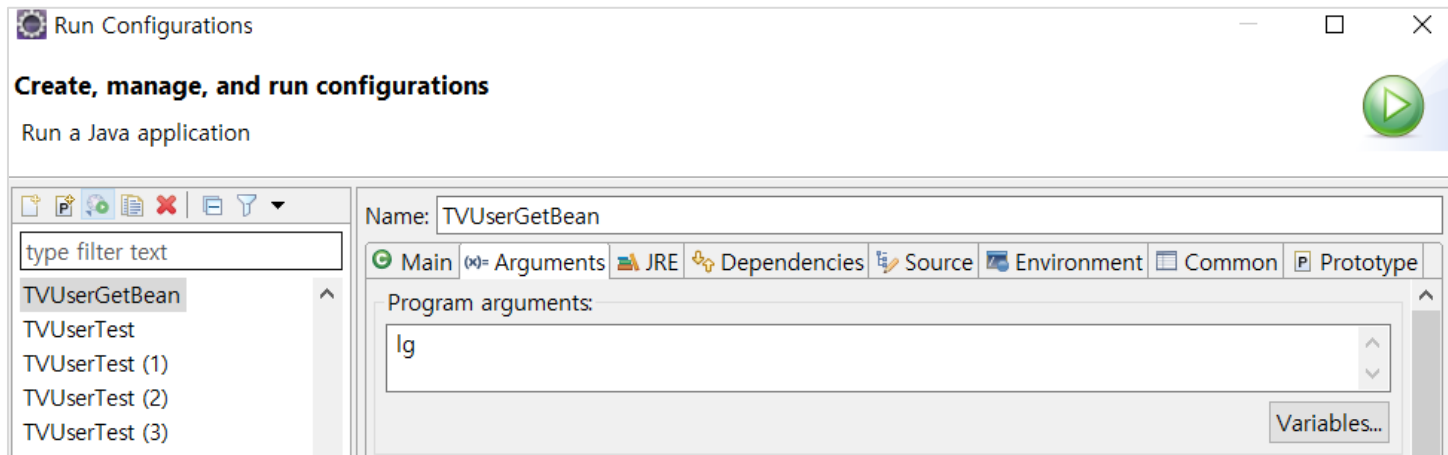


# 스프링 프레임워크

## ▪ Run Configuration으로 실행하기



**[Argument]** 탭 - 명령행 매개변수에 'lg' 나 'samsung'을 넘겨줌

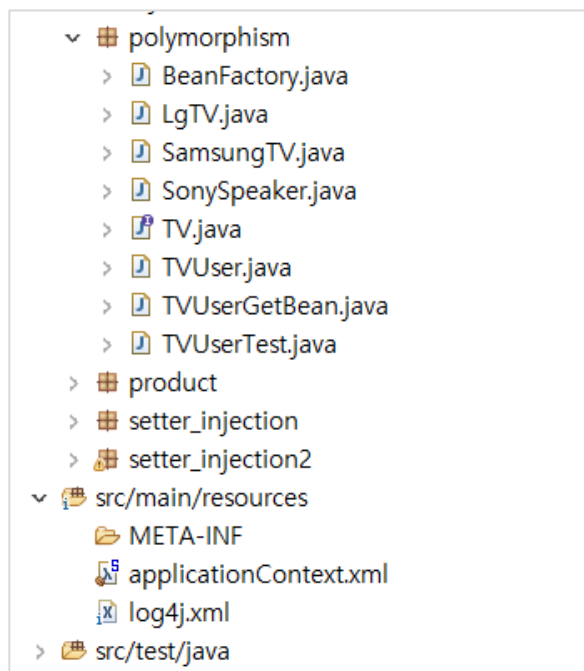
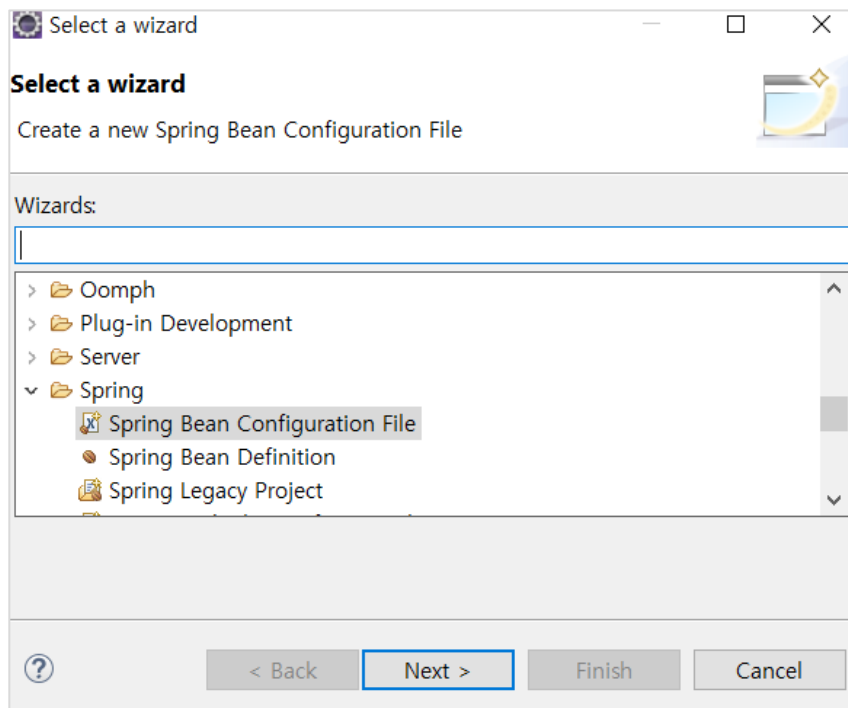


# 스프링 프레임워크

## ■ 스프링 컨테이너 및 설정 파일

**src/main/resources > applicationContext.xml**

클래스 소스를 수정하지 않고 설정 파일에서 빈 등록을 통하여 TV를 교체할 수 있음



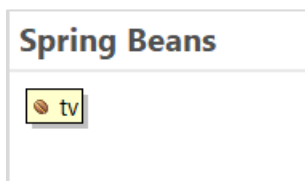
# 스프링 프레임워크

- 스프링 컨테이너 및 설정 파일

src/main/resources > applicationContext.xml

```
<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.sp
    http://www.springframework.org/schema/context http://www.springframework.

<!-- 빈 설정(객체) : SamsungTV -->
<bean id="tv" class="com.spring.polymorphism.SamsungTV" />
```



bean을 등록하지 않았을때 오류

" [org.springframework.beans.factory.NoSuchBeanDefinitionException](#): No bean named 'tv' available  
ework.beans.factory.support.DefaultListableBeanFactory.getBeanDefinition([DefaultListableBeanFact](#)





# 스프링 프레임워크

- 스프링 XML 설정

- <beans> 루트 엘리먼트**

- <beans> 엘리먼트 시작 태그에 네임스페이스를 비롯한 XML 스키마 관련 정보가 설정된다.

- <bean> 엘리먼트**

- 스프링 설정 파일에 클래스를 등록하려면 <bean> 엘리먼트를 사용한다.

- <bean> 객체를 위한 이름을 지정할 때 사용하는 속성이 id이며, class 속성에 클래스 파일 이름을 명시해야 함



# 스프링 프레임워크

## ▪ 구동 순서

```
public class SamsungTV implements TV{  
  
    public SamsungTV() {  
        System.out.println("==> SamsungTV 객체 생성");  
    }  
  
    @Override  
    public void powerOn() {  
        System.out.println("SamsungTV__전원 켜다");  
    }  
}
```

- ① TVUser 클라이언트가 스프링 설정 파일을 로딩하여 컨테이너 구동
- ② 스프링 설정 파일에 <bean> 등록된 SamsungTV 객체 생성
- ③ getBean() 메소드로 이름이 'tv'인 객체를 요청(Lookup)
- ④ SamsungTV 객체 반환



# 스프링 프레임워크

## ■ 스프링 컨테이너 및 설정 파일

```
public class TVUserTest {  
  
    public static void main(String[] args) {  
        //1.Spring 컨테이너를 구동한다.  
        AbstractApplicationContext factory =  
            new GenericXmlApplicationContext("applicationContext.xml");  
  
        //2. Spring 컨테이너로부터 필요한 객체를 요청(Lookup) 한다.  
        TV tv = (TV)factory.getBean("tv");  
        tv.powerOn();  
        tv.volumeUp();  
        tv.volumeDown();  
        tv.powerOff();  
  
        //Spring 컨테이너를 종료한다.  
        factory.close();  
    }  
}
```

==> SamsungTV 객체 생성  
SamsungTV\_\_전원 켜다  
SamsungTV--소리 올린다  
SamsungTV--소리 내린다  
SamsungTV\_\_전원 끈다

