

# 삼성 청년 SW 아카데미

Spring Framework

# Spring DI

- Spring Framework
- 의존관계 역전
- 의존성 주입
- Spring Container Build
- Spring DI
  - ① XML
  - ② Annotation

# Spring Framework

## ✓ Framework?

- 사전적 의미 : (건물 등의) 뼈대, (판단/결정 등을 위한) 틀
- SW에서의 의미 : SW 특정 문제를 해결하기 위해서 상호 협력하는 클래스와 인터페이스의 집합

## ✓ 왜 사용하는가?

- 웹 어플리케이션을 개발하기 위해서는 많은 기본 기능을 설계, 작성해야 한다.  
(요청처리, 세션관리, 리소스 관리, 멀티 쓰레드 등)
- 공통으로 사용되는 기본기능들을 일관되게 사용할 수 있으면 개발자는 웹 어플리케이션 기능 자체 개발에만 신경을 쓰면 되기 때문에 생산성이 높아진다.
- 개발자 입장에서는 완성된 구조에 맡은 기능을 개발하여 넣어주면 되기 때문에 개발 시간을 단축할 수 있다.

### ✓ Spring Framework의 등장

- 90년대 말부터, 웹사이트가 복잡해지면서 엔터프라이즈 급 서비스가 필요하게 되었다.
- 엔터프라이즈 어플리케이션을 개발하기 위한 많은 프레임워크들이 만들어지기 시작했다.
- 자바에서는 EJB(Enterprise JavaBeans)를 이용한 엔터프라이즈 급 어플리케이션을 제작이 유행하였다.
- 하지만, EJB를 기반으로 한 어플리케이션은 EJB 스펙에 따른 객체를 작성해야 하거나, 간단한 기능을 작성하고 테스트 할 때도 EJB 서버가 필요해서 개발 효율성이 떨어지는 단점이 있었다.
- 복잡한 구동환경과 하드웨어 구성이 필요하지 않은 경량 프레임워크(light-weight Framework)를 지향하는 방법론이 Rod Johnson 의 'Expert One-on-One J2EE Development without EJB'라는 저서에서 제시되었고 Spring Framework에 토대가 되었다.

## ✓ Spring Framework의 특징

- POJO( Plain Old Java Object) 방식의 프레임워크
- 의존성 주입(Dependency Injection)을 통한 객체관계 구성
- 관점지향 프로그래밍(AOP, Aspect Oriented Programming)
- 제어 역전(IoC, Inversion of Control)
- 높은 확장성과 다양한 라이브러리
- ...

### ✓ 왜 Spring Framework를 사용해야 하는가?

- Spring is everywhere

전세계 많은 개발자들이 스프링을 사용하고 있다. (Alibaba, Amazon, Google, Microsoft 등)

- Spring is flexible

유연하고 포괄적인 외부 라이브러리 확장을 통해 다양한 형태의 애플리케이션 개발가능

- Spring is Fast

기본적으로 빠른 시작 / 종료 / 최적화된 실행을 확인할 수 있다.

- Spring is productive

Spring boot는 프로그래밍 접근 방식을 변환하여 작업량을 줄여준다. 또한 애플리케이션 컨텍스트 및 웹서버 등을 결합하여 간단한 프로그래밍을 쉽게 해준다.

- Spring is secure

업계 표준 보안 체계와 쉽게 통합할 수 있고, 기본적으로 신뢰할 수 있는 솔루션을 제공한다.

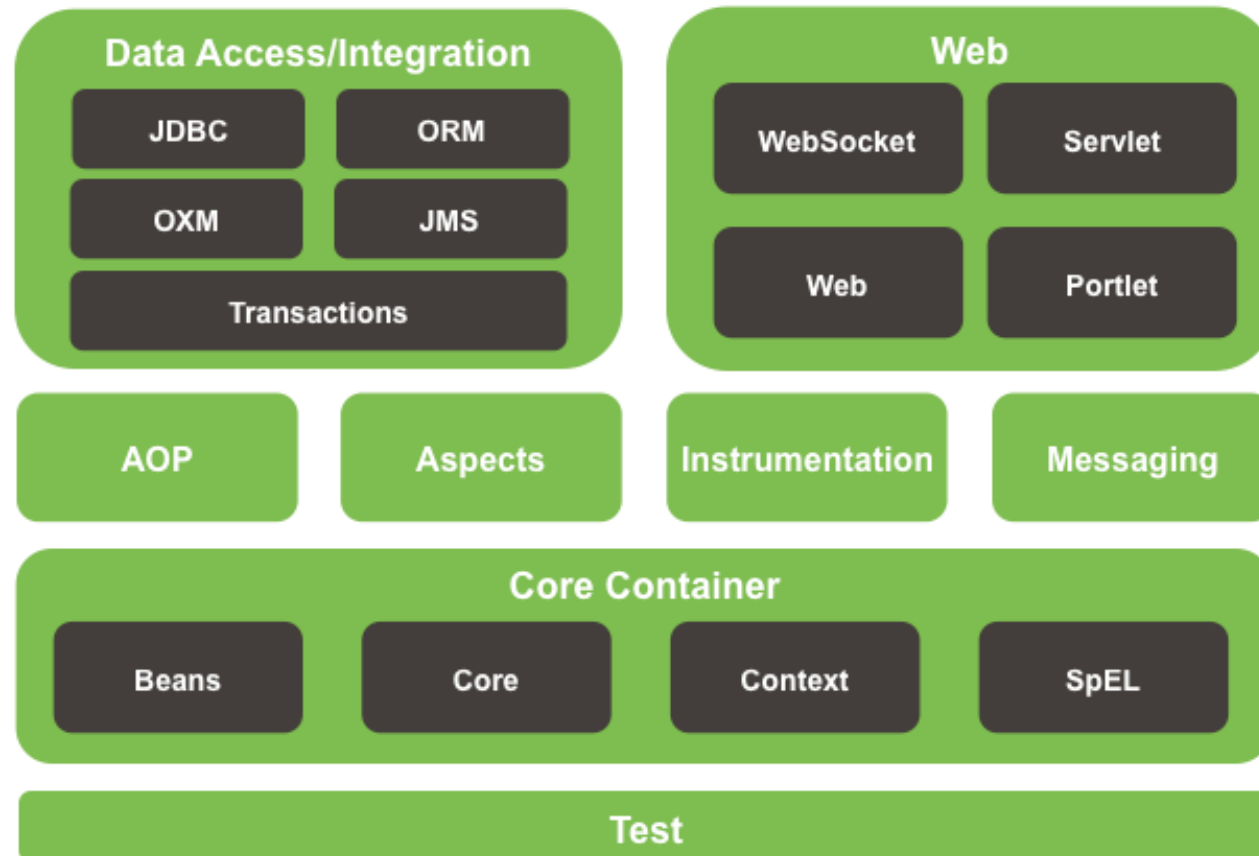
- Spring is supportive

커뮤니티가 잘 발달해 있으며, 빠른 시작, 가이드, 자습서 등의 리소스를 지원하고있다.

## ✓ Spring Framework Architecture



### Spring Framework Runtime

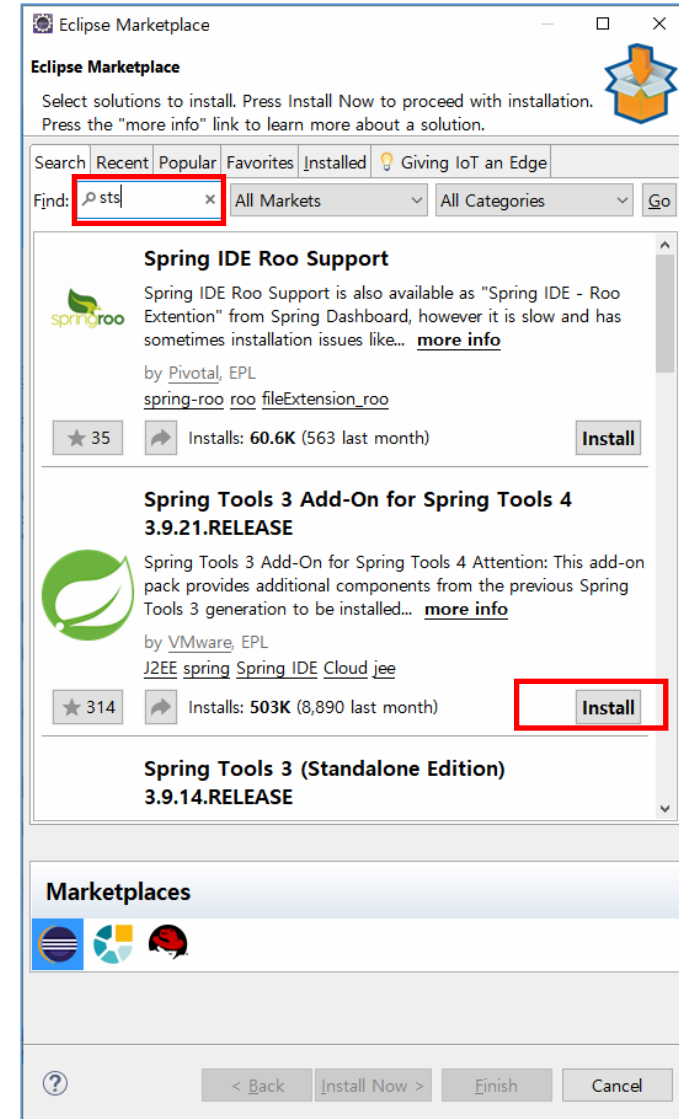
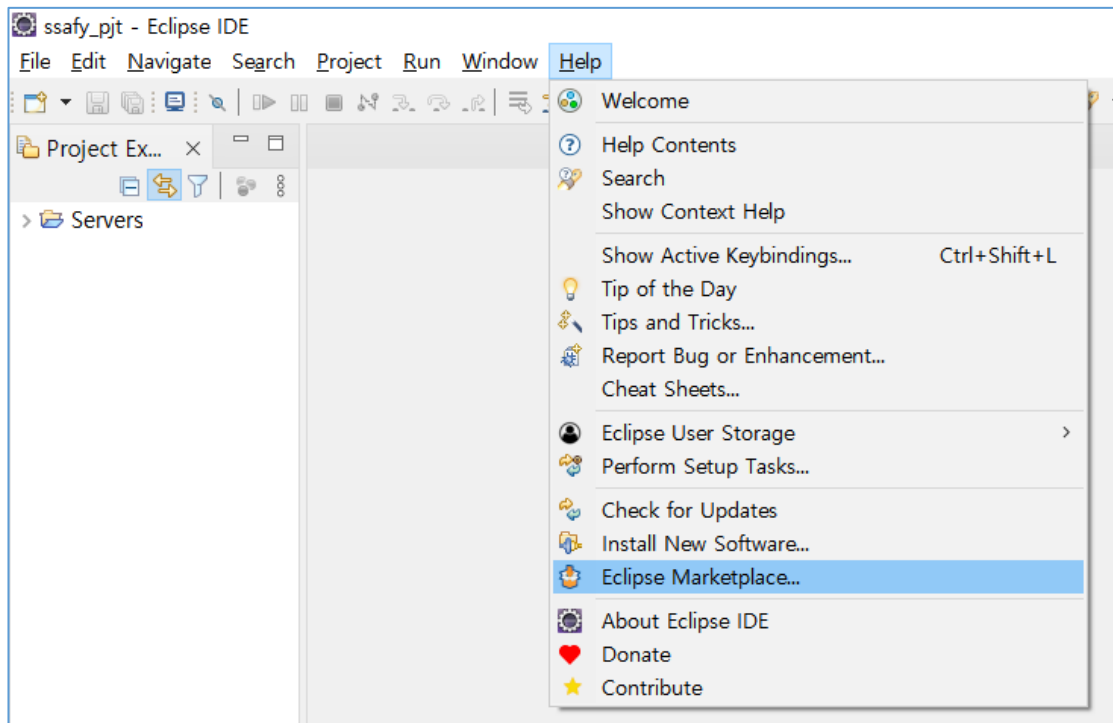




## ✓ STS (Spring Tools Suite) 설치

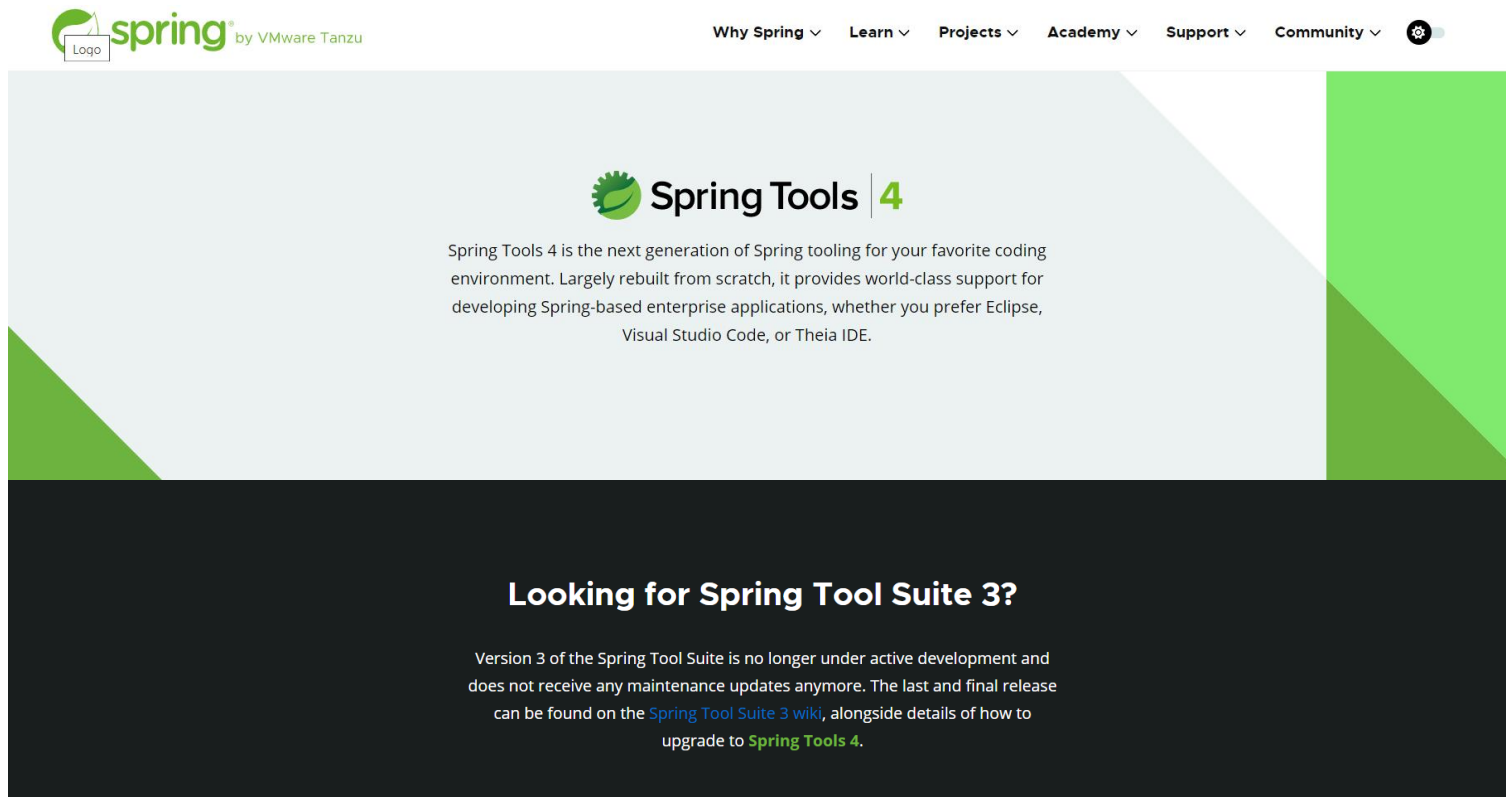
스프링 개발을 위한 개발환경

Help -> Eclipse Marketplace -> sts 검색 -> 설치



## ✓ STS (Spring Tools Suite) 설치

### 스프링 개발을 위한 개발환경



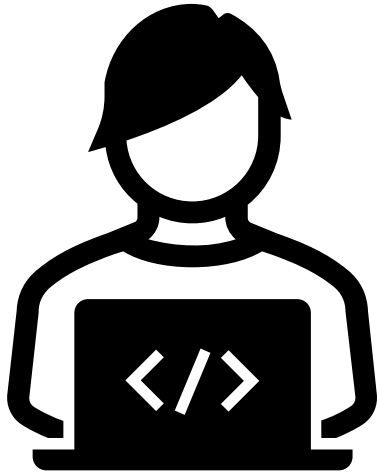
The screenshot shows the Spring Tools 4 website. At the top, there is a navigation bar with links: Why Spring, Learn, Projects, Academy, Support, and Community. The main content area features the Spring Tools 4 logo and a description: "Spring Tools 4 is the next generation of Spring tooling for your favorite coding environment. Largely rebuilt from scratch, it provides world-class support for developing Spring-based enterprise applications, whether you prefer Eclipse, Visual Studio Code, or Theia IDE." Below this, there is a section titled "Looking for Spring Tool Suite 3?" which states: "Version 3 of the Spring Tool Suite is no longer under active development and does not receive any maintenance updates anymore. The last and final release can be found on the Spring Tool Suite 3 wiki, alongside details of how to upgrade to Spring Tools 4."

full distribution on Eclipse 4.15

- [https://download.springsource.com/release/STS/3.9.14.RELEASE/dist/e4.15/spring-tool-suite-3.9.14.RELEASE-e4.15.0-win32-x86\\_64.zip](https://download.springsource.com/release/STS/3.9.14.RELEASE/dist/e4.15/spring-tool-suite-3.9.14.RELEASE-e4.15.0-win32-x86_64.zip)

# 의존관계역전

## ✓ 프로그래머와 컴퓨터(Desktop)



```
public class Programmer {  
    private Desktop computer;  
  
    public Programmer() {  
        computer = new Desktop();  
    }  
  
    public void coding() {  
        System.out.println(computer.getInfo() + "으로 개발을 합니다.");  
    }  
}
```



```
public class Desktop {  
    //필요한 필드...  
  
    //해당 컴퓨터 정보를 반환...  
    public String getInfo() {  
        return "데스크톱";  
    }  
}
```

## ✓ 프로그래머와 컴퓨터(Desktop)

```
public class Test {  
    public static void main(String[] args) {  
        Programmer p = new Programmer();  
        p.coding();  
    }  
}
```

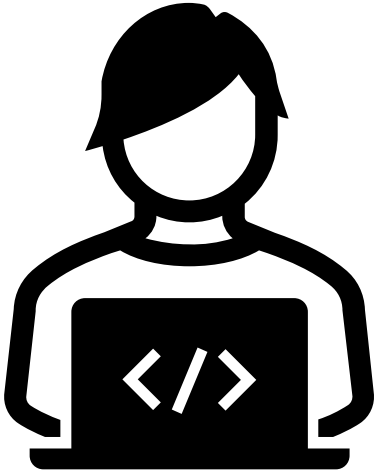
Console

<terminated> Test [Java Application] C:\Program Files\Zulu\zulu-8\bin\javav  
데스크톱으로 개발을 합니다.

ClassA 객체가 어떤 일을 처리하기 위해서 ClassB의 객체의 도움을 받아야만 일을 처리할 수 있다면

‘ClassA는 ClassB에 의존한다.’ 라고 표현

## ✓ 프로그래머와 컴퓨터(Laptop)



Programmer 가 Desktop 클래스에 의존성을 가지고 있기 때문에

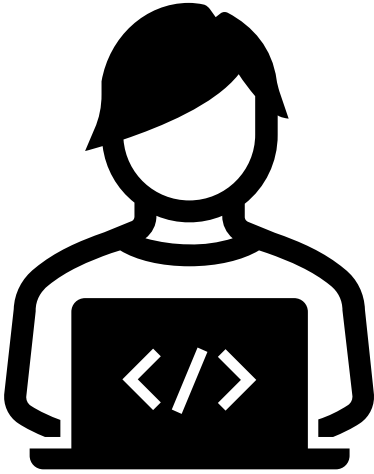


```
public class Programmer {  
    private Desktop computer;  
  
    public Programmer() {  
        computer = new Desktop();  
    }  
  
    public void coding() {  
        System.out.println(computer.getInfo() + "으로 개발을 합니다.");  
    }  
}
```

코드 수정 필요!

```
public class Laptop {  
    // 필요한 필드...  
  
    // 해당 컴퓨터 정보를 반환...  
    public String getInfo() {  
        return "랩톱";  
    }  
}
```

## ✓ 프로그래머와 컴퓨터(객체생성 의존성 제거)

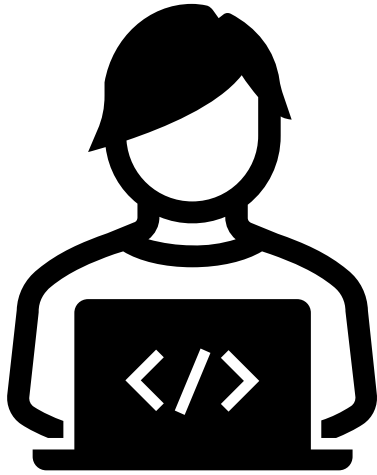


```
public class Programmer {  
    private Desktop computer;  
  
    public Programmer(Desktop desktop) {  
        computer = desktop;  
    }  
  
    public void coding() {  
        System.out.println(computer.getInfo() + "으로 개발을 합니다.");  
    }  
}
```

```
public class Test {  
    public static void main(String[] args) {  
        Desktop desktop = new Desktop();  
        Programmer p = new Programmer(desktop);  
        p.coding();  
    }  
}
```

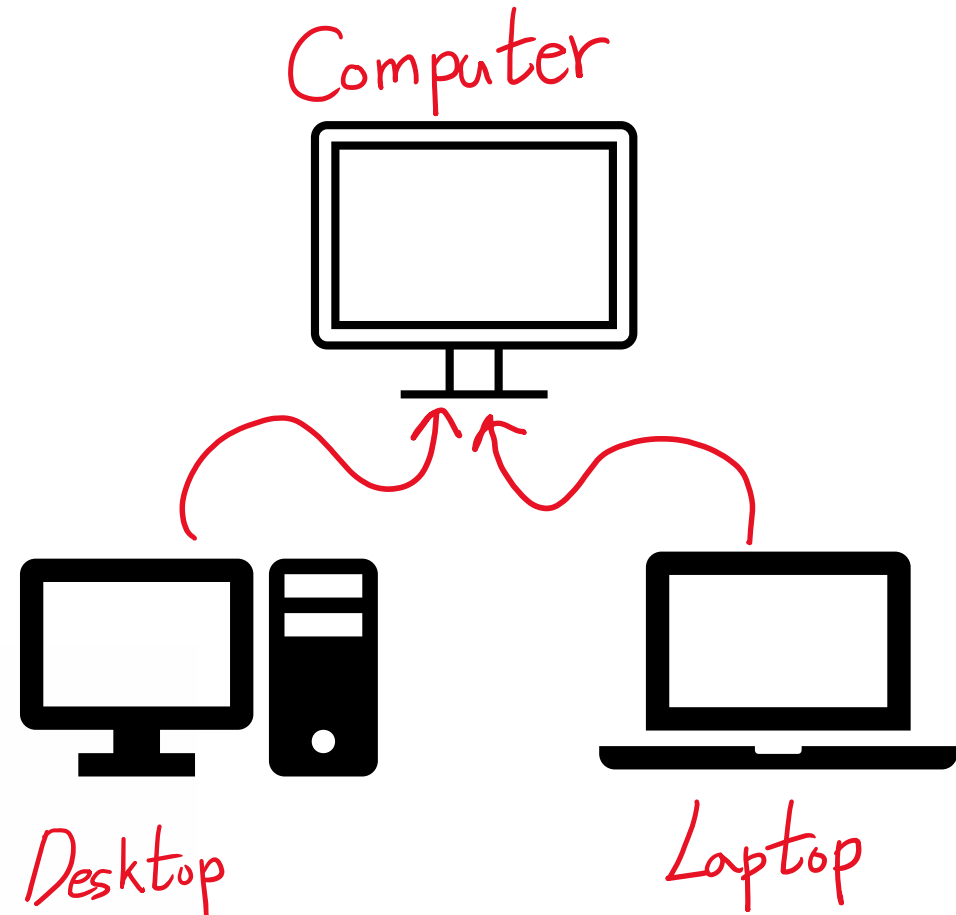
Desktop에 대한 객체생성 의존성  
Programmer → Test 로 (의존관계 역전)

- ✓ 프로그래머와 컴퓨터(타입 의존성 제거)



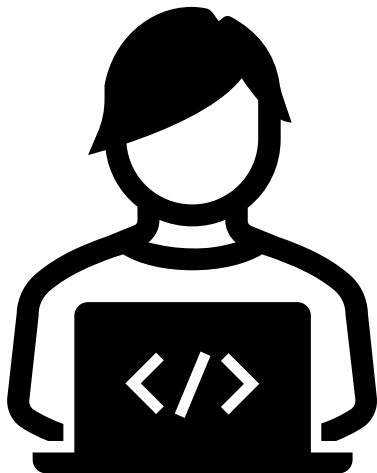
```
public class Programmer {  
    private Desktop computer;  
  
    public Programmer(Desktop desktop) {  
        computer = desktop;  
    }  
  
    public void coding() {  
        System.out.println(computer.getInfo() + "으로 개발을 합니다.");  
    }  
}
```

Laptop 으로 바꾸기?





## ✓ 프로그래머와 컴퓨터(타입 의존성 제거)



```
public class Programmer {  
    private Computer computer;  
  
    public Programmer(Computer computer) {  
        this.computer = computer;  
    }  
  
    public void coding() {  
        System.out.println(computer.getInfo() + "으로 개발을 합니다.");  
    }  
}
```

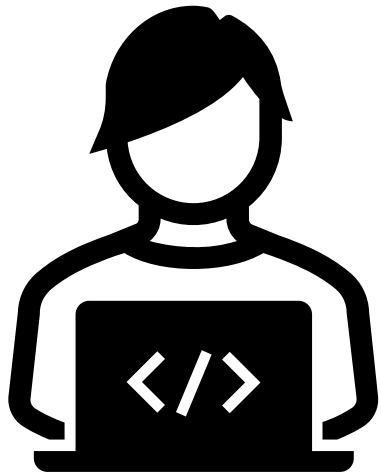
→ 느슨한 결합

```
public class Programmer {  
    private Desktop computer;  
  
    public Programmer(Desktop desktop) {  
        computer = desktop;  
    }  
  
    public void coding() {  
        System.out.println(computer.getInfo() + "으로 개발을 합니다.");  
    }  
}
```

Laptop 으로 바꾸기?

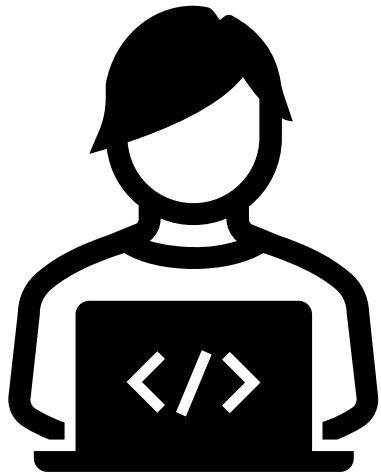
# 의존성 주입

## ✓ 프로그래머 의존성 주입 (생성자 이용)



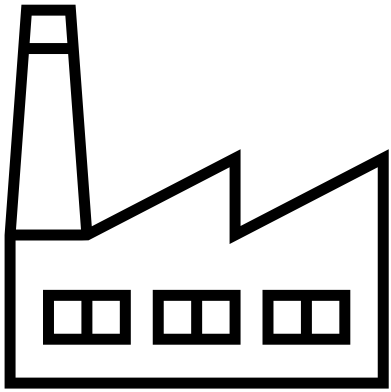
```
public class Programmer {  
    private Computer computer;  
  
    //생성자를 이용한 의존성 주입  
    public Programmer(Computer computer) {  
        this.computer = computer;  
    }  
  
    public void coding() {  
        System.out.println(computer.getInfo() + "으로 개발을 합니다.");  
    }  
}
```

## ✓ 프로그래머 의존성 주입 (설정자 이용)



```
public class Programmer {  
    private Computer computer;  
  
    // setter를 이용한 의존성 주입  
    public void setComputer(Computer computer) {  
        this.computer = computer;  
    }  
  
    public void coding() {  
        System.out.println(computer.getInfo() + "으로 개발을 합니다.");  
    }  
}
```

### ✓ 프로그래머 의존성 주입 (Factory)



Computer  
공장

```
public class ComputerFactory {
    public static Computer getComputer(String type) {
        if (type.equals("D"))
            return new Desktop();
        else if (type.equals("L"))
            return new Laptop();
        return null;
    }
}

public class Test {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        Programmer p = new Programmer();

        Computer computer = ComputerFactory.getComputer(sc.next());
        p.setComputer(computer);
        p.coding();

        Computer computer2 = ComputerFactory.getComputer(sc.next());
        p.setComputer(computer2);
        p.coding();
    }
}
```

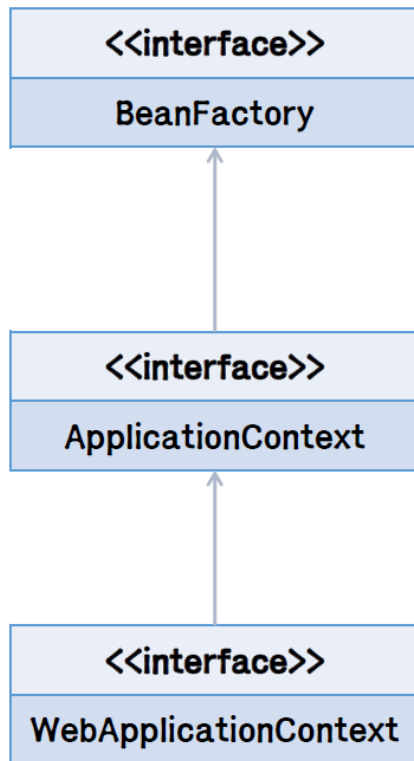
주의요!

# Spring Container Build

## ✓ Spring IoC Container

### ▪ Container?

- 스프링에서 핵심적인 역할을 하는 객체를 Bean이라고 하며,
- Container는 Bean의 인스턴스화 조립, 관리의 역할, 사용 소멸에 대한 처리를 담당한다.



- BeanFactory
  - 프레임워크 설정과 기본기능을 제공하는 컨테이너
  - 모든 유형의 객체를 관리할 수 있는 메커니즘 제공
- ApplicationContext
  - BeanFactory 하위인터페이스
  - 이벤트 처리, 국제화용 메시지 처리, AOP 통합 기능 제공
- WebApplicationContext
  - 웹 환경에서 Spring을 사용하기 위한 기능이 추가됨
  - 대표적인 구현 클래스로 XmlWebApplicationContext가 있음

### ✓ 스프링 설정 정보 (Spring configuration metadata)

- 애플리케이션 작성을 위해 생성할 Bean과 설정 정보, 의존성 등의 방법을 나타내는 정보
- 설정정보를 작성하는 방법은 XML 방식, Annotation 방식, Java 방식이 있다.

설정 정보 예)

```
<?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/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">

  <bean id="..." class="..."> 1 2
    <!-- collaborators and configuration for this bean go here -->
  </bean>

  <bean id="..." class="...">
    <!-- collaborators and configuration for this bean go here -->
  </bean>

  <!-- more bean definitions go here -->

</beans>
```



### ✓ Spring Container 빌드

- Project 생성 후 → Maven Project로 변경
- pom.xml → Spring Context 의존성 추가

일괄적 관리.

```
*Spring_01_DI_02/pom.xml ☒
3  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/maven-v4_0_0.xsd"
4  <modelVersion>4.0.0</modelVersion>
5  <groupId>Spring_01_DI_02</groupId>
6  <artifactId>Spring_01_DI_02</artifactId>
7  <version>0.0.1-SNAPSHOT</version>
8  <build>
9    <sourceDirectory>src</sourceDirectory>
10   <plugins>
11     <plugin>
12       <artifactId>maven-compiler-plugin</artifactId>
13       <version>3.8.0</version>
14       <configuration>
15         <source>1.8</source>
16         <target>1.8</target>
17       </configuration>
18     </plugin>
19   </plugins>
20 </build>
21 <dependencies>
22   <dependency>
23     <groupId>org.springframework</groupId>
24     <artifactId>spring-context</artifactId>
25     <version>5.3.18</version>
26   </dependency>
27 </dependencies>
28 </project>
```

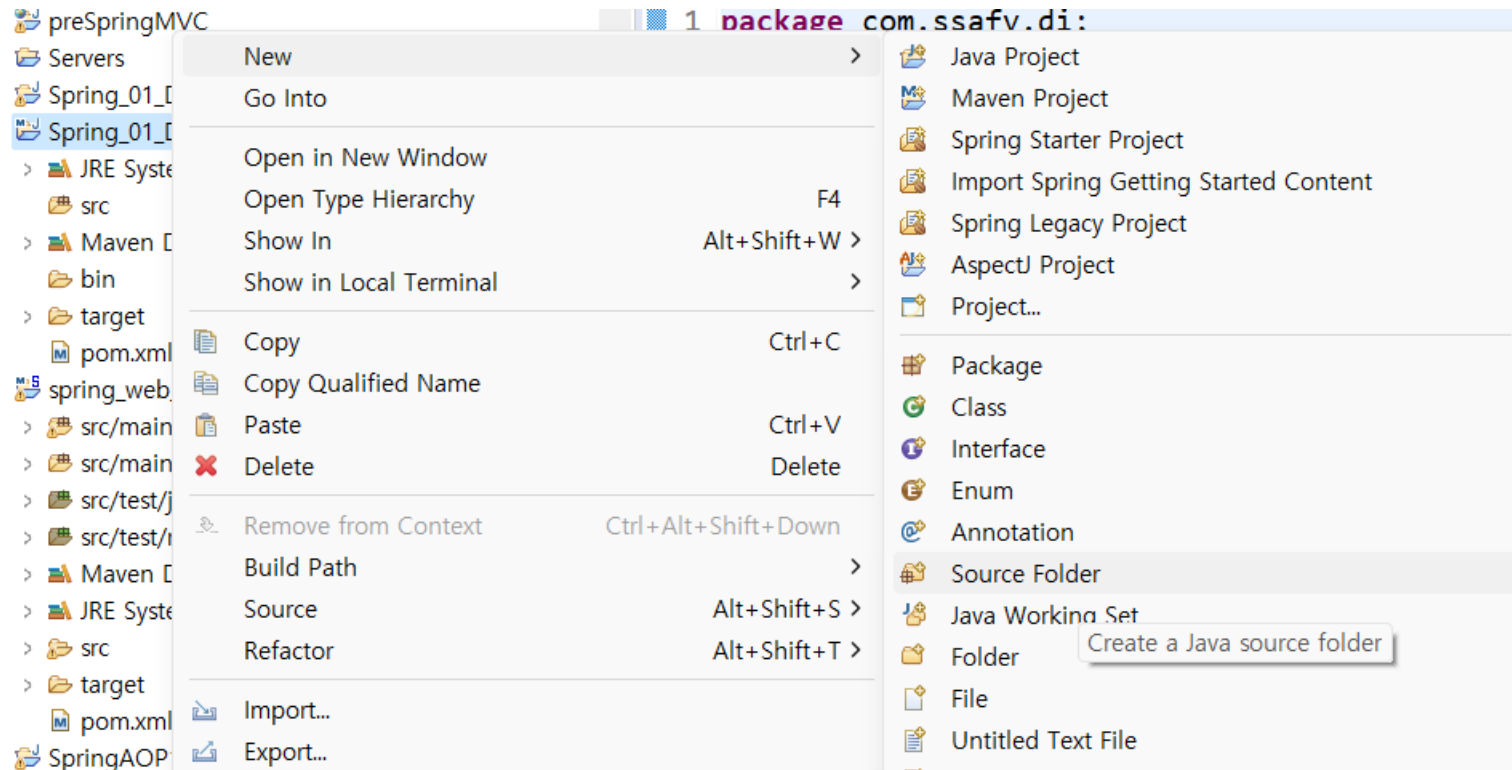
#### 📦 Maven Dependencies

- > 📦 spring-context-5.3.18.jar - C:\WU\spring-context-5.3.18.jar
- > 📦 spring-aop-5.3.18.jar - C:\WU\spring-aop-5.3.18.jar
- > 📦 spring-beans-5.3.18.jar - C:\WU\spring-beans-5.3.18.jar
- > 📦 spring-core-5.3.18.jar - C:\WU\spring-core-5.3.18.jar
- > 📦 spring-jcl-5.3.18.jar - C:\WU\spring-jcl-5.3.18.jar
- > 📦 spring-expression-5.3.18.jar - C:\WU\spring-expression-5.3.18.jar

```
<properties>
  <org.springframework-version>5.3.18</org.springframework-version>
</properties>
<build>
  <sourceDirectory>src</sourceDirectory>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.0</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${org.springframework-version}</version>
    </dependency>
  </dependencies>
```

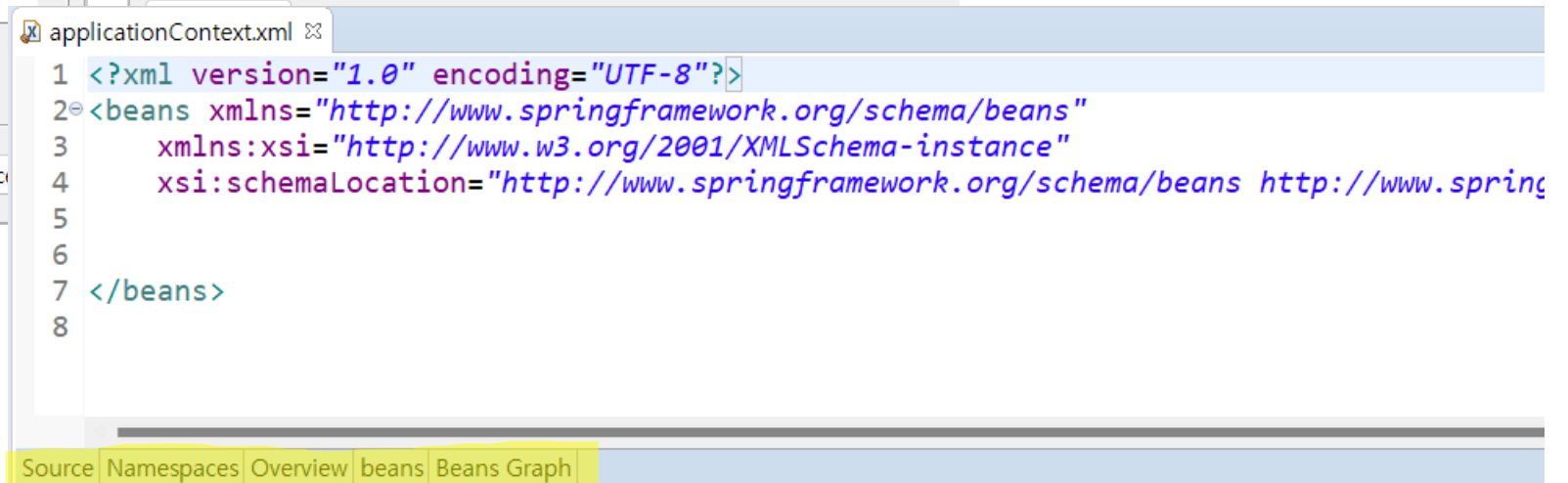
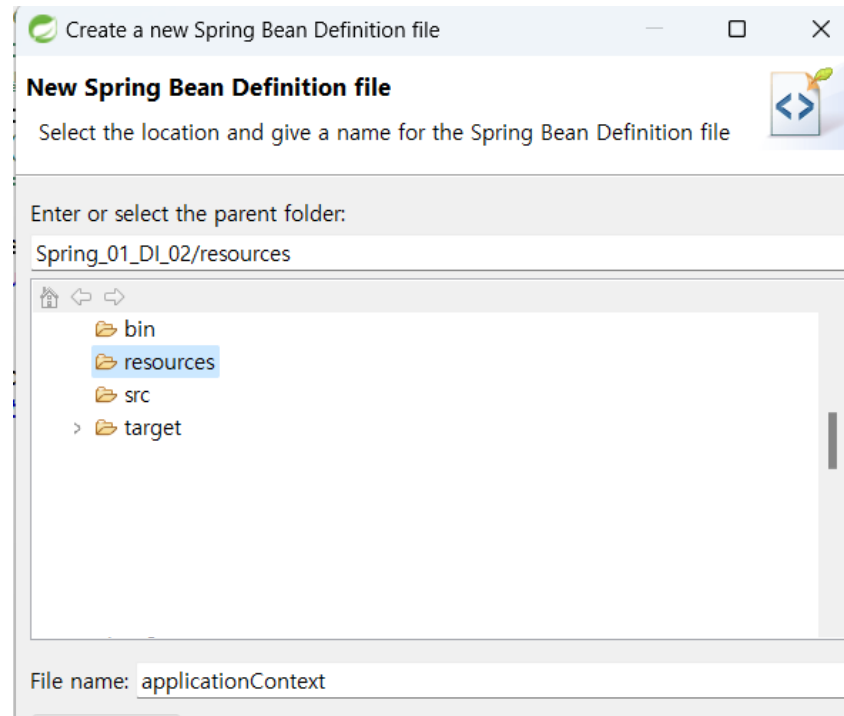
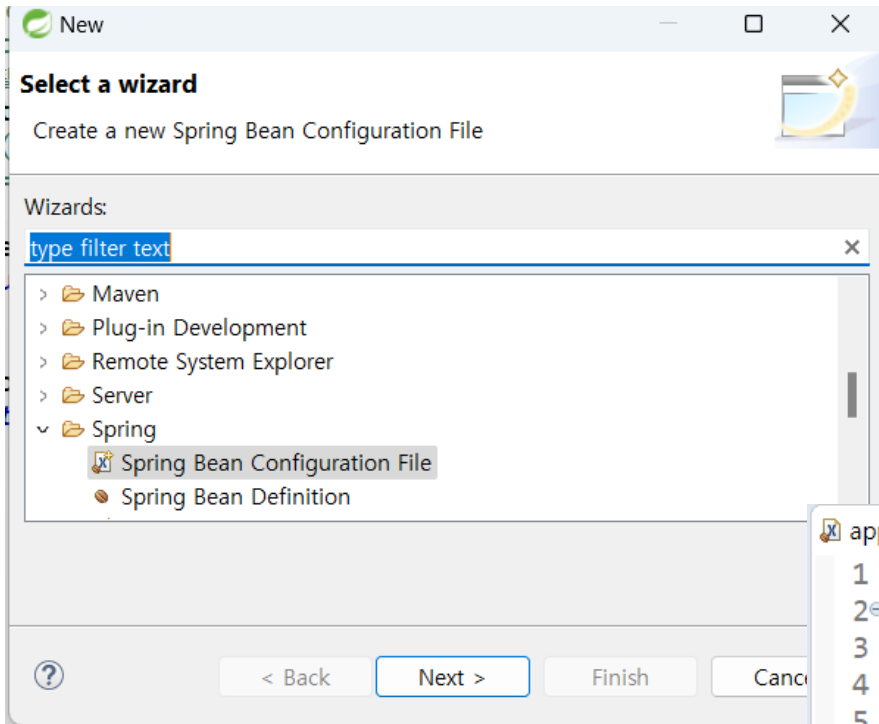
## ✓ Spring Container 빌드

### ▪ Source Folder 생성 (resources)



## ✓ Spring Container 빌드

### ■ 스프링 설정파일 생성



## ✓ Spring Container 빌드

- 빈(Beans) 등록 (폴패키지 명 작성)

```
<?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/schema/beans http://www.springframework.org/schema/be
    <bean class="com.ssafy.di.Desktop" id="desktop"></bean>
    <bean class="com.ssafy.di.Programmer" id="programmer"></bean>
  </beans>
```

## ✓ Spring Container 빌드

- 스프링 컨테이너를 이용하여 객체 가져오기

```
public class Test {  
    public static void main(String[] args) {  
  
        //스프링 컨테이너 객체 빌드하기  
        ApplicationContext context = new GenericXmlApplicationContext("applicationContext.xml");  
        //컨테이너로부터 내가 사용할 객체를 받아온다.  
        Programmer p = (Programmer) context.getBean("programmer");  
        Desktop desktop = context.getBean("desktop", Desktop.class);  
        p.setComputer(desktop);  
        p.coding();  
    }  
}
```

Console ✕

<terminated> Test (3) [Java Application] C:\Program Files\W.

데스크톱으로 개발을 합니다.

- ✓ 객체를 `getBean()` 해서 여러 개 가져와 보자!

```
Desktop desktop2 = context.getBean("desktop", Desktop.class);  
Desktop desktop3 = context.getBean("desktop", Desktop.class);  
  
System.out.println(desktop2);  
System.out.println(desktop3);
```

Console

```
<terminated> Test (3) [Java Application] C:\Program Files\Zulu  
com.ssafy.di.Desktop@598067a5  
com.ssafy.di.Desktop@598067a5
```



### ✓ Bean Scope

- Bean 정의를 작성하는 것은 Bean 객체를 생성하는 것과는 다르다.
- Bean 범위(Scope)를 정의해서 객체의 범위를 제어할 수 있다.
- Scopes

Scope	설명
singleton	<b>기본값</b> , Spring IoC 컨테이너에 대한 단일 객체 인스턴스
prototype	빈을 요청할 때마다 새로운 인스턴스 생성
request	HTTP Request 주기로 bean 인스턴스 생성
session	HTTP Session 주기로 bean 인스턴스 생성

```
<bean class="com.ssafy.di.Programmer" id="programmer" scope=""></bean>
```

```
beans>
```

- prototype
- request
- session
- singleton

# Spring DI



### ✓ 의존성 주입 (생성자)

- constructor-arg 를 이용하여 의존성 주입
- <ref>, <value> 와 같이 하위 태그를 이용하여 설정 or 속성을 이용하여 설정

```
public class Programmer {  
    private Computer computer;  
  
    //기본생성자  
    public Programmer() {  
    }  
  
    //생성자를 이용한 의존성 주입  
    public Programmer(Computer computer) {  
        this.computer = computer;  
    }  
}
```

```
<bean class="com.ssafy.di.Desktop" id="desktop"></bean>  
  
<bean class="com.ssafy.di.Programmer" id="programmer">  
    <constructor-arg ref="desktop"></constructor-arg>  
</bean>
```

### ✓ 의존성 주입 (설정자)

- setter 를 이용하여 의존성 주입
- <ref>, <value> 와 같이 하위 태그를 이용하여 설정 or 속성을 이용하여 설정

```
public class Programmer {  
    private Computer computer;  
  
    //기본생성자  
    public Programmer() {  
    }  
  
    //setter를 이용한 의존성 주입  
    public void setComputer(Computer computer) {  
        this.computer = computer;  
    }  
}
```

```
<bean class="com.ssafy.di.Laptop" id="laptop"></bean>  
  
<bean class="com.ssafy.di.Programmer" id="programmer">  
    <property name="computer" ref="laptop"></property>  
</bean>
```

### ✓ 그 외 의존관계 설정

- Inner bean

ref 속성을 이용하여 외부 bean을 참조하는 대신 간단히 사용할 수 있다.

```
<bean id="outer" class="...">
  <!-- ref 속성을 이용하는 대신에 간단히 내부 bean 선언도 가능 -->
  <property name="target">
    <bean class="com.example.Person"> <!-- this is the inner bean -->
      <property name="name" value="Fiona Apple"/>
      <property name="age" value="25"/>
    </bean>
  </property>
</bean>
```

인자가 객체가 아닌 문자열과 기초자료 형일 때 value 속성으로 지정할 수 있다.

## ✓ 그 외 의존관계 설정

- Collections

→ <list/>, <set/>, <map/>, <props/> 요소를 이용하여 List, Set, Map, Properties의 속성을 설정

- Properties

```
<!-- results in a setAdminEmails(java.util.Properties) call -->
<property name="adminEmails">
  <props>
    <prop key="administrator">administrator@example.org</prop>
    <prop key="support">support@example.org</prop>
    <prop key="development">development@example.org</prop>
  </props>
</property>
```

## ✓ 그 외 의존관계 설정

### ▪ List

```
<!-- results in a setSomeList(java.util.List) call -->
<property name="someList">
  <list>
    <value>a list element followed by a reference</value>
    <ref bean="myDataSource" />
  </list>
</property>
```

### ▪ Map

```
<!-- results in a setSomeMap(java.util.Map) call -->
<property name="someMap">
  <map>
    <entry key="an entry" value="just some string"/>
    <entry key="a ref" value-ref="myDataSource"/>
  </map>
</property>
```

### ✓ 그 외 의존관계 설정

#### ▪ Set

```
<!-- results in a setSomeSet(java.util.Set) call -->
<property name="someSet">
  <set>
    <value>just some string</value>
    <ref bean="myDataSource" />
  </set>
</property>
```

#### ▪ Null 또는 Empty String

```
<bean class="ExampleBean">
  <property name="email" value=""/>
</bean>
```

```
<bean class="ExampleBean">
  <property name="email">
    <null/>
  </property>
</bean>
```

### ✓ 빈 (Bean) 생성 및 설정 (@Component)

#### ■ Bean 생성 - @Component

```
import org.springframework.stereotype.Component;
```

```
@Component
```

```
public class Desktop implements Computer{
```

```
    //필요한 필드...
```

객체를 생성할 대상 클래스에 작성해주는 Annotation

- 생성되는 bean의 이름은 클래스의 첫 글자를 소문자로 바꾼 것이다. 예) Desktop -> desktop  
@Component(value = "bean-name") 으로 이름 지정 가능
- 스프링은 @Component, @Service, @Controller, @Repository 의 Stereotype Annotation을 제공
- 각 @Repository, @Service, @Controller 는 목적에 맞는 구체적인 사용을 위한 @Component의 확장.  
목적에 맞게 구체화 하여 사용하면 Spring 에서 더 효율적으로 사용가능

## ✓ 빈 (Bean) 생성 및 설정 (Component Scan)

\*applicationContext.xml ⓘ  
Spring\_01\_DI\_04/resources/applicationContext.xml

**Namespaces**

**Configure Namespaces**  
Select XSD namespaces to use in the configuration file

- ☐ aop - http://www.springframework.org/schema/aop
- ☒ beans - http://www.springframework.org/schema/beans
- ☐ c - http://www.springframework.org/schema/c
- ☐ cache - http://www.springframework.org/schema/cache
- ☒ context - http://www.springframework.org/schema/context
- ☐ jee - http://www.springframework.org/schema/jee
- ☐ lang - http://www.springframework.org/schema/lang
- ☐ p - http://www.springframework.org/schema/p
- ☐ task - http://www.springframework.org/schema/task
- ☐ util - http://www.springframework.org/schema/util

```
<?xml version="1.0" encoding="UTF-8"?>
<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.springframework.org/schema/beans
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-c
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-c

    <context:component-scan base-package="com.ssafy.di"></context:component-scan>

  </beans>
```

Source Namespaces Overview Beans Context Beans Graph

Annotation 방식으로 Bean을 등록  
의존성 설정을 위해서 대상 패키지를 지정

**<context:component-scan base-package="com.ssafy.di"></context:component-scan>**



## ✓ 의존성 주입 (@Autowired)

```
@Component
public class Programmer {

    private Computer computer;

    //기본 생성자
    public Programmer() {
    }

    //생성자를 이용한 의존성 주입
    @Autowired
    public Programmer(Computer computer) {
        this.computer = computer;
    }
}
```

의존성을 주입할 대상에 @Autowired annotation 작성

- Spring 컨테이너 내에서 타입 매칭 시행 (컨테이너에 해당하는 타입의 bean이 있다면 매칭)

### ✓ @Autowired 사용 가능 위치

#### ■ 생성자

```
//생성자를 이용한 의존성 주입
@Autowired
public Programmer(Computer computer) {
    this.computer = computer;
}
```

생성자를 하나만 정의한다면  
@Autowired 생략가능

#### ■ Setter

```
//setter를 이용한 의존성 주입
@Autowired
public void setComputer(@Qualifier("desktop")Computer computer) {
    this.computer = computer;
}
```

#### ■ field

```
@Autowired
@Qualifier("laptop")
private Computer computer;
```

@Qualifier 를 이용하여 같은 타입이 여러 개일  
경우, bean을 지정하여 식별가능

# 다음 방송에서 만나요!

삼성 청년 SW 아카데미