

1. STS(Spring Tools Suite) 소개

1) Spring 개발업체인 SpringSource가 직접 만들어 제공하는 Eclipse의 확장판으로 최신 Eclipse를 기반으로 주요한 Spring 지원 플러그인과 관련된 도구를 모아서 Spring 개발에 최적화되도록 만들어진 IDE이다.

2) STS가 제공하는 기능

- Bean 클래스 이름 자동완성
  - 현재 프로젝트의 모든 Source와 Library, JDK안의 모든 클래스 중에서 첫 글자가 SDD로 시작하는 클래스를 자동으로 보여줌
- 설정 파일 생성 위저드
  - Bean 설정 파일 생성 위저드 중 사용할 Namespace와 Schema 버전을 선택하는 화면 제공
- Bean 의존 관계 그래프
  - Spring IDE는 XML 설정 파일을 읽어서 자동으로 그래프 그려줌
  - 각 Bean이 어떻게 참조되고, 어떤 Property를 갖는지 알 수 있음.
- AOP 적용 대상 표시
  - Spring IDE의 XML 설정파일 편집기를 이용하면 AOP의 적용 대상을 손쉽게 확인할 수 있다.

3) Downloads

- Visit to <https://spring.io/tools>
- Click [See all versions]
- In [Windows], Click Menu button > WIN, 64BIT, Click zip
- Filename : spring-tool-suite-3.9.1.RELEASE-e4.7.1a-win32-x86)64.zip

2. STS 시작하기

1) Download 받은 spring-tool-suite-3.9.1.RELEASE-e4.7.1a-win32-x86)64.zip의 압축 푼다.

2) 압축을 풀면 sts-bundle 폴더가 생성되는데, 여기서 sts-3.9.1.RELEASE 폴더를 잘라내기해서 C:\Program Files에 붙인다.

3) 바탕화면에 바로가기 아이콘 생성을 통해 링크를 C:\Program Files\sts-3.9.1.RELEASE\STS.exe으로 연결한다.

4) 바로가기 아이콘을 실행한다.

5) Workspace를 C:\SpringHome으로 잡고 [Use this as the default and do not ask again] 체크한 뒤, [OK] 버튼을 누른다.

3. 간단한 Maven Project

1) Java Project 생성

- Project Name : HelloWorld
- Class Name : Hello

```

public class Hello {
    public static void main(String [] args){
        System.out.println("Hello, World");
    }
}

```

- 실행 확인

2) Maven Project로 전환

- HelloWorld Project > right-click > Configure > Convert to Maven Project
  - Project : /HelloWorld
  - Group Id : HelloWorld
  - Artifact Id : HelloWorld
  - version : 0.0.1-SNAPSHOT
  - Packaging : jar
  - Finish

3) Spring Project로 전환

- HelloWorld Project > right-click > Spring Tools > Add Spring Project Nature
- <http://mvnrepository.com>에서 spring context module 검색
- <https://spring.io> > [PROJECTS] 메뉴 > [SPRING FRAMEWORK] > current 버전 확인
- 이 문서를 작성하는 현재 버전은 4.3.12.RELEASE이다.
- 현재 버전의 Dependency를 복사한 다음 pom.xml에 붙여 넣는다.
 

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>HelloWorld</groupId>
    <artifactId>HelloWorld</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>

```

```

61      <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
62      <dependency>
63          <groupId>org.springframework</groupId>
64          <artifactId>spring-context</artifactId>
65          <version>4.3.12.RELEASE</version>
66      </dependency>
67
68  </dependencies>
69  <build>
70      <sourceDirectory>src</sourceDirectory>
71      <plugins>
72          <plugin>
73              <artifactId>maven-compiler-plugin</artifactId>
74              <version>3.7.0</version>
75              <configuration>
76                  <source>1.8</source>
77                  <target>1.8</target>
78              </configuration>
79          </plugin>
80      </plugins>
81  </build>
82 </project>
83 -pom.xml > right-click > Run As > Maven Install > BUILD SUCCESS <--at Console Window
84 -Dependencies tab에서 spring-context : 5.0.1.RELEASE 설치된 것을 확인 함.
85
86

```

#### 4. 환경설정

##### 1) Tomcat 새로 설치

```

89 -msi 버전이 아닌 zip 버전으로
90 -tomcat-users.xml
91     <user name="admin" password="admin" roles="admin-gui,manager-gui" />
92 -tomcat home directory 변경
93     %CATALINA_HOME%/webapps/homecontext.xml
94     <Context path="" docBase="C:/SpringHome" debug="0" reloadable="true"
95     crossContext="true" privileged="true" />

```

##### 2) STS 설치

```

97 -http://spring.io/tools/sts
98
99

```

#### 5. BeforeSpring Java Project

##### 1)com.javasoft.Calculator.java

```

102 package com.javasoft;
103
104 public class Calculator {
105     public void addAction(int a, int b){
106         System.out.println("Called addAction()");
107         System.out.printf("%d + %d = %d\n", a, b, (a + b));
108     }
109     public void subAction(int a, int b){
110         System.out.println("Called subAction()");
111         System.out.printf("%d - %d = %d\n", a, b, (a - b));
112     }
113     public void multiAction(int a, int b){
114         System.out.println("Called multiAction()");
115         System.out.printf("%d x %d = %d\n", a, b, (a * b));
116     }
117     public void divAction(int a, int b){
118         System.out.println("Called divAction()");
119         System.out.printf("%d / %d = %d\n", a, b, (a / b));
120     }
121 }
122

```

##### 2)com.javasoft.MyCalculator.java

```

124 package com.javasoft;
125
126 public class MyCalculator {

```

```

127     private Calculator calculator;
128     private int firstNum;
129     private int secondNum;
130     public void setFirstNum(int firstNum) {
131         this.firstNum = firstNum;
132     }
133     public void setSecondNum(int secondNum) {
134         this.secondNum = secondNum;
135     }
136     public void setCalculator(Calculator calculator){
137         this.calculator = calculator;
138     }
139     public void add(){
140         this.calculator.addAction(firstNum, secondNum);
141     }
142     public void sub(){
143         this.calculator.subAction(firstNum, secondNum);
144     }
145     public void multi(){
146         this.calculator.multiAction(firstNum, secondNum);
147     }
148     public void div(){
149         this.calculator.divAction(firstNum, secondNum);
150     }
151 }

```

### 3)com.javasoft.MainClass

```

154 package com.javasoft;
155
156 public class MainClass {
157     public static void main(String[] args) {
158         MyCalculator myCalculator = new MyCalculator();
159         myCalculator.setCalculator(new Calculator());
160
161         myCalculator.setFirstNum(10);
162         myCalculator.setSecondNum(2);
163
164         myCalculator.add();
165         myCalculator.sub();
166         myCalculator.multi();
167         myCalculator.div();
168     }
169 }

```

## 6. New > Spring Legacy Project > Simple Projects > Simple Spring Maven

-Project Name : StartSpring

1)Create package to src/main/java/com.javasoft

2)Copy MyCalculator.java, Calculator.java from BeforeSpring project to StartSpring's package

3)Create com.javasoft.MainClass.java

```

177 package info.javaexpert;
178
179 public class MainClass {
180     public static void main(String[] args) {
181
182     }
183 }

```

4)src/main/resources/Mouse Right-click >New > Spring Bean Configuration File >

-Name : applicationContext.xml > Finish

<?xml version="1.0" encoding="UTF-8"?>

```

188 <beans xmlns="http://www.springframework.org/schema/beans"
189     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
190     xsi:schemaLocation="http://www.springframework.org/schema/beans
191     http://www.springframework.org/schema/beans/spring-beans.xsd">

```

```

192     <bean id="calculator" class="com.javasoft.Calculator" />

```

```

193     <bean id="myCalculator" class="com.javasoft.MyCalculator">
194         <property name="calculator">
195             <ref bean="calculator" />
196         </property>
197         <property name="firstNum" value="10" />
198         <property name="secondNum" value="2" />
199     </bean>
200 </beans>
201

```

## 202 5)MainClass.java

```

203 package com.javasoft;
204
205 import org.springframework.context.support.AbstractApplicationContext;
206 import org.springframework.context.support.GenericXmlApplicationContext;
207
208 public class MainClass {
209     public static void main(String[] args) {
210         String configFile = "classpath:applicationContext.xml";
211         AbstractApplicationContext ctx = new GenericXmlApplicationContext(configFile);
212         MyCalculator myCalculator = ctx.getBean("myCalculator", MyCalculator.class);
213
214         myCalculator.add();
215         myCalculator.sub();
216         myCalculator.multi();
217         myCalculator.div();
218
219         ctx.close();
220     }
221 }
222

```

223 #Error Failed to collect dependencies for org.apache.maven.plugins:maven-resources-plugin:jar:2.0.6

- 224 1. Close Eclipse.
- 225 2. Navigate to user home directory. (For example: "C:\Users\YourUserName.m2")
- 226 3. Delete the "repository" folder.
- 227 4. Re-open Eclipse.
- 228 5. Click on the Maven project that has an issue and go to "Project" --> "Clean".
- 229 6. Right-click on the project and go to "Maven" --> "Update Project...".
- 230 7. Close Eclipse.
- 231 8. Open Eclipse.
- 232 9. Click on the project folder in the "Project Explorer" window (usually on the left).
- 233 10. Hit the "F5" key a few times to Refresh your project.
- 234 11. Done!

## 235 7. IoC(Inversion of Control)

### 236 1)개념

- 237 -객체의 생성, 생명주기의 관리까지 모든 객체에 대한 제어권이 바뀌었다는 것을 의미
- 238 -컴포넌트 의존관계 결정(Component Dependency Resolution), 설정(Configuration) 및 Lifecycle를
- 239 해결하기 위한 디자인 패턴
- 240 -의존이란 변경에 의해 영향을 받는 관계라는 의미이다.
- 241 -한 클래스의 내부 코드가 변경되었을 때 이와 관련된 다른 클래스도 함께 변경해야 한다면 이를 변경에
- 242 따른 영향이 전파되는 관계로서 '의존'한다고 표현한다.
- 243 -의존하는 대상이 있으면, 그 대상을 구하는 방법이 필요하다.
- 244 -가장 쉬운 방법은 의존 대상 객체를 직접 생성하는 것이다.
- 245 -그래서 의존받는 클래스를 생성하면 그 클래스가 의존하고 있는 클래스도 동시에 생성이 된다.
- 246 -이렇게 클래스 내부에서 직접 의존 객체를 생성하는 것은 쉽지만, 유지 보수 관점에서 보면 문제점이
- 247 유발될 수 있다.

### 248 2)IoC 컨테이너

- 249 -스프링 프레임워크도 객체에 대한 생성 및 생명주기를 관리할 수 있는 기능을 제공하고 있음.
- 250 -IoC 컨테이너 기능을 제공한다.
- 251 -IoC 컨테이너는 객체의 생성을 책임지고, 의존성을 관리한다.
- 252 -POJO의 생성, 초기화, 서비스, 소멸에 대한 권한을 가진다.
- 253 -개발자들이 직접 POJO를 생성할 수 있지만 컨테이너에게 맡긴다.

### 254 3)IoC의 분류

- DI : Dependency Injection
  - Spring, PiconContainer
  - Setter Injection, Constructor Injection, Method Injection
  - 각 클래스간의 의존관계를 빈 설정(Bean Definition) 정보를 바탕으로 컨테이너가 자동으로 연결해주는 것
- DL : Dependency Lookup
  - EJB, Spring
  - 의존성 검색 : 저장소에 저장되어 있는 Bean에 접근하기 위해 컨테이너가 제공하는 API를 이용하여 Bean을 Lookup 하는 것
- DL 사용시 컨테이너 종속성이 증가하여, 주로 DI를 사용함.

## 8. DI

### 1)DI의 개념

- 각 클래스간의 의존관계를 빈 설정(Bean Definition) 정보를 바탕으로 컨테이너가 자동으로 연결해 주는 것을 말함.
- 개발자들은 단지 빈 설정파일에서 의존관계가 필요하다는 정보를 추가하면 된다.
- 객체 레퍼런스를 컨테이너로부터 주입 받아서, 실행시에 동적으로 의존관계가 생성된다.
- 컨테이너가 흐름의 주체가 되어 어플리케이션 코드에 의존관계를 주입해주는 것이다.
- 장점
  - 코드가 단순해진다.
  - 컴포넌트 간의 결합도가 제거된다.

### 2)유형

- Setter Injection
  - Setter 메소드를 이용한 의존성 삽입
  - 의존성을 입력 받는 setter 메소드를 만들고, 이를 통해 의존성을 주입한다.
- Constructor Injection
  - 생성자를 이용한 의존성 삽입
  - 필요한 의존성을 포함하는 클래스의 생성자를 만들고 이를 통해 의존성을 주입한다.
- Method Injection
  - 일반 메소드를 이용한 의존성 삽입
  - 의존성을 입력받는 일반 메소드를 만들고 이를 통해 의존성을 주입한다.

### 3)DI를 이용한 클래스 호출방식

```
Hello<Class> --> Printer<Interface>
                |           |
                |           |
                String Printer Console Printer
```

beans.xml

- Hello class가 직접 String Printer나 Console Printer를 찾아서 사용하는 것이 아니라 설정파일(Spring Bean Configuration File)에 설정하면 컨테이너가 연결해준다.

#### -Setter Injection

<beans.xml>

```
<bean id="hello" class="bean.Hello">  <!--bean은 Srping이 관리해주는 객체라는 뜻
    <property name="name" value="Spring" />
    <property name="printer" ref="printer" />
</bean>
<bean id="printer" class="bean.StringPrinter" />
<bean id="consolePrinter" class="bean.ConsolePrinter" />
```

<Hello.java>

```
package bean;

import java.util.List;

public class Hello{
    String name;
    Printer printer;

    public Hello(){
    public void setName(String name){
        this.name = name;
    }
    public void setPrinter(Printer printer){
```

```

320         this.printer = printer;
321     }
322 }
323

```

#### 324 -Constructor Injection

```

325 <beans.xml>
326     <bean id="hello" class="bean.Hello">    <!--bean은 Srping이 관리해주는 객체라는 뜻
327         <constructor-arg index="0" value="Spring" />
328         <constructor-arg index="1" ref="printer" />
329     </bean>
330     <bean id="printer" class="bean.StringPrinter" />
331     <bean id="consolePrinter" class="bean.ConsolePrinter" />
332
333 <Hello.java>
334     package bean;
335
336     import java.util.List;
337
338     public class Hello{
339         String name;
340         Printer printer;
341
342         public Hello(){
343         public Hello(String name, Printer printer){
344             this.name = name;
345             this.printer = printer;
346         }
347     }
348
349

```

#### 350 9. Spring DI Container의 개념

- 351 1)Spring DI Container가 관리하는 객체를 빈(bean)이라고 하고, 이 빈들을 관리한다는 의미로 컨테이너를 빈 팩토리(BeanFactory)라고 부른다.
- 352 2)객체의 생성과 객체 사이의 런타임(run-time) 관계를 DI 관점에서 볼 때는 컨테이너를 BeanFactory라고 한다.
- 353 3)Bean Factory에 여러 가지 컨테이너 기능을 추가하여 어플리케이션 컨텍스트(ApplicationContext)라고 부른다.

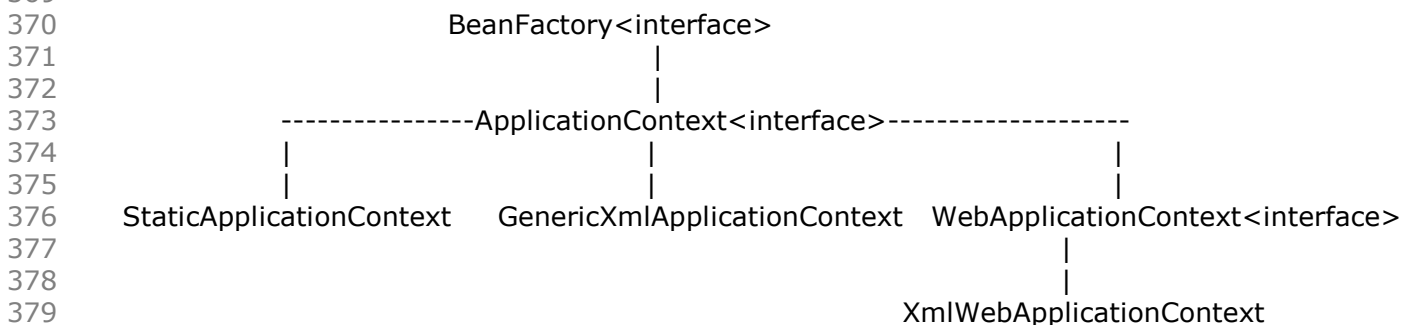
```

354 BeanFactory<interface>
355 |
356 |
357 |
358 ApplicationContext<interface>
359

```

#### 360 4)BeanFactory와 ApplicationContext

- 361 -BeanFactory
  - 362 --Bean을 등록, 생성, 조회, 반환 관리함
  - 363 --보통은 BeanFactory를 바로 사용하지 않고, 이를 확장한 ApplicationContext를 사용함
  - 364 --getBean() 메소드가 정의되어 있음.
- 365 -ApplicationContext
  - 366 --Bean을 등록, 생성, 조회, 반환 관리하는 기능은 BeanFactory와 같음.
  - 367 --Spring의 각종 부가 서비스를 추가로 제공함.
  - 368 --Spring이 제공하는 ApplicationContext 구현 클래스가 여러가지 종류가 있음.



#### 382 10. Spring DI 용어

- 383 1)Bean

- Spring이 IoC 방식으로 관리하는 객체라는 뜻
- Spring이 직접 생성과 제어를 담당하는 객체를 **Bean**이라고 부른다.
- 2)BeanFactory
  - Spring의 IoC를 담당하는 핵심 **Container**
  - Bean을 등록, 생성, 조회, 반환하는 기능을 담당.
  - 이 BeanFactory를 바로 사용하지 않고 이를 확장한 **ApplicationContext**를 주로 이용
- 3)ApplicationContext
  - BeanFactory를 확장한 Ioc Container
  - Bean을 등록하고 관리하는 기능은 BeanFactory와 동일하지만 Spring이 제공하는 각종 부가 서비스를 추가로 제공
  - Spring에서는 ApplicationContext를 BeanFactory보다 더 많이 사용
- 4)Configuration metadata
  - ApplicationContext 또는 BeanFactory가 IoC를 적용하기 위해 사용하는 메타정보
  - 설정 메타정보는 IoC Container에 의해 관리되는 Bean 객체를 생성하고 구성할 때 사용됨.

## 11. 간단한 DI 프로젝트

- 1)In Package Explorer > right-click > New > Java Project  
Project name : DIDemo

- 2)src > right-click > New > Package  
Package name : info.javaexpert

- 3)POJO class 작성

- info.javaexpert > right-click > New > Class  
<Hello.java>

```
package info.javaexpert;

public class Hello{
    private String name;
    private 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());
    }
}
```

- info.javaexpert > right-click > New > Interface  
interface name : Printer

```
<Printer.java>
package info.javaexpert;

public interface Printer{
    public void print(String message);
}
```

- info.javaexpert > right-click > New > Class  
Class Name : StringPrinter

```
<StringPrinter.java>
package info.javaexpert;
```

```

450 public class StringPrinter implements Printer{
451     private StringBuffer buffer = new StringBuffer();
452
453     public void print(String message){
454         this.buffer.append(message);
455     }
456
457     public String toString(){
458         return this.buffer.toString();
459     }
460 }

```

462 -info.javaexpert > right-click > New > Class  
463 Class Name : ConsolePrinter

```

464 <ConsolePrinter.java>
465 package info.javaexpert;
466
467 public class ConsolePrinter implements Printer{
468     public void print(String message){
469         System.out.println(message);
470     }
471 }
472 }

```

#### 4)Java Project를 Spring Project로 변환

475 -DIDemo Project > right-click > Configuration > Convert to Maven Project

```

476 --Project : /DIDemo
477 --Group Id : DIDemo
478 --Artifact Id : DIDemo
479 --version : 0.0.1-SNAPSHOT
480 --Packaging : jar
481 --Finish
482 --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.

```

484 -DIDemo Project > right-click > Spring Tools > Add Spring Project Nature  
485 --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.

487 -pom.xml 파일에 Spring Context Dependency 추가하기

```

488 <version>0.0.1-SNAPSHOT</version>
489 <!--여기부터 작성 -->
490 <dependencies>
491     <dependency>
492         <groupId>org.springframework</groupId>
493         <artifactId>spring-context</artifactId>
494         <version>4.3.9.RELEASE</version>
495     </dependency>
496 </dependencies>

```

498 -pom.xml > right-click > Run As > Maven install

#### 5)src/config folder 생성

501 -/src > right-click > New > Folder  
502 Folder name : config <--설정 Meta 정보 XML 작성

#### 6)Bean Configuration XML 작성

505 -/src/config > right-click > New > Other > Spring > Spring Bean Configuration File  
506 File name : beans.xml > Next  
507 Check [beans - <http://www.springframework.org/schema/beans>]  
508 Check [<http://www.springframework.org/schema/beans/spring-beans-4.3.xsd>]  
509 Finish

```

511 <?xml version="1.0" encoding="UTF-8"?>
512 <beans xmlns="http://www.springframework.org/schema/beans"
513     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
514     xsi:schemaLocation="http://www.springframework.org/schema/beans
515         http://www.springframework.org/schema/beans/spring-beans.xsd">

```



```

516     <bean id="hello" class="info.javaexpert.Hello">
517         <property name="name" value="Spring" />
518         <property name="printer" ref="printer" />
519     </bean>
520     <bean id="printer" class="info.javaexpert.StringPrinter" />
521     <bean id="consolePrinter" class="info.javaexpert.ConsolePrinter" />
522
523 </beans>
524

```

## 7)Beans Graph 사용하기

- Windows menu > Show View > Other > Spring > Spring Explorer
- In Spring Explorer
  - DIDemo > Beans > beans.xml > right-click > Open Beans Graphs

## 8)DI Test 클래스 작성

```

531 -/src/info.javaexpert > right-click > New > Package
532     Package Name : test
533 -/src/info.javaexpert/test/HelloBeanTest.java
534
535     package info.javaexpert.test;
536
537     import org.springframework.context.ApplicationContext;
538     import org.springframework.context.support.GenericXmlApplicationContext;
539
540     import info.javaexpert.Hello;
541     import info.javaexpert.Printer;
542
543     public class HelloBeanTest {
544         public static void main(String [] args){
545             //1. IoC Container 생성
546             ApplicationContext context =
547                 new GenericXmlApplicationContext("config/beans.xml");
548
549             //2. Hello Beans 가져오기
550             Hello hello = (Hello)context.getBean("hello");
551             System.out.println(hello.sayHello());
552             hello.print();
553
554             //3. StringPrinter 가져오기
555             Printer printer = (Printer)context.getBean("printer");
556             System.out.println(printer.toString());
557
558             Hello hello2 = context.getBean("hello", Hello.class);
559             hello2.print();
560
561             System.out.println(hello == hello2); //Singleton Pattern
562         }
563     }
564
565     -----
566     Hello Spring
567     Hello Spring
568     true
569
570

```

## 12. junit의 개요와 특징

### 1)junit의 특징

- TDD의 창시자인 **Kent Beck**과 디자인 패턴 책의 저자인 **Erich Gamma**가 작성
- 단정(**Assert**) 메소드로 테스트 케이스의 수행 결과를 판별 --> **assertEquals**(예상 값, 실제 값)
- junit4**부터는 테스트를 지원하는 어노테이션 제공, **@Test**, **@Before**, **@After**
- 각 **@Test** 메소드가 호출할 때마다 새로운 인스턴스를 생성하여 독립적인 테스트가 이루어지도록 한다.

### 2)junit

- junit Library 설치
  - <http://mvnrepository.com>에 접근
  - junit으로 검색
  - junit 4.12 버전을 pom.xml에 추가

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

--pom.xml > right-click > Run As > Maven Install

-JUnit에서 테스트를 지원하는 어노테이션

--@Test

---이것이 선언된 메소드는 테스트를 수행하는 메소드가 된다.

---JUnit은 각각의 테스트가 서로 영향을 주지 않고 독립적으로 실행됨을 원칙으로 하므로 @Test 마다 객체를 생성한다.

--@Ignore

---이것이 선언된 메소드는 테스트를 실행하지 않게 한다.

--@Before

---이것이 선언된 메소드는 @Test가 실행되기 전에 반드시 실행된다.

---@Test 메소드에서 공통으로 사용하는 코드를 @Before 메소드에 선언하여 사용하면 된다.

--@After

---이것이 선언된 메소드는 @Test 메소드가 실행된 후 실행된다.

--@BeforeClass

---이 어노테이션은 @Test 메소드보다 먼저 한번만 수행되어야 할 경우에 사용하면 된다.

--@AfterClass

---이 어노테이션은 @Test 메소드보다 나중에 한번만 수행되어야 할 경우에 사용하면 된다.

-테스트 결과를 확인하는 단정(Assert) 메소드 종류

--org.junit.Assert

+assertArrayEquals(expected, actual)

+assertEquals(expected, actual)

+assertNotNull(object)

+assertSame(expected, actual)

+assertTrue(object)

-assertEquals(a, b)

--객체 a와 b가 일치함을 확인

-assertArrayEquals(a, b)

--배열 a, b가 일치함을 확인

-assertSame(a, b)

--객체 a, b가 같은 객체임을 확인

--assertEquals() 메소드는 값이 같은지를 확인하는 것이고, assertEquals() 메소드는 두 객체의 레퍼런스가 같은지를 확인한다.(==연산자)

-assertTrue(a)

--조건 a가 참인가를 확인

-assertNotNull(a)

--객체 a가 null이 아님을 확인한다.

-이외에도 다양한 assert 메소드가 존재함

<http://junit.sourceforge.net/javadoc/org/junit/Assert.html>

3)JUnit을 사용한 DI 테스트 클래스 작성하기

-JUnit을 사용한 DI 테스트 클래스(HelloBeanJUnitTest.java) 작성

--/src/info.javaexpert.test/HelloBeanTest.java 복사

--/src/info.javaexpert.test/ 붙여넣고 이름변경 -> HelloBeanJUnitTest.java

```
package info.javaexpert.test;
```

```
import org.junit.Before;
```

```
import org.junit.Test;
```

```
import org.springframework.context.ApplicationContext;
```

```
import org.springframework.context.support.GenericXmlApplicationContext;
```

```

648 import info.javaexpert.Hello;
649 import info.javaexpert.Printer;
650
651 import static org.junit.Assert.assertEquals;
652 import static org.junit.Assert.assertSame;
653
654 public class HelloBeanJUnitTest {
655     ApplicationContext context;
656
657     @Before
658     public void init(){
659         //항상 먼저 ApplicationContext를 생성해야 하기 때문에
660         //1. IoC Container 생성
661         context = new GenericXmlApplicationContext("config/beans.xml");
662     }
663
664     @Test
665     public void test1(){
666         //2. Hello Beans 가져오기
667         Hello hello = (Hello)context.getBean("hello");
668         assertEquals("Hello Spring", hello.sayHello());
669         hello.print();
670
671         //3. SpringPrinter 가져오기
672         Printer printer = (Printer)context.getBean("printer");
673         assertEquals("Hello Spring", printer.toString());
674     }
675
676     @Test
677     public void test2(){
678         Hello hello = (Hello)context.getBean("hello");
679
680         Hello hello2 = context.getBean("hello", Hello.class);
681         assertEquals(hello, hello2);
682     }
683 }

```

685 -right-click > Run As > Junit Test

686 -결과 -> Junit View에 초록색 bar

687 -만일, test1() 메소드를 junit에서 제외하고 싶을 때에는 @Test 옆에 @Ignore를 선언한다.

```

689 import org.junit.Ignore;
690 ...
691 @Test @Ignore
692 public void test1(){
693     ...

```

695 -right-click > Run As > Junit Test

696 --JUnit Test 목록에서 test1()는 실행되지 않는다.

## 699 13. Spring TestContext Framework

### 700 1)Spring-Test library 설치

701 -<http://mvnrepository.com>에서 'spring-test'로 검색

702 -검색 결과 목록에서 'Spring TestContext Framework' 클릭

703 -버전 목록에서 4.3.9.RELEASE 클릭

704 -dependency 복사해서 pom.xml에 붙여넣기

```

706 <dependency>
707 <groupId>org.springframework</groupId>
708 <artifactId>spring-test</artifactId>
709 <version>4.3.9.RELEASE</version>
710 <scope>test</scope>
711 </dependency>

```

713 -pom.xml > right-click > Maven Install

## 2)Spring-Test에서 테스트를 지원하는 어노테이션

-@RunWith(SpringJUnit4ClassRunner.class)

--JUnit 프레임워크의 테스트 실행방법을 확장할 때 사용하는 어노테이션

--SpringJUnit4ClassRunner라는 클래스를 지정해주면 JUnit이 테스트를 진행하는 중에 ApplicationContext를 만들고 관리하는 작업을 진행해 준다.

--이 어노테이션은 각각의 테스트 별로 객체가 생성되더라도 Singleton의 ApplicationContext를 보장한다.

-@ContextConfiguration

--Spring bean 설정 파일의 위치를 지정할 때 사용되는 어노테이션

-@Autowired

--Spring DI에서 사용되는 특별한 어노테이션

--해당 변수에 자동으로 빈(Beans)을 매핑해준다.

--Spring bean 설정 파일을 읽기 위해 굳이 GenericXmlApplicationContext를 사용할 필요가 없다.

## 3)Spring-Test를 사용할 DI 테스트 클래스-HelloBeanJUnitSpringTest.java 작성하기

~/src/info.javaexpert.test/HelloBeanJUnitTest.java 복사해서

~/src/info.javaexpert.test/HelloBeanJUnitSpringTest.java 로 붙여넣기

--ApplicationContext 생성하는 부분을 매번 수행하는 것이 아니라 이 부분을 자동으로 해주는 것은 SpringTest Framework가 하게 한다.

--따라서 init()이 필요하지 않도록 설정한다.

```
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
...
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations="classpath:config/beans.xml")
//beans.xml경로를 수정한다. 경로 앞에 classpath:를 넣는다.
public class HelloBeanJUnitSpringTest {
```

```
    @Autowired
    ApplicationContext context;
```

-아래의 init()가 필요없어짐으로 삭제한다.

```
/*
@Before
public void init(){
    //항상 먼저 ApplicationContext를 생성해야 하기 때문에
    //1. IoC Container 생성
    context = new GenericXmlApplicationContext("config/beans.xml");
}
*/
```

-right-click > Run As > Junit Test

-결과 -> Junit View에 초록색 bar

## 14. Dependency Injection(의존주입) 방법의 종류

### 1)XML 파일을 이용한 DI 설정 방법

-setter 이용하기

-생성자 이용하기

### 2)Java Annotation 이용한 DI 설정 방법

### 3)Java Annotation과 XML 을 이용한 DI 설정 방법

-XML 파일에 Java 파일을 포함시켜 사용하는 방법

-Java 파일에 XML 파일을 포함시켜 사용하는 방법

## 15. setter를 이용한 의존주입하기 -> Setter Injection

1)setter 메소드를 통해 의존 관계가 있는 Bean을 주입하려면 <property> 태그를 사용할 수 있다.

2)ref 속성은 사용하면 Bean이름을 이용해서 주입할 Bean을 찾는다.

3)value 속성은 단순 값 또는 Bean이 아닌 객체를 주입할 때 사용한다.

4)단순 값(문자열이나 숫자)의 주입

-setter 메소드를 통해 Bean의 레퍼런스가 아니라 단순 값을 주입하려고 할 때는 <property> 태그의 value속성을 사용한다.

```

778
779 -/src/info.javaexpert.Hello
780     public class Hello {
781         private String name;
782         private Printer printer;
783
784         public Hello(){}
785
786         public void setName(String name){
787             this.name = name;
788             ...
789
790 -/src/config/beans.xml
791     <bean id="hello" class="info.javaexpert.Hello">
792         <property name="name" value="Spring" />
793         <property name="printer" ref="printer" />
794     </bean>
795
796

```

#### 5)Collection 타입의 값 주입

-Spring은 List, Set, Map, Properties와 같은 Collection 타입을 XML로 작성해서 property에 주입하는 방법을 제공한다.

-List 타입 : <list>와 <value> 태그를 이용

-Set 타입 : <set>과 <value> 태그를 이용

```

803     public class Hello{
804         List<String> names;
805         public void setNames(List<String> list){
806             this.names = list;
807         }
808     }
809
810     <bean id="hello" class="info.javaexpert">
811         <property name="names">
812             <list>
813                 <value>Spring</value>
814                 <value>IoC</value>
815                 <value>DI</value>
816             </list>
817         </property>
818     </bean>
819

```

-Map 타입 : <map>과 <entry> 태그를 이용

```

822     public class Hello{
823         Map<String, Integer> ages;
824
825         public void setAges(Map<String, Integer> ages){
826             this.ages = ages;
827         }
828     }
829
830     <bean id="hello" class="info.javaexpert.Hello">
831         <property name="ages">
832             <map>
833                 <entry key="나훈아" value="30" />
834                 <entry key="이미자" value="50" />
835                 <entry key="설운도" value="60" />
836             </map>
837         </property>
838     </bean>
839
840

```

#### 16. setter를 이용한 의존주입하기 실습

1)In Package Explorer > right-click > New > Java Project

Project name : DIDemo1

2)src > right-click > New > Package  
Package name : info.javaexpert

3)POJO class 작성

-info.javaexpert > right-click > New > Class  
<Hello.java>

```
package info.javaexpert;

public class Hello{
    private String name;
    private 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());
    }
}
```

-info.javaexpert > right-click > New > Interface  
interface name : Printer

```
<Printer.java>
package info.javaexpert;

public interface Printer{
    public void print(String message);
}
```

-info.javaexpert > right-click > New > Class  
Class Name : StringPrinter

```
<StringPrinter.java>
package info.javaexpert;

public class StringPrinter implements Printer{
    private StringBuffer buffer = new StringBuffer();

    public void print(String message){
        this.buffer.append(message);
    }

    public String toString(){
        return this.buffer.toString();
    }
}
```

-info.javaexpert > right-click > New > Class  
Class Name : ConsolePrinter

```
<ConsolePrinter.java>
package info.javaexpert;

public class ConsolePrinter implements Printer{
```

```
911         public void print(String message){
912             System.out.println(message);
913         }
914     }
915 }
```

#### 916 4)Java Project를 Spring Project로 변환

917 -DIDemo Project > right-click > Configuration > Convert to Maven Project

918 --Project : /DIDemo1

919 --Group Id : DIDemo1

920 --Artifact Id : DIDemo1

921 --version : 0.0.1-SNAPSHOT

922 --Packaging : jar

923 --Finish

924 -DIDemo Project > right-click > Spring Tools > Add Spring Project Nature

925 -pom.xml 파일에 Spring Context Dependency 추가하기

```
926 <version>0.0.1-SNAPSHOT</version>
```

```
927 <!--여기부터 작성 -->
```

```
928 <dependencies>
```

```
929 <dependency>
```

```
930 <groupId>org.springframework</groupId>
```

```
931 <artifactId>spring-context</artifactId>
```

```
932 <version>4.3.9.RELEASE</version>
```

```
933 </dependency>
```

```
934 </dependencies>
```

935 -pom.xml > right-click > Run As > Maven install

#### 936 5)src/config folder 생성

937 -/src > right-click > New > Folder

938 Folder name : config

#### 939 6)Bean Configuration XML 작성

940 -/src/config > right-click > New > Other > Spring > Spring Bean Configuration File

941 File name : beans.xml > Next

942 Check [beans - <http://www.springframework.org/schema/beans>]

943 Check [<http://www.springframework.org/schema/beans/spring-beans-4.3.xsd>]

944 Finish

```
945 <?xml version="1.0" encoding="UTF-8"?>
```

```
946 <beans xmlns="http://www.springframework.org/schema/beans"
```

```
947 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
948 xsi:schemaLocation="http://www.springframework.org/schema/beans
```

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

```
950 <bean id="hello" class="info.javaexpert.Hello">
```

```
951 <property name="name" value="Spring" />
```

```
952 <property name="printer" ref="printer" />
```

```
953 </bean>
```

```
954 <bean id="printer" class="info.javaexpert.StringPrinter" />
```

```
955 <bean id="consolePrinter" class="info.javaexpert.ConsolePrinter" />
```

```
956 </beans>
```

#### 957 7)DI Test 클래스 작성

958 -/src/info.javaexpert > right-click > New > Package

959 Package Name : test

960 -/src/info.javaexpert/test/HelloBeanTest.java

```
961 package info.javaexpert.test;
```

```
962 import org.springframework.context.ApplicationContext;
```

```
963 import org.springframework.context.support.GenericXmlApplicationContext;
```

```
964 import info.javaexpert.Hello;
```

```
965 import info.javaexpert.Printer;
```

```

977
978 public class HelloBeanTest {
979     public static void main(String [] args){
980         //1. IoC Container 생성
981         ApplicationContext context =
982             new GenericXmlApplicationContext("config/beans.xml");
983
984         //2. Hello Beans 가져오기
985         Hello hello = (Hello)context.getBean("hello");
986         System.out.println(hello.sayHello());
987         hello.print();
988
989         //3. SpringPrinter 가져오기
990         Printer printer = (Printer)context.getBean("printer");
991         System.out.println(printer.toString());
992
993         Hello hello2 = context.getBean("hello", Hello.class);
994         hello2.print();
995
996         System.out.println(hello == hello2);
997     }
998 }
999

```

#### 8)Test

~/src/info.javaexpert.test/HelloBeanTest.java > right-click > Run As > Java Application

```

-----
Hello Spring
Hello Spring
true

```

#### 9)jUnit으로 테스트

-jUnit Library 설치

--jUnit 4.12 버전을 pom.xml에 추가

```

<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>

```

--pom.xml > right-click > Run As > Maven Install

-jUnit을 사용한 DI 테스트 클래스(HelloBeanJUnitTest.java) 작성

--~/src/info.javaexpert.test/HelloBeanTest.java 복사

--~/src/info.javaexpert.test/ 붙여넣고 이름변경 -> HelloBeanJUnitTest.java

```

package info.javaexpert.test;

import org.junit.Before;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

import info.javaexpert.Hello;
import info.javaexpert.Printer;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertSame;

public class HelloBeanJUnitTest {
    ApplicationContext context;

    @Before
    public void init(){
        context = new GenericXmlApplicationContext("config/beans.xml");
    }

```



```

1044
1045     @Test
1046     public void test1(){
1047         Hello hello = (Hello)context.getBean("hello");
1048         assertEquals("Hello Spring", hello.sayHello());
1049         hello.print();
1050
1051         Printer printer = (Printer)context.getBean("printer");
1052         assertEquals("Hello Spring", printer.toString());
1053     }
1054
1055     @Test
1056     public void test2(){
1057         Hello hello = (Hello)context.getBean("hello");
1058
1059         Hello hello2 = context.getBean("hello", Hello.class);
1060         assertSame(hello, hello2);
1061     }
1062 }

```

1064 -right-click > Run As > Junit Test

1065 -결과 -> Junit View에 초록색 bar

1066  
1067 10)Spring-Test를 사용할 DI 테스트 클래스-HelloBeanJUnitSpringTest.java 작성하기

1068 -Spring-Test library 설치

1069 -pom.xml 코드 추가

```

1070     <dependency>
1071     <groupId>org.springframework</groupId>
1072     <artifactId>spring-test</artifactId>
1073     <version>4.3.9.RELEASE</version>
1074     <scope>test</scope>
1075 </dependency>

```

1076  
1077 -pom.xml > right-click > Maven Install

1078  
1079 -Spring-Test를 사용할 DI 테스트 클래스-HelloBeanJUnitSpringTest.java 작성하기

1080 --/src/info.javaexpert.test/HelloBeanJUnitTest.java 복사해서

1081 --/src/info.javaexpert.test/HelloBeanJUnitSpringTest.java 로 붙여넣기

```

1082
1083     import org.junit.runner.RunWith;
1084     import org.springframework.beans.factory.annotation.Autowired;
1085     import org.springframework.test.context.ContextConfiguration;
1086     import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
1087     ...
1088     @RunWith(SpringJUnit4ClassRunner.class)
1089     @ContextConfiguration(locations="classpath:config/beans.xml")
1090     public class HelloBeanJUnitSpringTest {

```

```

1091
1092         @Autowired
1093         ApplicationContext context;

```

1094  
1095 -right-click > Run As > Junit Test

1096 -결과 -> Junit View에 초록색 bar

1097  
1098 11)Hello class 수정

```

1099     ...
1100     private List<String> names;
1101     ...
1102     public void setNames(List<String> list){
1103         this.names = list;
1104     }
1105
1106     public List<String> getNames(){
1107         return this.names;
1108     }
1109     ...
1110

```

## 12)beans.xml 수정

```
<bean id="hello2" class="info.javaexpert.Hello">
  <property name="names">
    <list>
      <value>AOP</value>
      <value>Spring</value>
      <value>DI</values>
    </list>
  </property>
</bean>
```

## 13)HelloBeanJUnitTest로 테스트하기

```
@Test <--@Ignore 붙여서 테스트하지 않고
public void test1(){
    //2. Hello Beans 가져오기
    Hello hello = (Hello)context.getBean("hello");
    assertEquals("Hello Spring", hello.sayHello());
    hello.print();

    //3. SpringPrinter 가져오기
    Printer printer = (Printer)context.getBean("printer");
    assertEquals("Hello Spring", printer.toString());
}

@Test @Ignore <-- @Ignore를 해제하여 코드 수정하기
public void test2(){
    Hello hello = (Hello)context.getBean("hello");

    Hello hello2 = context.getBean("hello", Hello.class);
    assertEquals(hello, hello2);

    //아래 코드 추가
    assertEquals(3, hello2.getNames().size());
    List<String> list = hello.getNames();
    for(String value : list){
        System.out.println(value);
    }
}
```

-right-click > Run As > Junit Test

-결과 -> Junit View에 초록색 bar

## 17. setter를 이용한 의존주입하기 실습

-New > Spring Legacy Project > Simple Projects > Simple Spring Maven

Project Name : SpringDemo

1)src/main/java > Create Package

-info.javaexpert

2)info.javaexpert.BmiCalculator.java

```
package info.javaexpert;

public class BmiCalculator {
    private double lowWeight;
    private double normal;
    private double overWeight;
    private double obesity;

    public void setLowWeight(double lowWeight) {
        this.lowWeight = lowWeight;
    }

    public void setNormal(double normal) {
        this.normal = normal;
    }
}
```

```

1178
1179     public void setOverWeight(double overWeight) {
1180         this.overWeight = overWeight;
1181     }
1182
1183     public void setObesity(double obesity) {
1184         this.obesity = obesity;
1185     }
1186     public void bmiCalcu(double weight, double height){
1187         double h = height * 0.01;
1188         double result = weight / (h * h);
1189
1190         System.out.println("BMI 지수 : " + (int)result);
1191
1192         if(result > obesity)
1193             System.out.println("비만입니다.");
1194         else if(result > overWeight)
1195             System.out.println("과체중입니다.");
1196         else if(result > normal)
1197             System.out.println("정상입니다.");
1198         else
1199             System.out.println("저체중입니다.");
1200     }
1201 }
1202
1203 3)info.javaexpert.MyInfo.java
1204     package info.javaexpert;
1205
1206     import java.util.ArrayList;
1207
1208     public class MyInfo {
1209         private String name;
1210         private double height;
1211         private double weight;
1212         private ArrayList<String> hobby;
1213         private BmiCalculator bmiCalculator;
1214
1215         public void setBmiCalculator(BmiCalculator bmiCalculator) {
1216             this.bmiCalculator = bmiCalculator;
1217         }
1218         public void setName(String name) {
1219             this.name = name;
1220         }
1221         public void setHeight(double height) {
1222             this.height = height;
1223         }
1224         public void setWeight(double weight) {
1225             this.weight = weight;
1226         }
1227         public void setHobby(ArrayList<String> hobby) {
1228             this.hobby = hobby;
1229         }
1230         public void getInfo(){
1231             System.out.println("Name : " + this.name);
1232             System.out.println("Height : " + this.height);
1233             System.out.println("Weight : " + this.weight);
1234             System.out.println("Hobby : " + this.hobby);
1235             this.bmiCalcu();
1236         }
1237         public void bmiCalcu(){
1238             this.bmiCalculator.bmiCalcu(this.weight, this.height);
1239         }
1240     }
1241
1242 4)src/main/resources/ApplicationContext.xml
1243     <bean id="bmiCalculator" class="info.javaexpert.BmiCalculator">
1244         <property name="lowWeight" value="18.5" />

```

```

1245     <property name="normal" value="23" />
1246     <property name="overWeight" value="25" />
1247     <property name="obesity">
1248         <value>30</value>
1249     </property>
1250 </bean>
1251 <bean id="myInfo" class="info.javaexpert.MyInfo">
1252     <property name="name" value="한지민" />
1253     <property name="height" value="170.5" />
1254     <property name="weight" value="67" />
1255     <property name="hobby">
1256         <list>
1257             <value>수영</value>
1258             <value>요리</value>
1259             <value>독서</value>
1260         </list>
1261     </property>
1262     <property name="bmiCalculator">
1263         <ref bean="bmiCalculator" />
1264     </property>
1265 </bean>

```

5)info.javaexpert.MainClass.java

```

1266 package info.javaexpert;
1267
1268 import org.springframework.context.AbstractApplicationContext;
1269 import org.springframework.context.support.GenericXmlApplicationContext;
1270
1271 public class MainClass {
1272     public static void main(String[] args) {
1273         String configFile = "classpath:ApplicationContext.xml";
1274
1275         //Spring Container 생성
1276         AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);
1277
1278         //Spring Container 에서 객체를 가져옴
1279         MyInfo myInfo = context.getBean("myInfo", MyInfo.class);
1280
1281         myInfo.getInfo();
1282         context.close();
1283     }
1284 }
1285
1286 }

```

## 18. 생성자 이용하여 의존 주입하기 -> Constructor Injection

- 1)Constructor를 통해 의존관계가 있는 Bean을 주입하려면 <constructor-arg> 태그를 사용할 수 있다.
- 2)Constructor 주입방식은 생성자의 파라미터를 이용하기 때문에 한번에 여러 개의 객체를 주입할 수 있다.

## 19. 생성자 이용하여 의존 주입하기 실습

- 1)In Package Explorer > right-click > New > Java Project  
Project name : DIDemo2

- 2)src > right-click > New > Package  
Package name : info.javaexpert

### 3)POJO class 작성

```

1302 -info.javaexpert > right-click > New > Class
1303 <Hello.java>
1304 package info.javaexpert;
1305
1306 public class Hello{
1307     private String name;
1308     private Printer printer;
1309
1310     public Hello(){}

```

```

1312     public void setName(String name){
1313         this.name = name;
1314     }
1315
1316     public void setPrinter(Printer printer){
1317         this.printer = printer;
1318     }
1319
1320     public String sayHello(){
1321         return "Hello " + name;
1322     }
1323
1324     public void print(){
1325         this.printer.print(sayHello());
1326     }
1327 }

```

```

1329 -info.javaexpert > right-click > New > Interface
1330     interface name : Printer

```

```

1331 <Printer.java>
1332     package info.javaexpert;
1333
1334     public interface Printer{
1335         public void print(String message);
1336     }

```

```

1339 -info.javaexpert > right-click > New > Class
1340     Class Name : StringPrinter

```

```

1341 <StringPrinter.java>
1342     package info.javaexpert;
1343
1344     public class StringPrinter implements Printer{
1345         private StringBuffer buffer = new StringBuffer();
1346
1347         public void print(String message){
1348             this.buffer.append(message);
1349         }
1350
1351         public String toString(){
1352             return this.buffer.toString();
1353         }
1354     }

```

```

1356 -info.javaexpert > right-click > New > Class
1357     Class Name : ConsolePrinter

```

```

1358 <ConsolePrinter.java>
1359     package info.javaexpert;
1360
1361     public class ConsolePrinter implements Printer{
1362         public void print(String message){
1363             System.out.println(message);
1364         }
1365     }

```

#### 1369 4)Java Project를 Spring Project로 변환

```

1370 -DIDemo Project > right-click > Configuration > Convert to Maven Project
1371     --Project : /DIDemo2
1372     --Group Id : DIDemo2
1373     --Artifact Id : DIDemo2
1374     --version : 0.0.1-SNAPSHOT
1375     --Packaging : jar
1376     --Finish
1377     --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.

```

-DIDemo Project > right-click > Spring Tools > Add Spring Project Nature  
--Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.

-pom.xml 파일에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<!--여기부터 작성 -->
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>4.3.9.RELEASE</version>
  </dependency>
</dependencies>
```

-pom.xml > right-click > Run As > Maven install

5)src/config folder 생성

-/src > right-click > New > Folder  
Folder name : config

6)Bean Configuration XML 작성

-/src/config > right-click > New > Other > Spring > Spring Bean Configuration File  
File name : beans.xml > Next  
Check [beans - <http://www.springframework.org/schema/beans>]  
Check [<http://www.springframework.org/schema/beans/spring-beans-4.3.xsd>]  
Finish

```
<?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="hello" class="info.javaexpert.Hello">
    <property name="name" value="Spring" />
    <property name="printer" ref="printer" />
  </bean>
  <bean id="printer" class="info.javaexpert.StringPrinter" />
  <bean id="consolePrinter" class="info.javaexpert.ConsolePrinter" />

</beans>
```

7)DI Test 클래스 작성

-/src/info.javaexpert > right-click > New > Package  
Package Name : test  
-/src/info.javaexpert/test/HelloBeanTest.java

```
package info.javaexpert.test;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

import info.javaexpert.Hello;
import info.javaexpert.Printer;

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

    //2. Hello Beans 가져오기
    Hello hello = (Hello)context.getBean("hello");
    System.out.println(hello.sayHello());
    hello.print();

    //3. SpringPrinter 가져오기
```

```

1445         Printer printer = (Printer)context.getBean("printer");
1446         System.out.println(printer.toString());
1447
1448         Hello hello2 = context.getBean("hello", Hello.class);
1449         hello2.print();
1450
1451         System.out.println(hello == hello2); //Singleton Pattern
1452     }
1453 }
1454

```

#### 8)Test

~/src/info.javaexpert.test/HelloBeanTest.java > right-click > Run As > Java Application

```

-----
Hello Spring
Hello Spring
true

```

#### 9)/src/info.javaexpert.Hello 생성자 추가

```

public Hello(String name, Printer printer) {
    this.name = name;
    this.printer = printer;
}

```

#### 10)/src/config/beans.xml에 추가

```

<bean id="hello2" class="info.javaexpert.Hello">
    <constructor-arg index="0" value="Spring" />
    <constructor-arg index="1" ref="printer" />
</bean>

```

#### 11)/src/info.javaexpert.test/HelloBeanTest.java 수정

```

...
//2. Hello Beans 가져오기
Hello hello = (Hello)context.getBean("hello2");
...
Hello hello2 = context.getBean("hello2", Hello.class);
...

```

#### 12)Test

~/src/info.javaexpert.test/HelloBeanTest.java > right-click > Run As > Java Application

```

-----
Hello Spring
Hello Spring
true

```

### 20. 생성자 이용하여 의존 주입하기 실습

1)New > Spring Legacy Project > Simple Projects > Simple Spring Maven  
Project Name : SpringDemo1

2)Create Package : src/main/java/info.javaexpert

3)info.javaexpert.Student.java  
package info.javaexpert;

```

public class Student {
    private String name;
    private int age;
    private int grade;
    private int classNum;
    public Student(String name, int age, int grade, int classNum) {
        this.name = name;
        this.age = age;
        this.grade = grade;
        this.classNum = classNum;
    }
}

```

```

1512     }
1513     public String getName() {
1514         return name;
1515     }
1516     public void setName(String name) {
1517         this.name = name;
1518     }
1519     public int getAge() {
1520         return age;
1521     }
1522     public void setAge(int age) {
1523         this.age = age;
1524     }
1525     public int getGrade() {
1526         return grade;
1527     }
1528     public void setGrade(int grade) {
1529         this.grade = grade;
1530     }
1531     public int getClassNum() {
1532         return classNum;
1533     }
1534     public void setClassNum(int classNum) {
1535         this.classNum = classNum;
1536     }
1537 }

```

4)info.javaexpert.StudentInfo.java

```

1539 package info.javaexpert;
1540
1541
1542 public class StudentInfo {
1543     private Student student;
1544
1545     public StudentInfo(Student student) {
1546         this.student = student;
1547     }
1548
1549     public void printInfo(){
1550         if(this.student != null){
1551             System.out.println("Name : " + this.student.getName());
1552             System.out.println("Age : " + this.student.getAge());
1553             System.out.println("Grade : " + this.student.getGrade());
1554             System.out.println("Class : " + this.student.getClassNum());
1555             System.out.println("-----");
1556         }
1557     }
1558
1559     public void setStudent(Student student){
1560         this.student = student;
1561     }
1562 }

```

5)src/main/resources/ApplicationContext.xml

```

1565 <?xml version="1.0" encoding="UTF-8"?>
1566 <beans xmlns="http://www.springframework.org/schema/beans"
1567     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1568     xsi:schemaLocation="http://www.springframework.org/schema/beans
1569         http://www.springframework.org/schema/beans/spring-beans.xsd">
1570
1571     <bean id="student1" class="info.javaexpert.Student">
1572         <constructor-arg>
1573             <value>한지민</value>
1574         </constructor-arg>
1575         <constructor-arg>
1576             <value>15</value>
1577         </constructor-arg>

```



```

1578         <value>2</value>
1579     </constructor-arg>
1580     <constructor-arg>
1581         <value>5</value>
1582     </constructor-arg>
1583 </bean>
1584
1585 <bean id="student2" class="info.javaexpert.Student">
1586     <constructor-arg value="설운도" />
1587     <constructor-arg value="16" />
1588     <constructor-arg value="3" />
1589     <constructor-arg value="7" />
1590 </bean>
1591
1592 <bean id="studentInfo" class="info.javaexpert.StudentInfo">
1593     <constructor-arg>
1594         <ref bean="student1"/>
1595     </constructor-arg>
1596 </bean>
1597 </beans>
1598
1599 6)info.javaexpert.MainClass.java
1600 package info.javaexpert;
1601
1602 import org.springframework.context.support.AbstractApplicationContext;
1603 import org.springframework.context.support.GenericXmlApplicationContext;
1604
1605 public class MainClass {
1606     public static void main(String[] args) {
1607         String configFile = "classpath:ApplicationContext.xml";
1608         AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);
1609         StudentInfo studentInfo = context.getBean("studentInfo", StudentInfo.class);
1610         studentInfo.printInfo();
1611
1612         Student student2 = context.getBean("student2", Student.class);
1613         studentInfo.setStudent(student2);
1614         studentInfo.printInfo();
1615
1616         context.close();
1617     }
1618 }

```

## 21. DI의 장점

1)Java파일의 수정 없이 스프링 설정 파일만을 수정하여 부품들을 생성/조립할 수 있다.

### 2)info.javaexpert.Car.java Interface

```

1625 package kr.co.javaexpert;
1626
1627 public interface Car {
1628     void drive();
1629 }

```

### 3)info.javaexpert.Sonata.java

```

1632 package info.javaexpert;
1633
1634 public class Sonata implements Car {
1635
1636     @Override
1637     public void drive() {
1638         System.out.println("Drive a Sonata");
1639     }
1640 }

```

### 4)info.javaexpert.Carnival.java

```

1643 package info.javaexpert;

```

```

1645 public class Carnival implements Car {
1646
1647     @Override
1648     public void drive() {
1649         System.out.println("Drive a Carnival");
1650     }
1651 }

```

#### 5)info.javaexpert.HybridCar.java

```

1654 package info.javaexpert;
1655
1656 public class HybridCar extends Sonata implements Car {
1657     @Override
1658     public void drive(){
1659         System.out.println("Drive a HybridCar with Sonata");
1660     }
1661 }

```

#### 6)CarContext.xml

```

1664 <?xml version="1.0" encoding="UTF-8"?>
1665 <beans xmlns="http://www.springframework.org/schema/beans"
1666     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1667     xsi:schemaLocation="http://www.springframework.org/schema/beans
1668         http://www.springframework.org/schema/beans/spring-beans.xsd">
1669
1670     <!-- <bean id="car" class="info.javaexpert.Sonata" /> -->
1671     <!-- <bean id="car" class="info.javaexpert.Carnival" /> -->
1672     <bean id="car" class="info.javaexpert.HybridCar" />
1673     //CarMainClass를 변경하지 않고, CarContext.xml만 변경해도 여러 클래스를 이용할 수 있다.
1674 </beans>

```

#### 7)info.javaexpert.CarMainClass.java

```

1676 package info.javaexpert;
1677
1678 import org.springframework.context.support.AbstractApplicationContext;
1679 import org.springframework.context.support.GenericXmlApplicationContext;
1680
1681 public class CarMainClass {
1682     public static void main(String[] args) {
1683         String configFile = "classpath:CarContext.xml";
1684         AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);
1685         Car car = context.getBean("car", Car.class);
1686         car.drive();
1687
1688         context.close();
1689     }
1690 }

```

## 22. Context 파일 여러개 사용하기

1)New > Spring Legacy Project > Simple Projects > Simple Spring Maven  
Project Name : SpringDemo2

2)Create Package : src/main/java/info.javaexpert

#### 3)info.javaexpert.Student.java

```

1700 package info.javaexpert;
1701
1702 import java.util.ArrayList;
1703
1704 public class Student {
1705     private String name;
1706     private int age;
1707     private ArrayList<String> hobbies;
1708     private double height;
1709     private double weight;
1710     public Student(String name, int age, ArrayList<String> hobbies) {

```

```

1711         this.name = name;
1712         this.age = age;
1713         this.hobbys = hobbys;
1714     }
1715     public void setName(String name) {
1716         this.name = name;
1717     }
1718     public void setAge(int age) {
1719         this.age = age;
1720     }
1721     public void setHobbys(ArrayList<String> hobbys) {
1722         this.hobbys = hobbys;
1723     }
1724     public void setHeight(double height) {
1725         this.height = height;
1726     }
1727     public void setWeight(double weight) {
1728         this.weight = weight;
1729     }
1730     @Override
1731     public String toString() {
1732         return String.format("Student [name=%s, age=%s, hobbys=%s, height=%s,
1733             weight=%s]", name, age, hobbys, height,
1734             weight);
1735     }
1736 }

```

#### 4)info.javaexpert.StudentInfo.java

```

1738 package info.javaexpert;
1739 public class StudentInfo {
1740     private Student student;
1741
1742     public Student getStudent() {
1743         return student;
1744     }
1745
1746     public void setStudent(Student student) {
1747         this.student = student;
1748     }
1749 }
1750

```

#### 5)info.javaexpert.Product.java

```

1752 package info.javaexpert;
1753 public class Product {
1754     private String pName;
1755     private int pPrice;
1756     private String maker;
1757     private String color;
1758     public Product(String pName, int pPrice) {
1759         this.pName = pName;
1760         this.pPrice = pPrice;
1761     }
1762     public void setpName(String pName) {
1763         this.pName = pName;
1764     }
1765     public void setpPrice(int pPrice) {
1766         this.pPrice = pPrice;
1767     }
1768     public void setMaker(String maker) {
1769         this.maker = maker;
1770     }
1771     public void setColor(String color) {
1772         this.color = color;
1773     }
1774     @Override
1775     public String toString() {
1776         return String.format("Product [pName=%s, pPrice=%s, maker=%s, color=%s]", pName,

```

```
        pPrice, maker, color);
```

```
1777    }
```

```
1778 }
```

```
1779
```

```
1780 6)src/main/resources/ApplicationContext.xml
```

```
1781 <?xml version="1.0" encoding="UTF-8"?>
```

```
1782 <beans xmlns="http://www.springframework.org/schema/beans"
```

```
1783     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
1784     xsi:schemaLocation="http://www.springframework.org/schema/beans
```

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

```
1785
```

```
1786 <bean id="student1" class="info.javaexpert.Student">
```

```
1787     <constructor-arg value="한지민" />
```

```
1788     <constructor-arg value="25" />
```

```
1789     <constructor-arg>
```

```
1790         <list>
```

```
1791             <value>독서</value>
```

```
1792             <value>영화감상</value>
```

```
1793             <value>요리</value>
```

```
1794         </list>
```

```
1795     </constructor-arg>
```

```
1796     <property name="height" value="165" />
```

```
1797     <property name="weight">
```

```
1798         <value>45</value>
```

```
1799     </property>
```

```
1800 </bean>
```

```
1801
```

```
1802 <bean id="studentInfo1" class="info.javaexpert.StudentInfo">
```

```
1803     <property name="student">
```

```
1804         <ref bean="student1" />
```

```
1805     </property>
```

```
1806 </bean>
```

```
1807 </beans>
```

```
1808
```

```
1809 7)src/main/resources/ApplicationContext1.xml
```

```
1810 <?xml version="1.0" encoding="UTF-8"?>
```

```
1811 <beans xmlns="http://www.springframework.org/schema/beans"
```

```
1812     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
1813     xmlns:c="http://www.springframework.org/schema/c"
```

```
1814     xmlns:p="http://www.springframework.org/schema/p"
```

```
1815     xsi:schemaLocation="http://www.springframework.org/schema/beans
```

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

```
1816
```

```
1817 <bean id="student3" class="info.javaexpert.Student">
```

```
1818     <constructor-arg value="설운도" />
```

```
1819     <constructor-arg value="50" />
```

```
1820     <constructor-arg>
```

```
1821         <list>
```

```
1822             <value>노래부르기</value>
```

```
1823             <value>게임</value>
```

```
1824         </list>
```

```
1825     </constructor-arg>
```

```
1826     <property name="height" value="175" />
```

```
1827     <property name="weight">
```

```
1828         <value>75</value>
```

```
1829     </property>
```

```
1830 </bean>
```

```
1831
```

```
1832 <bean id="product" class="info.javaexpert.Product" c:pName="Computer"
```

```
1833     c:pPrice="2000000" p:maker="Samsung">
```

```
1834     <property name="color" value="Yellow" />
```

```
1835 </bean>
```

```
1836 </beans>
```

```
1837
```

```
1838 8)info.javaexpert.MainClass
```

```
1839     package kr.co.javaexpert;
```

```
1839
```

```

import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

public class MainClass {
    public static void main(String[] args) {
        String configFile = "classpath:ApplicationContext.xml";
        String configFile1 = "classpath:ApplicationContext1.xml";
        AbstractApplicationContext context = new GenericXmlApplicationContext(configFile,
            configFile1);
        Student student1 = context.getBean("student1", Student.class);
        System.out.println(student1);

        StudentInfo studentInfo = context.getBean("studentInfo1", StudentInfo.class);
        Student student2 = studentInfo.getStudent();
        System.out.println(student2);
        if(student1.equals(student2)) System.out.println("Equals");
        else System.out.println("Different");

        Student student3 = context.getBean("student3", Student.class);
        System.out.println(student3);

        if(student1.equals(student3)) System.out.println("Equals");
        else System.out.println("Different");

        Product product = context.getBean("product", Product.class);
        System.out.println(product);
        context.close();
    }
}

```

## 23. Java Annotation을 이용한 DI 설정하기

### 1)@Configuration

```

public class ApplicationConfig{} // @Configuration : 이 클래스는 스프링 설정에 사용되는 클래스
입니다'라고 명시해 주는 어노테이션

```

### 2)@Bean

```

public class Student1(){ } // @Bean : 객체생성

```

### 3)New > Spring Legacy Project > Simple Projects > Simple Spring Maven

Project Name : SpringDemo3

### 4)Create Package : src/main/java/info.javaexpert

### 5)info.javaexpert.Student.java

```

package info.javaexpert;

import java.util.ArrayList;

public class Student {
    private String name;
    private int age;
    private ArrayList<String> hobbies;
    private double height;
    private double weight;
    public Student(String name, int age, ArrayList<String> hobbies) {
        this.name = name;
        this.age = age;
        this.hobbies = hobbies;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public void setHobbies(ArrayList<String> hobbies) {

```

```

1905         this.hobbys = hobbys;
1906     }
1907     public void setHeight(double height) {
1908         this.height = height;
1909     }
1910     public void setWeight(double weight) {
1911         this.weight = weight;
1912     }
1913     @Override
1914     public String toString() {
1915         return String.format("Student [name=%s, age=%s, hobbys=%s, height=%s,
1916             weight=%s]", name, age, hobbys, height,
1917             weight);
1918     }
1919 }

```

#### 6)info.javaexpert.ApplicationConfig.java

```

1921 import org.springframework.context.annotation.Bean;
1922 import org.springframework.context.annotation.Configuration;
1923
1924 @Configuration
1925 public class ApplicationConfig {
1926
1927     @Bean
1928     public Student student1(){
1929         ArrayList<String> hobbys = new ArrayList<String>();
1930         hobbys.add("독서");
1931         hobbys.add("영화감상");
1932         hobbys.add("요리");
1933
1934         Student student = new Student("한지민", 25, hobbys);
1935         student.setHeight(165);
1936         student.setWeight(45);
1937
1938         return student;
1939     }
1940
1941     @Bean
1942     public Student student2(){
1943         ArrayList<String> hobbys = new ArrayList<String>();
1944         hobbys.add("노래부르기");
1945         hobbys.add("게임");
1946         Student student = new Student("설운도", 50, hobbys);
1947         student.setHeight(175);
1948         student.setWeight(75);
1949
1950         return student;
1951     }
1952 }

```

#### 7)info.javaexpert.MainClass.java

```

1955 package info.javaexpert;
1956
1957 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
1958
1959 public class MainClass {
1960     public static void main(String[] args) {
1961         AnnotationConfigApplicationContext context = new
1962             AnnotationConfigApplicationContext(ApplicationConfig.class);
1963         Student student1 = context.getBean("student1", Student.class);
1964         System.out.println(student1);
1965
1966         Student student2 = context.getBean("student2", Student.class);
1967         System.out.println(student2);
1968
1969         context.close();
1970     }

```

1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035

}

1)New > Spring Legacy Project > Simple Projects > Simple Spring Maven  
Project Name : SpringDemo4

```
3)info.javaexpert.Student.java
   package info.javaexpert;
```

```
public class Student {
    private String name;
    private int age;
    private ArrayList<String> hobbies;
    private double height;
    private double weight;
    public Student(String name, int age, ArrayList<String> hobbies) {
        this.name = name;
        this.age = age;
        this.hobbies = hobbies;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public void setHobbies(ArrayList<String> hobbies) {
        this.hobbies = hobbies;
    }
    public void setHeight(double height) {
        this.height = height;
    }
    public void setWeight(double weight) {
        this.weight = weight;
    }
    @Override
    public String toString() {
        return String.format("Student [name=%s, age=%s, hobbies=%s, weight=%s]", name, age, hobbies, height, weight);
    }
}
```

```
package info.javaexpert;

import java.util.ArrayList;
```

## @Configuration

@Bean

```
public Student student1(){
    ArrayList<String> hobbies = new ArrayList<String>();
    hobbies.add("독서");
    hobbies.add("영화감상");
    hobbies.add("요리");
}
```

```
Student student = new Student("한지민", 25, hobbies);
student.setHeight(165);
```

```

2036         student.setWeight(45);
2037
2038     return student;
2039 }
2040 }
2041
2042 5)src/main/resources/ApplicationContext.xml
2043 <?xml version="1.0" encoding="UTF-8"?>
2044 <beans xmlns="http://www.springframework.org/schema/beans"
2045     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2046     xmlns:context="http://www.springframework.org/schema/context"
2047     xsi:schemaLocation="http://www.springframework.org/schema/beans
2048         http://www.springframework.org/schema/beans/spring-beans.xsd">
2049
2050     <bean class="org.springframework.context.annotation.ConfigurationClassPostProcessor" />
2051     <bean class="info.javaexpert.ApplicationConfig" />
2052     <bean id="student3" class="info.javaexpert.Student">
2053         <constructor-arg value="설운도" />
2054         <constructor-arg value="50" />
2055         <constructor-arg>
2056             <list>
2057                 <value>노래부르기</value>
2058                 <value>게임</value>
2059             </list>
2060         </constructor-arg>
2061         <property name="height" value="175" />
2062         <property name="weight">
2063             <value>75</value>
2064         </property>
2065     </bean>
2066 </beans>

```

6)info.javaexpert.MainClass.java

```

2068 package info.javaexpert;
2069
2070 import org.springframework.context.support.AbstractApplicationContext;
2071 import org.springframework.context.support.GenericXmlApplicationContext;
2072
2073 public class MainClass {
2074     public static void main(String[] args) {
2075         String configFile = "classpath:ApplicationContext.xml";
2076         AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);
2077         Student student1 = context.getBean("student1", Student.class);
2078         System.out.println(student1);
2079
2080         Student student3 = context.getBean("student3", Student.class);
2081         System.out.println(student3);
2082     }
2083 }

```

25. Java Annotation과 XML 을 이용한 DI 설정 방법 : Java 파일에 XML 파일을 포함시켜 사용하는 방법

1)New > Spring Legacy Project > Simple Projects > Simple Spring Maven  
Project Name : SpringDemo5

2)Create Package : src/main/java/info.javaexpert

3)info.javaexpert.Student.java

```

2093 package info.javaexpert;
2094
2095 import java.util.ArrayList;
2096
2097 public class Student {
2098     private String name;
2099     private int age;
2100     private ArrayList<String> hobbies;
2101     private double height;

```



```

2102     private double weight;
2103     public Student(String name, int age, ArrayList<String> hobbies) {
2104         this.name = name;
2105         this.age = age;
2106         this.hobbies = hobbies;
2107     }
2108     public void setName(String name) {
2109         this.name = name;
2110     }
2111     public void setAge(int age) {
2112         this.age = age;
2113     }
2114     public void setHobbies(ArrayList<String> hobbies) {
2115         this.hobbies = hobbies;
2116     }
2117     public void setHeight(double height) {
2118         this.height = height;
2119     }
2120     public void setWeight(double weight) {
2121         this.weight = weight;
2122     }
2123     @Override
2124     public String toString() {
2125         return String.format("Student [name=%s, age=%s, hobbies=%s, height=%s,
2126             weight=%s]", name, age, hobbies, height,
2127             weight);
2128     }
2129 }

```

4)src/main/resources/ApplicationContext.xml

```

2131 <?xml version="1.0" encoding="UTF-8"?>
2132 <beans xmlns="http://www.springframework.org/schema/beans"
2133     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2134     xsi:schemaLocation="http://www.springframework.org/schema/beans
2135         http://www.springframework.org/schema/beans/spring-beans.xsd">
2136
2137     <bean id="student3" class="info.javaexpert.Student">
2138         <constructor-arg value="설운도" />
2139         <constructor-arg value="50" />
2140         <constructor-arg>
2141             <list>
2142                 <value>노래부르기</value>
2143                 <value>게임</value>
2144             </list>
2145         </constructor-arg>
2146         <property name="height" value="175" />
2147         <property name="weight">
2148             <value>75</value>
2149         </property>
2150     </bean>
2151 </beans>

```

5)info.javaexpert.ApplicationConfig.java

```

2153 package info.javaexpert;
2154
2155 import java.util.ArrayList;
2156
2157 import org.springframework.context.annotation.Bean;
2158 import org.springframework.context.annotation.Configuration;
2159 import org.springframework.context.annotation.ImportResource;
2160
2161 @Configuration
2162 @ImportResource("classpath:ApplicationContext.xml")
2163 public class ApplicationConfig {
2164
2165     @Bean
2166     public Student student1(){

```

```

2167         ArrayList<String> hobbies = new ArrayList<String>();
2168         hobbies.add("독서");
2169         hobbies.add("영화감상");
2170         hobbies.add("요리");
2171
2172         Student student = new Student("한지민", 25, hobbies);
2173         student.setHeight(165);
2174         student.setWeight(45);
2175
2176         return student;
2177     }
2178 }
2179
2180 6)info.javaexpert.MainClass.java
2181 package info.javaexpert;
2182
2183 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
2184
2185 public class MainClass {
2186     public static void main(String[] args) {
2187         AnnotationConfigApplicationContext context = new
2188             AnnotationConfigApplicationContext(ApplicationConfig.class);
2189         Student student1 = context.getBean("student1", Student.class);
2190         System.out.println(student1);
2191
2192         Student student3 = context.getBean("student3", Student.class);
2193         System.out.println(student3);
2194
2195         context.close();
2196     }
2197 }

```

## 26. Bean 등록 메타 정보 구성 전략

### 1) 전략 1 - XML 단독 사용

- 모든 **Bean**을 명시적으로 **XML**에 등록하는 방법이다.
- 생성되는 모든 **Bean**을 **XML**에서 확인할 수 있다는 장점이 있으나 **Bean**의 갯수가 많아지면 **XML** 파일을 관리하기 번거로울 수 있다.
- 여러 개발자가 같은 설정파일을 공유해서 개발하다보면 설정파일을 동시에 수정하다가 충돌이 일어나는 경우도 적지 않다.
- DI**에 필요한 적절한 **setter** 메소드 또는 **constructor**가 코드 내에 반드시 존재해야 한다.
- 개발 중에는 어노테이션 설정방법을 사용했지만, 운영중에는 관리의 편의성을 위해 **XML** 설정으로 변경하는 전략을 쓸 수도 있다.

### 2) 전략 2 - XML과 Bean Scanning의 혼용

- Bean**으로 사용될 클래스에 특별한 어노테이션을 부여해주면 이런 클래스를 자동으로 찾아서 **Bean**으로 등록한다.
- 특정 어노테이션이 붙은 클래스를 자동으로 찾아서 **Bean**으로 등록해 주는 방식을 **Bean Scanning**을 통한 자동인식 **Bean** 등록기능이라고 한다.
- 어노테이션을 부여하고 자동 스캔으로 빈을 등록하면 **XML** 문서 생성과 관리에 따른 수고를 덜어주고 개발 속도를 향상시킬 수 있다.
- 어플리케이션에 등록될 **bean**이 어떤 것들이 있고, **bean**들 간의 의존관계가 어떻게 되는지를 한눈에 파악할 수 없다는 단점이 있다.

## 27. Bean등록 및 의존관계 설정하는 Annotation

### 1)Bean 등록 Annotation

- @Component** : 컴포넌트를 나타내는 일반적인 스테레오 타입으로 **<bean>** 태그와 동일한 역할을 함
- @Repository** : **Persistence** 레이어, 영속성을 가지는 속성(파일, 데이터베이스)을 가진 클래스
- @Service** : 서비스 레이어, 비즈니스 로직을 가진 클래스
- @Controller** : 프리젠테이션 레이어, 웹 어플리케이션에서 웹 요청과 응답을 처리하는 클래스
- @Repository**, **@Service**, **@Controller**는 더 특정한 유즈케이스에 대한 **@Component**의 구체화된 형태이다.

### 2)Bean 의존관계 주입 Annotation

- @Autowired**, **@Resource** 어노테이션은 의존하는 객체를 자동으로 주입해 주는 어노테이션이다.
- @Autowired**는 타입으로, **@Resource**는 이름으로 연결한다는 점이 다르다.

## -@Autowired

- 정밀한 의존관계 주입(Dependency Injection)이 필요한 경우에 유용하다.
- property, setter 메소드, 생성자, 일반메소드에 적용 가능하다.
- 의존하는 객체를 주입할 때 주로 Type을 이용하게 된다.
- <property>, <constructor-arg>태그와 동일한 역할을 한다.

## -@Resource

- 어플리케이션에서 필요로 하는 자원을 자동 연결할 때 사용된다.
- 프로퍼티, setter 메소드에 적용 가능하다.
- 의존하는 객체를 주입할 때 주로 Name을 이용하게 된다.

## -@Value

- 단순한 값을 주입할 때 사용되는 어노테이션이다.
- @Value("Spring")은 <property ... value="Spring" />와 동일한 역할을 한다.

## -@Qualifier

- @Autowired 어노테이션과 같이 사용되어 진다.
- @Autowired는 타입으로 찾아서 주입하므로, 동일한 타입의 Bean객체가 여러 개 존재할 때 특정 Bean을 찾기 위해서는 @Qualifier를 같이 사용해야 한다.

## 3)Component Scan을 지원하는 태그

### -<context:component-scan> 태그

- @Component를 통해 자동으로 Bean을 등록하고, @Autowired로 의존관계를 주입받는 어노테이션을 클래스에서 선언하여 사용했을 경우에는 해당 클래스가 위치한 특정 패키지를 Scan하기 위한 설정을 XML에 해주어야 한다.

```
<context:component-scan base-package="info.javaexpert" />
```

- <context:include-filter> 태그와 <context:exclude-filter> 태그를 같이 사용하면 자동 스캔 대상에 포함시킬 클래스와 포함시키지 않을 클래스를 구체적으로 명시할 수 있다.

## 4)사용 예

### <SpringPrinter.java>

```
package info.javaexpert;

import org.springframework.stereotype.Component;

@Component("stringPrinter")
public class StringPrinter implements Printer{
    private StringBuffer buffer = new StringBuffer();

    public void print(String message){
        this.buffer.append(message);
    }

    public String toString(){
        return this.buffer.toString();
    }
}
```

## 28. Lab

- 1)In Package Explorer > right-click > New > Java Project  
Project name : DIDemo3

- 2)src > right-click > New > Package  
Package name : info.javaexpert

- 3)POJO class 작성

```
-info.javaexpert > right-click > New > Class
<Hello.java>
package info.javaexpert;

public class Hello{
    private String name;
    private Printer printer;
```

```

2288         public Hello(){}
2289
2290         public void setName(String name){
2291             this.name = name;
2292         }
2293
2294         public void setPrinter(Printer printer){
2295             this.printer = printer;
2296         }
2297
2298         public String sayHello(){
2299             return "Hello " + name;
2300         }
2301
2302         public void print(){
2303             this.printer.print(sayHello());
2304         }
2305     }
2306
2307 -info.javaexpert > right-click > New > Interface
2308     interface name : Printer
2309
2310 <Printer.java>
2311     package info.javaexpert;
2312
2313     public interface Printer{
2314         public void print(String message);
2315     }
2316
2317 -info.javaexpert > right-click > New > Class
2318     Class Name : StringPrinter
2319
2320 <StringPrinter.java>
2321     package info.javaexpert;
2322
2323     public class StringPrinter implements Printer{
2324         private StringBuffer buffer = new StringBuffer();
2325
2326         public void print(String message){
2327             this.buffer.append(message);
2328         }
2329
2330         public String toString(){
2331             return this.buffer.toString();
2332         }
2333     }
2334
2335 -info.javaexpert > right-click > New > Class
2336     Class Name : ConsolePrinter
2337
2338 <ConsolePrinter.java>
2339     package info.javaexpert;
2340
2341     public class ConsolePrinter implements Printer{
2342         public void print(String message){
2343             System.out.println(message);
2344         }
2345     }
2346
2347 4)Java Project를 Spring Project로 변환
2348     -DIDemo3 Project > right-click > Configuration > Convert to Maven Project
2349         --Project : /DIDemo3
2350         --Group Id : DIDemo3
2351         --Artifact Id : DIDemo3
2352         --version : 0.0.1-SNAPSHOT
2353         --Packaging : jar
2354         --Finish

```

--Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.

-DIDemo3 Project > right-click > Spring Tools > Add Spring Project Nature  
--Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.

-pom.xml 파일에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<!--여기부터 작성 -->
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>4.3.9.RELEASE</version>
    </dependency>
</dependencies>
```

-pom.xml > right-click > Run As > Maven install

5)src/config folder 생성

~/src > right-click > New > Folder  
Folder name : config

6)Bean Configuration XML 작성

~/src/config > right-click > New > Other > Spring > Spring Bean Configuration File  
File name : beans.xml > Next  
Check [beans - <http://www.springframework.org/schema/beans>]  
Check [<http://www.springframework.org/schema/beans/spring-beans-4.3.xsd>]  
Finish

```
<?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-4.3.xsd">

    <bean id="hello" class="info.javaexpert.Hello">
        <property name="name" value="Spring" />
        <property name="printer" ref="printer" />
    </bean>
    <bean id="printer" class="info.javaexpert.StringPrinter" />
    <bean id="consolePrinter" class="info.javaexpert.ConsolePrinter" />

</beans>
```

7)DI Test 클래스 작성

~/src/info.javaexpert > right-click > New > Package  
Package Name : test

~/src/info.javaexpert/test/HelloBeanTest.java

```
package info.javaexpert.test;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

import info.javaexpert.Hello;
import info.javaexpert.Printer;

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

        //2. Hello Beans 가져오기
        Hello hello = (Hello)context.getBean("hello");
        System.out.println(hello.sayHello());
        hello.print();
    }
}
```

```

2421
2422 //3. SpringPrinter 가져오기
2423 Printer printer = (Printer)context.getBean("printer");
2424 System.out.println(printer.toString());
2425
2426 Hello hello2 = context.getBean("hello", Hello.class);
2427 hello2.print();
2428
2429 System.out.println(hello == hello2); //Singleton Pattern
2430 }
2431 }
2432

```

```

2433 -----
2434 Hello Spring
2435 Hello Spring
2436 true
2437

```

#### 8)jUnit Library 설치

-jUnit 4.12 버전을 pom.xml에 추가

```

2440
2441 <dependency>
2442     <groupId>junit</groupId>
2443     <artifactId>junit</artifactId>
2444     <version>4.12</version>
2445     <scope>test</scope>
2446 </dependency>
2447

```

-pom.xml > right-click > Run As > Maven Install

#### 9)jUnit을 사용한 DI 테스트 클래스(HelloBeanJUnitTest.java) 작성

~/src/info.javaexpert.test/HelloBeanTest.java 복사

~/src/info.javaexpert.test/ 붙여넣고 이름변경 -> HelloBeanJUnitTest.java

```

2453
2454 package info.javaexpert.test;
2455
2456 import org.junit.Before;
2457 import org.junit.Test;
2458 import org.springframework.context.ApplicationContext;
2459 import org.springframework.context.support.GenericXmlApplicationContext;
2460
2461 import info.javaexpert.Hello;
2462 import info.javaexpert.Printer;
2463
2464 import static org.junit.Assert.assertEquals;
2465 import static org.junit.Assert.assertSame;
2466
2467 public class HelloBeanJUnitTest {
2468     ApplicationContext context;
2469
2470     @Before
2471     public void init(){
2472         //항상 먼저 ApplicationContext를 생성해야 하기 때문에
2473         //1. IoC Container 생성
2474         context = new GenericXmlApplicationContext("config/beans.xml");
2475     }
2476
2477     @Test
2478     public void test1(){
2479         //2. Hello Beans 가져오기
2480         Hello hello = (Hello)context.getBean("hello");
2481         assertEquals("Hello Spring", hello.sayHello());
2482         hello.print();
2483
2484         //3. SpringPrinter 가져오기
2485         Printer printer = (Printer)context.getBean("printer");
2486         assertEquals("Hello Spring", printer.toString());
2487     }

```

```

2488
2489     @Test
2490     public void test2(){
2491         Hello hello = (Hello)context.getBean("hello");
2492
2493         Hello hello2 = context.getBean("hello", Hello.class);
2494         assertEquals(hello, hello2);
2495     }
2496 }

```

2497  
2498 -right-click > Run As > Junit Test  
2499 -결과 -> Junit View에 초록색 bar

## 2500 10)Spring TestContext Framework

2501 -Spring-Test library 설치  
2502 --pom.xml 수정

```

2503
2504
2505     <dependency>
2506     <groupId>org.springframework</groupId>
2507     <artifactId>spring-test</artifactId>
2508     <version>4.3.9.RELEASE</version>
2509     <scope>test</scope>
2510 </dependency>

```

2511  
2512 -pom.xml > right-click > Maven Install

2513  
2514 -Spring-Test를 사용할 DI 테스트 클래스-HelloBeanJUnitSpringTest.java 작성하기  
2515 --/src/info.javaexpert.test/HelloBeanJUnitTest.java 복사해서  
2516 --/src/info.javaexpert.test/HelloBeanJUnitSpringTest.java 로 붙여넣기

```

2517
2518     import org.junit.runner.RunWith;
2519     import org.springframework.beans.factory.annotation.Autowired;
2520     import org.springframework.test.context.ContextConfiguration;
2521     import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
2522     ...
2523     @RunWith(SpringJUnit4ClassRunner.class)
2524     @ContextConfiguration(locations="classpath:config/beans.xml")
2525     public class HelloBeanJUnitSpringTest {

```

```

2526
2527         @Autowired
2528         ApplicationContext context;
2529

```

2530 -right-click > Run As > Junit Test  
2531 -결과 -> Junit View에 초록색 bar

## 2532 11)src/info.javaexpert/StringPrinter.java 수정

```

2533 package info.javaexpert;
2534
2535
2536     import org.springframework.stereotype.Component;
2537
2538     @Component("stringPrinter")
2539     public class StringPrinter implements Printer{
2540         private StringBuffer buffer = new StringBuffer();
2541         ...

```

## 2542 12)src/info.javaexpert/ConsolePrinter.java 수정

```

2543 package info.javaexpert;
2544
2545
2546     import org.springframework.stereotype.Component;
2547
2548     @Component("consolePrinter")
2549     public class ConsolePrinter implements Printer{
2550         ...

```

## 2552 13)/src/info.javaexpert/Hello.java 수정

```

2553 package info.javaexpert;

```

```
2555
2556 import org.springframework.beans.factory.annotation.Autowired;
2557 import org.springframework.beans.factory.annotation.Qualifier;
2558 import org.springframework.beans.factory.annotation.Value;
2559 import org.springframework.stereotype.Component;
```

```
2560
2561 @Component
2562 public class Hello {
2563     @Value("Spring")
2564     private String name;
2565
2566     @Autowired
2567     @Qualifier("stringPrinter")
2568     private Printer printer;
2569
2570     //setter 메소드가 필요 없음.
2571
2572     public String sayHello(){
2573         return "Hello " + name;
2574     }
2575
2576     public void print(){
2577         this.printer.print(sayHello());
2578     }
2579 }
```

2580  
2581 14) 기존의 설정파일과 충돌이 발생하기 때문에 /src/config/beans.xml 삭제

2582  
2583 15) 새로운 설정 파일 생성

```
2584 -/src/config/beans.xml 새로 생성
2585 -/src/config > right-click > New > Spring Bean Configuration File
2586   File name : annos.xml
2587 - Next > context - http://www.springframework.org/schema/context Check
2588 - Check http://www.springframework.org/schema/context/spring-context-4.3.xsd
2589 - Finish
```

```
2590
2591 <?xml version="1.0" encoding="UTF-8"?>
2592 <beans xmlns="http://www.springframework.org/schema/beans"
2593     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2594     xmlns:context="http://www.springframework.org/schema/context"
2595     xsi:schemaLocation="http://www.springframework.org/schema/beans
2596         http://www.springframework.org/schema/beans/spring-beans.xsd
2597         http://www.springframework.org/schema/context
2598         http://www.springframework.org/schema/context/spring-context-4.3.xsd">
2599
2600     <context:component-scan base-package="info.javaexpert" />
2601 </beans>
```

2602  
2603 16)/src/info.javaexpert.test/HelloBeanJUnitSpringTest.java 수정하기

```
2604 package info.javaexpert.test;
2605
2606 import static org.junit.Assert.assertEquals;
2607
2608 import org.junit.Test;
2609 import org.junit.runner.RunWith;
2610 import org.springframework.beans.factory.annotation.Autowired;
2611 import org.springframework.context.ApplicationContext;
2612 import org.springframework.test.context.ContextConfiguration;
2613 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
2614
2615 import info.javaexpert.Hello;
2616
2617 @RunWith(SpringJUnit4ClassRunner.class)
2618 @ContextConfiguration(locations="classpath:config/annos.xml")
2619 public class HelloBeanJUnitSpringTest {
2620     @Autowired
2621     ApplicationContext context;
```



```
2620
2621     @Test
2622     public void test(){
2623         Hello hello = context.getBean("hello", Hello.class);
2624         assertEquals("Hello Spring", hello.sayHello());
2625     }
2626 }
```

2627  
2628 -right-click > Run As > Junit Test

2629 -결과 -> Junit View에 초록색 bar