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.com에서 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>
```

```
            <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>4.3.12.RELEASE</version>
            </dependency>

          </dependencies>
          <build>
            <sourceDirectory>src</sourceDirectory>
            <plugins>
              <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.7.0</version>
                <configuration>
                  <source>1.8</source>
                  <target>1.8</target>
                </configuration>
              </plugin>
            </plugins>
          </build>
        </project>
    -pom.xml > right-click > Run As > Maven Install > BUILD SUCCESS <--at Console Window
    -Dependencies tab에서 spring-context : 5.0.1.RELEASE 설치된 것을 확인 함.


4. 환경설정
  1) Tomcat 새로 설치
    -msi 버전이 아닌 zip 버전으로
    -tomcat-users.xml
      <user name="admin" password="admin" roles="admin-gui,manager-gui" />
    -tomcat home directory 변경
      %CATALINA_HOME%/webapps/homecontext.xml
      <Context path="" docBase="C:/SpringHome" debug="0" reloadable="true"
      crossContext="true"  privileged="true" />

  2) STS 설치
    -http://spring.io/tools/sts


5. BeforeSpring Java Project
  1)com.javasoft.Calculator.java
    package com.javasoft;

    public class Calculator {
      public void addAction(int a, int b){
        System.out.println("Called addAction()");
        System.out.printf("%d + %d = %d\n", a, b, (a + b));
      }
      public void subAction(int a, int b){
        System.out.println("Called subAction()");
        System.out.printf("%d - %d = %d\n", a, b, (a - b));
      }
      public void multiAction(int a, int b){
        System.out.println("Called multiAction()");
        System.out.printf("%d x %d = %d\n", a, b, (a * b));
      }
      public void divAction(int a, int b){
        System.out.println("Called divAction()");
        System.out.printf("%d / %d = %d\n", a, b, (a / b));
      }
    }

  2)com.javasoft.MyCalculator.java
    package com.javasoft;

    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      }
152
153   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      }
170
171
172 6. New > Spring Legacy Project > Simple Projects > Simple Spring Maven
173    -Project Name : StartSpring
174    1)Create package to src/main/java/com.javasoft
175    2)Copy MyCalculator.java, Calculator.java from BeforeSpring project to StartSpring's package
176    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       }
184
185    4)src/main/resources/Mouse Right-click >New > Spring Bean Configuration File >
186       -Name : applicationContext.xml > Finish
187       <?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
          http://www.springframework.org/schema/beans/spring-beans.xsd">
191
192          <bean id="calculator" class="com.javasoft.Calculator" />
```

```
        <bean id="myCalculator" class="com.javasoft.MyCalculator">
            <property name="calculator">
                <ref bean="calculator" />
            </property>
            <property name="firstNum" value="10" />
            <property name="secondNum" value="2" />
        </bean>
    </beans>
```

5)MainClass.java

```
    package com.javasoft;

    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";
            AbstractApplicationContext ctx = new GenericXmlApplicationContext(configFile);
            MyCalculator myCalculator = ctx.getBean("myCalculator", MyCalculator.class);

            myCalculator.add();
            myCalculator.sub();
            myCalculator.multi();
            myCalculator.div();

            ctx.close();
        }
    }
```

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

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

7. IoC(Inversion of Control)

1)개념

-객체의 생성, 생명주기의 관리까지 모든 객체에 대한 제어권이 바뀌었다는 것을 의미

-컴포넌트 의존관계 결정(Component Dependency Resolution), 설정(Configuration) 및 Lifecycle를 해결하기 위한 디자인 패턴

-의존이란 변경에 의해 영향을 받는 관계라는 의미이다.

-한 클래스의 내부 코드가 변경되었을 때 이와 관련된 다른 클래스도 함께 변경해야 한다면 이를 변경에 따른 영향이 전파되는 관계로서 '의존'한다고 표현한다.

-의존하는 대상이 있으면, 그 대상을 구하는 방법이 필요하다.

-가장 쉬운 방법은 의존 대상 객체를 직접 생성하는 것이다.

-그래서 의존받는 클래스를 생성하면 그 클래스가 의존하고 있는 클래스도 동시에 생성이 된다.

-이렇게 클래스 내부에서 직접 의존 객체를 생성하는 것은 쉽지만, 유지 보수 관점에서 보면 문제점이 유발될 수 있다.

2)IoC 컨테이너

-스프링 프레임워크도 객체에 대한 생성 및 생명주기를 관리할 수 있는 기능을 제공하고 있음.

-IoC 컨테이너 기능을 제공한다.

-IoC 컨테이너는 객체의 생성을 책임지고, 의존성을 관리한다.

-POJO의 생성, 초기화, 서비스, 소멸에 대한 권한을 가진다.

-개발자들이 직접 POJO를 생성할 수 있지만 컨테이너에게 맡긴다.

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){
```

```
            this.printer = printer;
        }
    }

-Constructor Injection
    <beans.xml>
        <bean id="hello" class="bean.Hello">   <--bean은 Srping이 관리해주는 객체라는 뜻
            <constructor-arg index="0" value="Spring" />
            <constructor-arg index="1" 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 Hello(String name, Printer printer){}
                this.name = name;
                this.printer = printer;
            }
        }


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

     BeanFactory<interface>
            |
            |
 ApplicationContext<interface>

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

                          BeanFactory<interface>
                                  |
                                  |
            ----------------ApplicationContext<interface>--------------------
            |                     |                                  |
            |                     |                                  |
StaticApplicationContext    GenericXmlApplicationContext   WebApplicationContext<interface>
                                                                     |
                                                                     |
                                                          XmlWebApplicationContext


10. Spring DI 용어
    1)Bean
```

-Spring이 IoC 방식으로 관리하는 객체라는 뜻
-Spring이 직접 생성과 제어를 담당하는 객체를 Bean이라고 부른다.

2)BeanFactory
-Spring의 IoC를 담당하는 핵심 Container
-Bean을 등록, 생성, 조회, 반환하는 기능을 담당.
-이 BeanFactory를 바로 사용하지 않고 이를 확장한 ApplicationContext를 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;
```

```
        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{
            public void print(String message){
                System.out.println(message);
            }
        }

4)Java Project를 Spring Project로 변환
    -DIDemo Project > right-click > Configuration > Convert to Maven Project
        --Project : /DIDemo
        --Group Id : DIDemo
        --Artifact Id : DIDemo
        --version : 0.0.1-SNAPSHOT
        --Packaging : jar
        --Finish
        --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    <--설정 Meta 정보 XML 작성

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)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
-/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
      Printer printer = (Printer)context.getBean("printer");
      System.out.println(printer.toString());

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

      System.out.println(hello == hello2);  //Singleton Pattern
   }
}

------------------------------------------------------------------
Hello Spring
Hello Spring
true
```

12. jUnit의 개요와 특징

1)jUnit의 특징
-TDD의 창시자인 Kent Beck과 디자인 패턴 책의 저자인 Erich Gamma가 작성
-단정(Assert) 메소드로 테스트 케이스의 수행 결과를 판별 --> assertEquals(예상 값, 실제 값)
-jUnit4부터는 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.com에 접근
--jUnit으로 검색
--jUnit 4.12 버전을 pom.xml에 추가

```
583
584         <dependency>
585             <groupId>junit</groupId>
586             <artifactId>junit</artifactId>
587             <version>4.12</version>
588             <scope>test</scope>
589         </dependency>
590
591         --pom.xml > right-click > Run As > Maven Install
592
593       -jUnit에서 테스트를 지원하는 어노테이션
594         --@Test
595           ---이것이 선언된 메소드는 테스트를 수행하는 메소드가 된다.
596           ---jUnit은 각각의 테스트가 서로 영향을 주지 않고 독립적으로 실행됨을 원칙으로 하므로 @Test
              마다 객체를 생성한다.
597
598         --@Ignore
599           ---이것이 선언된 메소드는 테스트를 실행하지 않게 한다.
600
601         --@Before
602           ---이것이 선언된 메소드는 @Test가 실행되기 전에 반드시 실행된다.
603           ---@Test 메소드에서 공통으로 사용하는 코드를 @Before 메소드에 선언하여 사용하면 된다.
604
605         --@After
606           ---이것이 선언된 메소드는 @Test 메소드가 실행된 후 실행된다.
607
608         --@BeforeClass
609           ---이 어노테이션은 @Test 메소드보다 먼저 한번만 수행되어야 할 경우에 사용하면 된다.
610
611         --@AfterClass
612           ---이 어노테이션은 @Test 메소드보다 나중에 한번만 수행되어야 할 경우에 사용하면 된다.
613
614       -테스트 결과를 확인하는 단정(Assert) 메소드 종류
615         --org.junit.Assert
616           +assertArrayEquals(expected, actual)
617           +assertEquals(expected, actual)
618           +assertNotNull(object)
619           +assertSame(expected, actual)
620           +assertTrue(object)
621
622       -assertEquals(a, b)
623         --객체 a와 b가 일치함을 확인
624       -assertArrayEquals(a, b)
625         --배열 a, b가 일치함을 확인
626       -assertSame(a, b)
627         --객체 a, b가 같은 객체임을 확인
628         --assertEquals() 메소드는 값이 같은지를 확인하는 것이고, assertSame() 메소든느 두 객체의
            레퍼런스가 같은지를 확인한다.(==연산자)
629       -assertTrue(a)
630         --조건 a가 참인가를 확인
631       -assertNotNull(a)
632         --객체 a가 null이 아님을 확인한다.
633       -이외에도 다양한 assert 메소드가 존재함
634         http://junit.sourceforge.net/javadoc/org/junit/Assert.html
635
636     3)jUnit을 사용한 DI 테스트 클래스 작성하기
637       -jUnit을 사용한 DI 테스트 클래스(HelloBeanJunitTest.java) 작성
638         --/src/info.javaexpert.test/HelloBeanTest.java 복사
639         --/src/info.javaexpert.test/ 붙여넣고 이름변경 -> HelloBeanJunitTest.java
640
641         package info.javaexpert.test;
642
643         import org.junit.Before;
644         import org.junit.Test;
645         import org.springframework.context.ApplicationContext;
646         import org.springframework.context.support.GenericXmlApplicationContext;
647
```

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

   @Test
   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
   public void test2(){
      Hello hello = (Hello)context.getBean("hello");

      Hello hello2 = context.getBean("hello", Hello.class);
      assertSame(hello, hello2);
   }
}
```

-right-click > Run As > Junit Test
-결과 -> Junit View에 초록색 bar
-만일, test1() 메소드를 jUnit에서 제외하고 싶을 때에는 @Test 옆에 @Ignore를 선언한다.

```
import import org.junit.Ignore;
...
@Test @Ignore
public void test1(){
...
```

-right-click > Run As > Junit Test
--jUnit Test 목록에서 test1()는 실행되지 않는다.

13. Spring TestContext Framework

1)Spring-Test library 설치

-http://mvnrepository.com에서 'spring-test'로 검색
-검색 결과 목록에서 'Spring TestContext Framework' 클릭
-버전 목록에서 4.3.9.RELEASE 클릭
-dependency 복사해서 pom.xml에 붙여넣기

```
   <dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-test</artifactId>
<version>4.3.9.RELEASE</version>
<scope>test</scope>
</dependency>
```

-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(Bean)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.

-/src/info.javaexpert.Hello

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

   public Hello(){}

   public void setName(String name){
      this.name = name;
   ...
```

-/src/config/beans.xml

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

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

```
public class Hello{
   List<String> names;
   public void setNames(List<String> list){
      this.names = list;
   }
}

<bean id="hello" class="info.javaexpert">
   <property name="names">
      <list>
         <value>Spring</value>
         <value>IoC</value>
         <value>DI</value>
      </list>
   </property>
</bean>
```

-Map 타입 : <map>과 <entry> 태그를 이용

```
public class Hello{
   Map<String, Integer> ages;

   public void setAges(Map<String, Integer> ages){
      this.ages = ages;
   }
}

<bean id="hello" class="info.javaexpert.Hello">
   <property name="ages">
      <map>
         <entry key="나훈아" value="30" />
         <entry key="이미자" value="50" />
         <entry key="설운도" value="60" />
      </map>
   </property>
</bean>
```

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
925       -DIDemo Project > right-click > Spring Tools > Add Spring Project Nature
926
927       -pom.xml 파일에 Spring Context Dependency 추가하기
928         <version>0.0.1-SNAPSHOT</version>
929          <!--여기부터 작성 -->
930          <dependencies>
931             <dependency>
932                <groupId>org.springframework</groupId>
933                <artifactId>spring-context</artifactId>
934                <version>4.3.9.RELEASE</version>
935             </dependency>
936          </dependencies>
937
938       -pom.xml > right-click > Run As > Maven install
939
940    5)src/config folder 생성
941       -/src > right-click > New > Folder
942          Folder name : config
943
944    6)Bean Configuration XML 작성
945       -/src/config > right-click > New > Other > Spring > Spring Bean Configuration File
946          File name : beans.xml > Next
947             Check [beans - http://www.springframework.org/schema/beans]
948             Check [http://www.springframework.org/schema/beans/spring-beans-4.3.xsd]
949             Finish
950
951          <?xml version="1.0" encoding="UTF-8"?>
952          <beans xmlns="http://www.springframework.org/schema/beans"
953             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
954             xsi:schemaLocation="http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd">
955
956             <bean id="hello" class="info.javaexpert.Hello">
957                <property name="name" value="Spring" />
958                <property name="printer" ref="printer" />
959             </bean>
960             <bean id="printer" class="info.javaexpert.StringPrinter" />
961             <bean id="consolePrinter" class="info.javaexpert.ConsolePrinter" />
962
963          </beans>
964
965    7)DI Test 클래스 작성
966       -/src/info.javaexpert > right-click > New > Package
967          Package Name : test
968       -/src/info.javaexpert/test/HelloBeanTest.java
969
970          package info.javaexpert.test;
971
972          import org.springframework.context.ApplicationContext;
973          import org.springframework.context.support.GenericXmlApplicationContext;
974
975          import info.javaexpert.Hello;
976          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
1000    8)Test
1001       -/src/info.javaexpert.test/HelloBeanTest.java > right-click > Run As > Java Application
1002       ---------------------------------------------------------------
1003          Hello Spring
1004          Hello Spring
1005          true
1006
1007    9)jUnit으로 테스트
1008       -jUnit Library 설치
1009          --jUnit 4.12 버전을 pom.xml에 추가
1010
1011          <dependency>
1012                <groupId>junit</groupId>
1013                <artifactId>junit</artifactId>
1014                <version>4.12</version>
1015                <scope>test</scope>
1016          </dependency>
1017
1018          --pom.xml > right-click > Run As > Maven Install
1019
1020       -jUnit을 사용한 DI 테스트 클래스(HelloBeanJunitTest.java) 작성
1021          --/src/info.javaexpert.test/HelloBeanTest.java 복사
1022          --/src/info.javaexpert.test/ 붙여넣고 이름변경 -> HelloBeanJunitTest.java
1023
1024          package info.javaexpert.test;
1025
1026          import org.junit.Before;
1027          import org.junit.Test;
1028          import org.springframework.context.ApplicationContext;
1029          import org.springframework.context.support.GenericXmlApplicationContext;
1030
1031          import info.javaexpert.Hello;
1032          import info.javaexpert.Printer;
1033
1034          import static org.junit.Assert.assertEquals;
1035          import static org.junit.Assert.assertSame;
1036
1037          public class HelloBeanJunitTest {
1038             ApplicationContext context;
1039
1040             @Before
1041             public void init(){
1042                context = new GenericXmlApplicationContext("config/beans.xml");
1043             }
```

```
        @Test
        public void test1(){
           Hello hello = (Hello)context.getBean("hello");
           assertEquals("Hello Spring", hello.sayHello());
           hello.print();

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

        @Test
        public void test2(){
           Hello hello = (Hello)context.getBean("hello");

           Hello hello2 = context.getBean("hello", Hello.class);
           assertSame(hello, hello2);
        }
     }
```

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

10)Spring-Test를 사용할 DI 테스트 클래스-HelloBeanJunitSpringTest.java 작성하기
-Spring-Test library 설치
-pom.xml 코드 추가

```
   <dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-test</artifactId>
<version>4.3.9.RELEASE</version>
<scope>test</scope>
</dependency>
```

-pom.xml > right-click > Maven Install

-Spring-Test를 사용할 DI 테스트 클래스-HelloBeanJunitSpringTest.java 작성하기
--/src/info.javaexpert.test/HelloBeanJunitTest.java 복사해서
--/src/info.javaexpert.test/HelloBeanJunitSpringTest.java 로 붙여넣기

```
        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")
        public class HelloBeanJunitSpringTest {

           @Autowired
           ApplicationContext context;
```

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

11)Hello class 수정

```
...
private List<String> names;
...
public void setNames(List<String> list){
   this.names = list;
}

public List<String> getNames(){
   return this.names;
}
...
```

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);
   assertSame(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" />
```

```
        <property name="normal" value="23" />
        <property name="overWeight" value="25" />
        <property name="obesity">
          <value>30</value>
        </property>
      </bean>
      <bean id="myInfo" class="info.javaexpert.MyInfo">
        <property name="name" value="한지민" />
        <property name="height" value="170.5" />
        <property name="weight" value="67" />
        <property name="hobby">
          <list>
            <value>수영</value>
            <value>요리</value>
            <value>독서</value>
          </list>
        </property>
        <property name="bmiCalculator">
          <ref bean="bmiCalculator" />
        </property>
      </bean>
```

5)info.javaexpert.MainClass.java

```
package info.javaexpert;

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

public class MainClass {
    public static void main(String[] args) {
        String configFile = "classpath:ApplicationContext.xml";

        //Spring Container 생성
        AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);

        //Spring Container 에서 객체를 가져옴
        MyInfo myInfo = context.getBean("myInfo", MyInfo.class);

        myInfo.getInfo();
        context.close();
    }
}
```

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

-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{
            public void print(String message){
               System.out.println(message);
            }
         }

   4)Java Project를 Spring Project로 변환
      -DIDemo Project > right-click > Configuration > Convert to Maven Project
         --Project : /DIDemo2
         --Group Id : DIDemo2
         --Artifact Id : DIDemo2
         --version : 0.0.1-SNAPSHOT
         --Packaging : jar
         --Finish
         --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
```

```
            Printer printer = (Printer)context.getBean("printer");
            System.out.println(printer.toString());

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

            System.out.println(hello == hello2);  //Singleton Pattern
         }
      }
```

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;
```

```
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public int getGrade() {
        return grade;
    }
    public void setGrade(int grade) {
        this.grade = grade;
    }
    public int getClassNum() {
        return classNum;
    }
    public void setClassNum(int classNum) {
        this.classNum = classNum;
    }
}
```

4)info.javaexpert.StudentInfo.java

```
package info.javaexpert;

public class StudentInfo {
    private Student student;

    public StudentInfo(Student student) {
        this.student = student;
    }

    public void printInfo(){
        if(this.student != null){
            System.out.println("Name : " + this.student.getName());
            System.out.println("Age : " + this.student.getAge());
            System.out.println("Grade : " + this.student.getGrade());
            System.out.println("Class : " + this.student.getClassNum());
            System.out.println("------------------------");
        }
    }

    public void setStudent(Student student){
        this.student = student;
    }
}
```

5)src/main/resources/ApplicationContext.xml

```
<?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="student1" class="info.javaexpert.Student">
        <constructor-arg>
            <value>한지민</value>
        </constructor-arg>
        <constructor-arg>
            <value>15</value>
        </constructor-arg>
        <constructor-arg>
```

```
            <value>2</value>
         </constructor-arg>
         <constructor-arg>
            <value>5</value>
         </constructor-arg>
      </bean>

      <bean id="student2" class="info.javaexpert.Student">
         <constructor-arg value="설운도" />
         <constructor-arg value="16" />
         <constructor-arg value="3" />
         <constructor-arg value="7" />
      </bean>

      <bean id="studentInfo" class="info.javaexpert.StudentInfo">
         <constructor-arg>
            <ref bean="student1"/>
         </constructor-arg>
      </bean>
   </beans>
```

6)info.javaexpert.MainClass.java

```
   package info.javaexpert;

   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";
         AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);
         StudentInfo studentInfo = context.getBean("studentInfo", StudentInfo.class);
         studentInfo.printInfo();

         Student student2 = context.getBean("student2", Student.class);
         studentInfo.setStudent(student2);
         studentInfo.printInfo();

         context.close();
      }
   }
```

21. DI의 장점

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

2)info.javaexpert.Car.java Interface

```
   package kr.co.javaexpert;

   public interface Car {
      void drive();
   }
```

3)info.javaexpert.Sonata.java

```
   package info.javaexpert;

   public class Sonata implements Car {

      @Override
      public void drive() {
         System.out.println("Drive a Sonata");
      }
   }
```

4)info.javaexpert.Carnival.javaa

```
   package info.javaexpert;
```

```
public class Carnival implements Car {

   @Override
   public void drive() {
      System.out.println("Drive a Carnival");
   }
}
```

5)info.javaexpert.HybridCar.java

```
package info.javaexpert;

public class HybridCar extends Sonata implements Car {
   @Override
   public void drive(){
      System.out.println("Drive a HybridCar with Sonata");
   }
}
```

6)CarContext.xml

```
<?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="car" class="info.javaexpert.Sonata" /> -->
   <!-- <bean id="car" class="info.javaexpert.Carnival" /> -->
   <bean id="car" class="info.javaexpert.HybridCar" />
   //CarMainClass를 변경하지 않고, CarContext.xml만 변경해도 여러 클래스를 이용할 수 있다.
</beans>
```

7)info.javaexpert.CarMainClass.java

```
package info.javaexpert;

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

public class CarMainClass {
   public static void main(String[] args) {
      String configFile = "classpath:CarContext.xml";
      AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);
      Car car = context.getBean("car", Car.class);
      car.drive();

      context.close();
   }
}
```

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

```
package info.javaexpert;

import java.util.ArrayList;

public class Student {
   private String name;
   private int age;
   private ArrayList<String> hobbys;
   private double height;
   private double weight;
   public Student(String name, int age, ArrayList<String> hobbys) {
```

```
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,
                weight=%s]", name, age, hobbys, height,
1733                  weight);
1734         }
1735      }
1736
1737   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
1751   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);
        }
    }

6)src/main/resources/ApplicationContext.xml
    <?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="student1" class="info.javaexpert.Student">
            <constructor-arg value="한지민" />
            <constructor-arg value="25" />
            <constructor-arg>
                <list>
                    <value>독서</value>
                    <value>영화감상</value>
                    <value>요리</value>
                </list>
            </constructor-arg>
            <property name="height" value="165" />
            <property name="weight">
                <value>45</value>
            </property>
        </bean>

        <bean id="studentInfo1" class="info.javaexpert.StudentInfo">
            <property name="student">
                <ref bean="student1" />
            </property>
        </bean>
    </beans>

7)src/main/resources/ApplicationContext1.xml
    <?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:c="http://www.springframework.org/schema/c"
        xmlns:p="http://www.springframework.org/schema/p"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

        <bean id="student3" class="info.javaexpert.Student">
            <constructor-arg value="설운도" />
            <constructor-arg value="50" />
            <constructor-arg>
                <list>
                    <value>노래부르기</value>
                    <value>게임</value>
                </list>
            </constructor-arg>
            <property name="height" value="175" />
            <property name="weight">
                <value>75</value>
            </property>
        </bean>

        <bean id="product" class="info.javaexpert.Product" c:pName="Computer"
        c:pPrice="2000000" p:maker="Samsung">
            <property name="color" value="Yellow" />
        </bean>
    </beans>

8)info.javaexpert.MainClass
    package kr.co.javaexpert;
```

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

```
        this.hobbys = hobbys;
    }
    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, hobbys=%s, height=%s,
        weight=%s]", name, age, hobbys, height,
              weight);
    }
}
```

6)info.javaexpert.ApplicationConfig.java

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ApplicationConfig {

    @Bean
    public Student student1(){
        ArrayList<String> hobbys = new ArrayList<String>();
        hobbys.add("독서");
        hobbys.add("영화감상");
        hobbys.add("요리");

        Student student = new Student("한지민", 25, hobbys);
        student.setHeight(165);
        student.setWeight(45);

        return student;
    }

    @Bean
    public Student student2(){
        ArrayList<String> hobbys = new ArrayList<String>();
        hobbys.add("노래부르기");
        hobbys.add("게임");
        Student student = new Student("설운도", 50, hobbys);
        student.setHeight(175);
        student.setWeight(75);

        return student;
    }
}
```

7)info.javaexpert.MainClass.java

```
package info.javaexpert;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainClass {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context = new
        AnnotationConfigApplicationContext(ApplicationConfig.class);
        Student student1 = context.getBean("student1", Student.class);
        System.out.println(student1);

        Student student2 = context.getBean("student2", Student.class);
        System.out.println(student2);

        context.close();
    }
```

```
    }


24. Java Annotation과 XML 을 이용한 DI 설정 방법 : XML 파일에 Java 파일을 포함시켜 사용하는 방법
  1)New > Spring Legacy Project > Simple Projects > Simple Spring Maven
    Project Name : SpringDemo4

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

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

    import java.util.ArrayList;

    public class Student {
      private String name;
      private int age;
      private ArrayList<String> hobbys;
      private double height;
      private double weight;
      public Student(String name, int age, ArrayList<String> hobbys) {
        this.name = name;
        this.age = age;
        this.hobbys = hobbys;
      }
      public void setName(String name) {
        this.name = name;
      }
      public void setAge(int age) {
        this.age = age;
      }
      public void setHobbys(ArrayList<String> hobbys) {
        this.hobbys = hobbys;
      }
      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, hobbys=%s, height=%s,
        weight=%s]", name, age, hobbys, height,
            weight);
      }
    }

  4)info.javaexpert.ApplicationConfig.java
    package info.javaexpert;

    import java.util.ArrayList;

    import org.springframework.context.annotation.Bean;
    import org.springframework.context.annotation.Configuration;

    @Configuration
    public class ApplicationConfig {
      @Bean
      public Student student1(){
        ArrayList<String> hobbys = new ArrayList<String>();
        hobbys.add("독서");
        hobbys.add("영화감상");
        hobbys.add("요리");

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

```
            student.setWeight(45);

            return student;
        }
    }
```

5)src/main/resources/ApplicationContext.xml

```
<?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/spring-beans.xsd">

    <bean class="org.springframework.context.annotation.ConfigurationClassPostProcessor" />
    <bean class="info.javaexpert.ApplicationConfig" />
    <bean id="student3" class="info.javaexpert.Student">
        <constructor-arg value="설운도" />
        <constructor-arg value="50" />
        <constructor-arg>
            <list>
                <value>노래부르기</value>
                <value>게임</value>
            </list>
        </constructor-arg>
        <property name="height" value="175" />
        <property name="weight">
            <value>75</value>
        </property>
    </bean>
</beans>
```

6)info.javaexpert.MainClass.java

```
package info.javaexpert;

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";
        AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);
        Student student1 = context.getBean("student1", Student.class);
        System.out.println(student1);

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

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

```
package info.javaexpert;

import java.util.ArrayList;

public class Student {
    private String name;
    private int age;
    private ArrayList<String> hobbys;
    private double height;
```

```
2102        private double weight;
2103        public Student(String name, int age, ArrayList<String> hobbys) {
2104           this.name = name;
2105           this.age = age;
2106           this.hobbys = hobbys;
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 setHobbys(ArrayList<String> hobbys) {
2115           this.hobbys = hobbys;
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, hobbys=%s, height=%s,
           weight=%s]", name, age, hobbys, height,
2126                 weight);
2127        }
2128     }
2129
2130  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
           http://www.springframework.org/schema/beans/spring-beans.xsd">
2135
2136        <bean id="student3" class="info.javaexpert.Student">
2137           <constructor-arg value="설운도" />
2138           <constructor-arg value="50" />
2139           <constructor-arg>
2140              <list>
2141                 <value>노래부르기</value>
2142                 <value>게임</value>
2143              </list>
2144           </constructor-arg>
2145           <property name="height" value="175" />
2146           <property name="weight">
2147              <value>75</value>
2148           </property>
2149        </bean>
2150     </beans>
2151
2152  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(){
```

```
            ArrayList<String> hobbys = new ArrayList<String>();
            hobbys.add("독서");
            hobbys.add("영화감상");
            hobbys.add("요리");

            Student student = new Student("한지민", 25, hobbys);
            student.setHeight(165);
            student.setWeight(45);

            return student;
        }
    }
```

6)info.javaexpert.MainClass.java

```
package info.javaexpert;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainClass {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context = new
        AnnotationConfigApplicationContext(ApplicationConfig.class);
        Student student1 = context.getBean("student1", Student.class);
        System.out.println(student1);

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

        context.close();
    }
}
```

26. Bean 등록 메타 정보 구성 전략

1)전략 1 - XML 단독 사용

-모든 Bean을 명시적으로 XML에 등록하는 방법이다.

-생성되는 모든 Bean을 XML에서 확인할 수 있다는 장점이 있으나 Bean의 갯수가 많아지면 XML 파일을 관리하기 번거로울 수 있다.

-여러 개발자가 같은 설정파일을 공유해서 개발하다보면 설정파일을 동시에 수정하다가 충돌이 일어나는 경우도 적지 않다.

-DI에 필요한 적절한 setter 메소드 또는 constructor가 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

-Bean으로 사용될 클래스에 특별한 어노테이션을 부여해주면 이런 클래스를 자동으로 찾아서 Bean으로 등록한다.

-특정 어노테이션이 붙은 클래스를 자동으로 찾아서 Bean으로 등록해 주는 방식을 Bean Scanning을 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 bean이 어떤 것들이 있고, bean들 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, @Controller는 더 특정한 유즈케이스에 대한 @Component의 구체화된 형태이다.

2)Bean 의존관계 주입 Annotation

-@Autowired, @Resource 어노테이션은 의존하는 객체를 자동으로 주입해 주는 어노테이션이다.

-@Autowired는 타입으로, @Resource는 이름으로 연결한다는 점이 다르다.

-@Autowired
--정밀한 의존관계 주입(Depencdency 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;
```

```
        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{
        public void print(String message){
            System.out.println(message);
        }
    }

4)Java Project를 Spring Project로 변환
  -DIDemo3 Project > right-click > Configuration > Convert to Maven Project
    --Project : /DIDemo3
    --Group Id : DIDemo3
    --Artifact Id : DIDemo3
    --version : 0.0.1-SNAPSHOT
    --Packaging : jar
    --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();
```

```
            //3. SpringPrinter 가져오기
            Printer printer = (Printer)context.getBean("printer");
            System.out.println(printer.toString());

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

            System.out.println(hello == hello2);  //Singleton Pattern
        }
    }

    ----------------------------------------------------------------
    Hello Spring
    Hello Spring
    true
```

8)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

9)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(){
            //항상 먼저 ApplicationContext를 생성해야 하기 때문에
            //1. IoC Container 생성
            context = new GenericXmlApplicationContext("config/beans.xml");
        }

        @Test
        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
        public void test2(){
           Hello hello = (Hello)context.getBean("hello");

           Hello hello2 = context.getBean("hello", Hello.class);
           assertSame(hello, hello2);
        }
      }
```

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

10)Spring TestContext Framework
-Spring-Test library 설치
--pom.xml 수정

```
   <dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-test</artifactId>
  <version>4.3.9.RELEASE</version>
  <scope>test</scope>
 </dependency>
```

-pom.xml > right-click > Maven Install

-Spring-Test를 사용할 DI 테스트 클래스-HelloBeanJunitSpringTest.java 작성하기
--/src/info.javaexpert.test/HelloBeanJunitTest.java 복사해서
--/src/info.javaexpert.test/HelloBeanJunitSpringTest.java 로 붙여넣기

```
        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")
        public class HelloBeanJunitSpringTest {

           @Autowired
           ApplicationContext context;
```

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

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

```
package info.javaexpert;

import org.springframework.stereotype.Component;

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

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

```
package info.javaexpert;

import org.springframework.stereotype.Component;

@Component("consolePrinter")
public class ConsolePrinter implements Printer{
...
```

13)/src/info.javaexpert/Hello.java 수정

```
package info.javaexpert;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component
public class Hello {
   @Value("Spring")
   private String name;

   @Autowired
   @Qualifier("stringPrinter")
   private Printer printer;

   //setter 메소드가 필요 없음.

   public String sayHello(){
      return "Hello " + name;
   }

   public void print(){
      this.printer.print(sayHello());
   }
}
```

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

15)새로운 설정 파일 생성

-/src/config/beans.xml 새로 생성

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

File name : annos.xml

- Next > context - http://www.springframework.org/schema/context Check
- Check http://www.springframework.org/schema/context/spring-context-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"
   xmlns:context="http://www.springframework.org/schema/context"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans.xsd
      http://www.springframework.org/schema/context
      http://www.springframework.org/schema/context/spring-context-4.3.xsd">

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

16)/src/info.javaexpert.test/HelloBeanJunitSpringTest.java 수정하기

```
package info.javaexpert.test;

import static org.junit.Assert.assertEquals;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import info.javaexpert.Hello;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations="classpath:config/annos.xml")
public class HelloBeanJunitSpringTest {
   @Autowired
   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
```