1. IoC(Inversion of Control)
  1)개념
    -객체의 생성, 생명주기의 관리까지 모든 객체에 대한 제어권이 바뀌었다는 것을 의미
    -Component 의존관계 결정(Component Dependency Resolution), 설정(Configuration) 및 Lifecycle를 해결하기 위한 design pattern
    -의존이란 변경에 의해 영향을 받는 관계라는 의미이다.
    -한 class의 내부 code가 변경되었을 때 이와 관련된 다른 class도 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 class를 생성하면 그 class가 의존하고 있는 class도 동시에 생성이 된다.
    -이렇게 class 내부에서 직접 의존 객체를 생성하는 것은 쉽지만, 유지 보수 관점에서 보면 문제점이 유발될 수 있다.

  2)IoC container
    -Spring Framework도 객체에 대한 생성 및 생명주기를 관리할 수 있는 기능을 제공하고 있음.
    -IoC container 기능을 제공한다.
    -IoC container는 객체의 생성을 책임지고, 의존성을 관리한다.
    -POJO의 생성, 초기화, service, 소멸에 대한 권한을 가진다.
    -개발자들이 직접 POJO를 생성할 수 있지만 container에게 맡긴다.

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

2. BeforeSpring Java Project
  1)com.example.Calculator.java
```
package com.example;

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.example.MyCalculator.java

```
package com.example;

public class MyCalculator {
  private Calculator calculator;
  private int firstNum;
  private int secondNum;

  public void setFirstNum(int firstNum) {
    this.firstNum = firstNum;
  }
  public void setSecondNum(int secondNum) {
    this.secondNum = secondNum;
  }
  public void setCalculator(Calculator calculator){
    this.calculator = calculator;
  }

  public void add(){
    this.calculator.addAction(firstNum, secondNum);
  }
  public void sub(){
    this.calculator.subAction(firstNum, secondNum);
  }
  public void multi(){
    this.calculator.multiAction(firstNum, secondNum);
  }
  public void div(){
    this.calculator.divAction(firstNum, secondNum);
  }
}
```

3)com.example.MainClass

```
package com.example;

public class MainClass {
  public static void main(String[] args) {
    MyCalculator myCalculator = new MyCalculator();
    myCalculator.setCalculator(new Calculator());

    myCalculator.setFirstNum(10);
    myCalculator.setSecondNum(2);

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

4)Result

```
Called addAction()
10 + 2 = 12
```

```
    Called subAction()
    10 - 2 = 8
    Called multiAction()
    10 x 2 = 20
    Called divAction()
    10 / 2 = 5


3. DI Demo in Spring
  1)New > Java Project
    -Project Name : StartSpring
    -JRE : Use default JRE (currently 'jdk1.8.0_212')
    -Finish
  2)Create package to src : com.example

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

  4)Create class : com.example.MainClass.java
    package com.example;

    public class MainClass {
      public static void main(String[] args) {

      }
    }

  5)Java Project를 Spring Project로 변환
    -StartSpring Project > right-click > Configuration > Convert to Maven Project
      --Project : /StartSpring
      --Group Id : StartSpring
      --Artifact Id : StartSpring
      --version : 0.0.1-SNAPSHOT
      --Packaging : jar
      --Finish
      --Package Explorer에서 보이는 Project icon에 Maven의 'M'자가 보임.

    -StartSpring Project > right-click > Spring > Add Spring Project Nature
      --Package Explorer에서 보이는 Project icon에 'M'자와 Spring의 'S'가 보임.

    -pom.xml file에 Spring Context Dependency 추가하기
      --https://mvnrepository.com에서 spring context로 검색
      --현재 Spring 4.x의 마지막 version인 4.3.24.RELEASE click
      --Copy하여 pom.xml에 paste

      <version>0.0.1-SNAPSHOT</version>
      <dependencies>   <--- dependencies element 추가
        <dependency>   <---여기에 paste
          <groupId>org.springframework</groupId>
          <artifactId>spring-context</artifactId>
          <version>4.3.24.RELEASE</version>
        </dependency>
      </dependencies>

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

```
160         [INFO] BUILD SUCCESS 확인
161
162     6)src/config folder 생성
163       -/src > right-click > New > Folder
164         Folder name : config    <--설정 Meta 정보 XML 작성
165
166     7)Bean Configuration XML 작성
167       -src/config > right-click >New > Spring Bean Configuration File
168       -Name : applicationContext.xml > Finish
169       <?xml version="1.0" encoding="UTF-8"?>
170       <beans xmlns="http://www.springframework.org/schema/beans"
171         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
172         xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
173
174         <bean id="calculator" class="com.example.Calculator" />
175
176         <bean id="myCalculator" class="com.example.MyCalculator">
177           <property name="calculator">
178             <ref bean="calculator" />
179           </property>
180           <property name="firstNum" value="10" />
181           <property name="secondNum" value="2" />
182         </bean>
183       </beans>
184
185     8)MainClass.java
186       package com.javasoft;
187
188       import org.springframework.context.support.AbstractApplicationContext;
189       import org.springframework.context.support.GenericXmlApplicationContext;
190
191       public class MainClass {
192         public static void main(String[] args) {
193           String configFile = "config/applicationContext.xml";
194           AbstractApplicationContext ctx = new GenericXmlApplicationContext(configFile);
195           MyCalculator myCalculator = ctx.getBean("myCalculator", MyCalculator.class);
196
197           myCalculator.add();
198           myCalculator.sub();
199           myCalculator.multi();
200           myCalculator.div();
201
202           ctx.close();
203         }
204       }
205
206     9)Result
207       BeforeSpring과 같음.
208
209
210   4. DI
211     1)DI의 개념
212       -각 class간의 의존관계를 빈 설정(Bean Definition) 정보를 바탕으로 container가 자동으로 연결해 주는 것을 말
```

함.
-개발자들은 단지 bean 설정 file에서 의존관계가 필요하다는 정보를 추가하면 된다.
-객체 reference를 container로부터 주입 받아서, 실행시에 동적으로 의존관계가 생성된다.
-container가 흐름의 주체가 되어 application code에 의존관계를 주입해주는 것이다.
-장점
--code가 단순해진다.
--component 간의 결합도가 제거된다.

2)유형
-Setter Injection
--Setter method를 이용한 의존성 삽입
--의존성을 입력 받는 setter method를 만들고, 이를 통해 의존성을 주입한다.
-Constructor Injection
--생성자를 이용한 의존성 삽입
--필요한 의존성을 포함하는 class의 생성자를 만들고 이를 통해 의존성을 주입한다.
-Method Injection
--일반 method를 이용한 의존성 삽입
--의존성을 입력받는 일반 method를 만들고 이를 통해 의존성을 주입한다.

3)DI를 이용한 class 호출방식
Hello<Class> --> Printer<Interface>
| |
| |
String Printer Console Printer
beans.xml

-Hello class가 직접 String Printer나 Console Printer를 찾아서 사용하는 것이 아니라 설정 file(Spring Bean Configuration File)에 설정하면 container가 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;
          }
        }


5. Spring DI Container의 개념
  1)Spring DI Container가 관리하는 객체를 빈(bean)이라고 하고, 이 빈들을 관리한다는 의미로 container를 빈 팩토리(BeanFactory)라고 부른다.
  2)객체의 생성과 객체 사이의 런타임(run-time) 관계를 DI 관점에서 볼 때는 container를 BeanFactory라고 한다.
  3)Bean Factory에 여러 가지 container 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() method가 정의되어 있음.
    -ApplicationContext
      --Bean을 등록, 생성, 조회, 반환 관리하는 기능은 BeanFactory와 같음.
      --Spring의 각종 부가 service를 추가로 제공함.
      --Spring이 제공하는 ApplicationContext 구현 class가 여러가지 종류가 있음.

                          BeanFactory<interface>
                                    |
                                    |
            ----------------ApplicationContext<interface>--------------------
```

```
            |                          |                                    |
            |                          |                                    |
  StaticApplicationContext    GenericXmlApplicationContext   WebApplicationContext<interface>
                                                                           |
                                                                           |
                                                                XmlWebApplicationContext
```

6. 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 service를 추가로 제공
    -Spring에서는 ApplicationContext를 BeanFactory보다 더 많이 사용
  4)Configuration metadata
    -ApplicationContext 또는 BeanFactory가 IoC를 적용하기 위해 사용하는 meta정보
    -설정 meta정보는 IoC Container에 의해 관리되는 Bean 객체를 생성하고 구성할 때 사용됨.

7. 간단한 DI Project
  1)In Package Explorer > right-click > New > Java Project
    Project name : DIDemo

  2)src > right-click > New > Package
    Package name : com.example

  3)Interface 작성
    -com.example > right-click > New > Interface
      interface name : Printer

    <Printer.java>

```
      package com.example;

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

  4)POJO class 작성
    -com.example > right-click > New > Class
    <Hello.java>

```
      package com.example;

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

        public Hello(){}
```

```
370
371         public void setName(String name){
372           this.name = name;
373         }
374
375         public void setPrinter(Printer printer){
376           this.printer = printer;
377         }
378
379         public String sayHello(){
380           return "Hello " + name;
381         }
382
383         public void print(){
384           this.printer.print(sayHello());
385         }
386       }
387
388   5)Printer interface의 child class 작성하기
389     -com.example > right-click > New > Class
390       --Class Name : StringPrinter
391       --Interfaces : com.example.Printer
392
393     <StringPrinter.java>
394       package com.example;
395
396       public class StringPrinter implements Printer{
397         private StringBuffer buffer = new StringBuffer();
398
399         @Override
400         public void print(String message){
401           this.buffer.append(message);
402         }
403
404         public String toString(){
405           return this.buffer.toString();
406         }
407       }
408
409     -com.example > right-click > New > Class
410       --Class Name : ConsolePrinter
411       --Interface : com.example.Printer
412
413     <ConsolePrinter.java>
414       package com.example;
415
416       public class ConsolePrinter implements Printer{
417
418         @Override
419         public void print(String message){
420           System.out.println(message);
421         }
422       }
423
```

```
6)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

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

  -pom.xml file에 Spring Context Dependency 추가하기
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
      <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>4.3.24.RELEASE</version>
      </dependency>
    </dependencies>

  -pom.xml > right-click > Run As > Maven install
  [INFO] BUILD SUCCESS 확인

7)src/config folder 생성
  -/src > right-click > New > Folder
    Folder name : config

8)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="com.example.Hello">
        <property name="name" value="Spring" />
        <property name="printer" ref="printer" />
      </bean>
      <bean id="printer" class="com.example.StringPrinter" />
      <bean id="consolePrinter" class="com.example.ConsolePrinter" />

    </beans>

9)Beans Graph 사용하기
  -Windows menu > Show View > Other > Spring > Spring Explorer
  -In Spring Explorer
    --DIDemo > Beans > beans.xml > right-click > Open Beans Graphs
```

```
477
478    10)DI Test class 작성
479      -/src/com.example > right-click > New > Package
480        Package Name : test
481      -/src/com.example/test/HelloBeanTest.java
482
483        package com.example.test;
484
485        import org.springframework.context.ApplicationContext;
486        import org.springframework.context.support.GenericXmlApplicationContext;
487
488        import com.example.Hello;
489        import com.example.Printer;
490
491        public class HelloBeanTest {
492          public static void main(String [] args){
493            //1. IoC Container 생성
494            ApplicationContext context =
495                 new GenericXmlApplicationContext("config/beans.xml");
496
497            //2. Hello Beans 가져오기
498            Hello hello = (Hello)context.getBean("hello");
499            System.out.println(hello.sayHello());
500            hello.print();
501
502            //3. SpringPrinter 가져오기
503            Printer printer = (Printer)context.getBean("printer");
504            System.out.println(printer.toString());
505
506            Hello hello2 = context.getBean("hello", Hello.class);
507            hello2.print();
508
509            System.out.println(hello == hello2);  //Singleton Pattern
510          }
511        }
512
513        ------------------------------------------------------------------
514        Hello Spring
515        Hello Spring
516        true
517
518
519  8. jUnit의 개요와 특징
520    1)jUnit의 특징
521      -TDD의 창시자인 Kent Beck과 design pattern 책의 저자인 Erich Gamma가 작성
522      -단정(Assert) method로 test case의 수행 결과를 판별 --> assertEquals(예상 값, 실제 값)
523      -jUnit4부터는 test를 지원하는 annotation 제공, @Test, @Before, @After
524      -각 @Test method가 호출할 때마다 새로운 instance를 생성하여 독립적인 test가 이루어지도록 한다.
525
526    2)jUnit
527      -jUnit Library 설치
528        --http://mvnrepository.com에 접근
529        --jUnit으로 검색
530        --jUnit 4.12 version을 pom.xml에 추가
```

```
531
532         <dependency>
533             <groupId>junit</groupId>
534             <artifactId>junit</artifactId>
535             <version>4.12</version>
536             <scope>test</scope>
537         </dependency>
538
539         --pom.xml > right-click > Run As > Maven Install
540
541       -jUnit에서 test를 지원하는 annotation
542         --@Test
543           ---이것이 선언된 method는 test를 수행하는 method가 된다.
544           ---jUnit은 각각의 test가 서로 영향을 주지 않고 독립적으로 실행됨을 원칙으로 하므로 @Test 마다 객체를 생성
              한다.
545
546         --@Ignore
547           ---이것이 선언된 method는 test를 실행하지 않게 한다.
548
549         --@Before
550           ---이것이 선언된 method는 @Test가 실행되기 전에 반드시 실행된다.
551           ---@Test method에서 공통으로 사용하는 code를 @Before method에 선언하여 사용하면 된다.
552
553         --@After
554           ---이것이 선언된 method는 @Test method가 실행된 후 실행된다.
555
556         --@BeforeClass
557           ---이 annotation은 @Test method보다 먼저 한번만 수행되어야 할 경우에 사용하면 된다.
558
559         --@AfterClass
560           ---이 annotation은 @Test method보다 나중에 한번만 수행되어야 할 경우에 사용하면 된다.
561
562       -test 결과를 확인하는 단정(Assert) method 종류
563         --org.junit.Assert
564           +assertArrayEquals(expected, actual)
565           +assertEquals(expected, actual)
566           +assertNotNull(object)
567           +assertSame(expected, actual)
568           +assertTrue(object)
569
570       -assertEquals(a, b)
571         --객체 a와 b가 일치함을 확인
572       -assertArrayEquals(a, b)
573         --배열 a, b가 일치함을 확인
574       -assertSame(a, b)
575         --객체 a, b가 같은 객체임을 확인
576         --assertEquals() method는 값이 같은지를 확인하는 것이고, assertSame() method는 두 객체의
            reference가 같은지를 확인한다.(==연산자)
577       -assertTrue(a)
578         --조건 a가 참인가를 확인
579       -assertNotNull(a)
580         --객체 a가 null이 아님을 확인한다.
581       -이외에도 다양한 assert method가 존재함
582         http://junit.sourceforge.net/javadoc/org/junit/Assert.html
```

```
583
584   3)jUnit을 사용한 DI test class 작성하기
585     -jUnit을 사용한 DI test class(HelloBeanJunitTest.java) 작성
586       --/src/com.example.test/HelloBeanTest.java 복사
587       --/src/com.example.test/ 붙여넣고 이름변경 -> HelloBeanJunitTest.java
588
589       package com.example.test;
590
591       import org.junit.Before;
592       import org.junit.Test;
593       import org.springframework.context.ApplicationContext;
594       import org.springframework.context.support.GenericXmlApplicationContext;
595
596       import com.example.Hello;
597       import com.example.Printer;
598
599       import static org.junit.Assert.assertEquals;
600       import static org.junit.Assert.assertSame;
601
602       public class HelloBeanJunitTest {
603         ApplicationContext context;
604
605         @Before
606         public void init(){
607           //항상 먼저 ApplicationContext를 생성해야 하기 때문에
608           //1. IoC Container 생성
609           context = new GenericXmlApplicationContext("config/beans.xml");
610         }
611
612         @Test
613         public void test1(){
614           //2. Hello Beans 가져오기
615           Hello hello = (Hello)context.getBean("hello");
616           assertEquals("Hello Spring", hello.sayHello());
617           hello.print();
618
619           //3. SpringPrinter 가져오기
620           Printer printer = (Printer)context.getBean("printer");
621           assertEquals("Hello Spring", printer.toString());
622         }
623
624         @Test
625         public void test2(){
626           Hello hello = (Hello)context.getBean("hello");
627
628           Hello hello2 = context.getBean("hello", Hello.class);
629           assertSame(hello, hello2);
630         }
631       }
632
633     -@Before에 mouse를 올려놓으면 Fix project setup... click
634       --Add archive 'junit-4.12.jar ... > OK
635       --import org.junit...에 mouse를 올려놓으면 Fix project setup... click
636       --Add JUnit 4 library to the build path > OK
```

-right-click > Run As > Junit Test
-결과 -> Junit View에 초록색 bar
-만일, test1() method를 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.

9. Spring TestContext Framework
1)Spring-Test library 설치
-http://mvnrepository.com에서 'spring-test'로 검색
-검색 결과 목록에서 'Spring TestContext Framework' 클릭
-version 목록에서 4.3.24.RELEASE 클릭
-dependency 복사해서 pom.xml에 붙여넣기

```
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-test</artifactId>
<version>4.3.24.RELEASE</version>
<scope>test</scope>
</dependency>
```

-pom.xml > right-click > Maven Install

2)Spring-Test에서 test를 지원하는 annotation
-@RunWith(SpringJUnit4ClassRunner.class)
--jUnit Framework의 test 실행방법을 확장할 때 사용하는 annotation
--SpringJUnit4ClassRunner라는 class를 지정해주면 jUnit이 test를 진행하는 중에 ApplicationContext를 만들고 관리하는 작업을 진행해 준다.
--이 annotation은 각각의 test 별로 객체가 생성되더라도 Singleton의 ApplicationContext를 보장한다.

-@ContextConfiguration
--Spring bean 설정 file의 위치를 지정할 때 사용되는 annotation

-@Autowired
--Spring DI에서 사용되는 특별한 annotation
--해당 변수에 자동으로 빈(Bean)을 매핑해준다.
--Spring bean 설정 file을 읽기 위해 굳이 GenericXmlApplicationContext를 사용할 필요가 없다.

3)Spring-Test를 사용할 DI test class-HelloBeanJunitSpringTest.java 작성하기
-/src/com.example.test/HelloBeanJunitTest.java 복사해서
-/src/com.example.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;
```

```
689         import org.springframework.test.context.ContextConfiguration;
690         import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
691         ...
692         @RunWith(SpringJUnit4ClassRunner.class)
693         @ContextConfiguration(locations="classpath:config/beans.xml")
694         //beans.xml경로를 수정한다. 경로 앞에 classpath:를 넣는다.
695         public class HelloBeanJunitSpringTest {
696
697           @Autowired
698           ApplicationContext context;
699
700     -아래의 init()가 필요 없어짐으로 삭제한다.
701           /*
702           @Before
703           public void init(){
704             //항상 먼저 ApplicationContext를 생성해야 하기 때문에
705             //1. IoC Container 생성
706             context = new GenericXmlApplicationContext("config/beans.xml");
707           }
708           */
709
710     -right-click > Run As > Junit Test
711     -결과 -> Junit View에 초록색 bar
712
713
714 10. Dependency Injection(의존주입) 방법의 종류
715   1)XML file을 이용한 DI 설정 방법
716     -setter 이용하기
717     -생성자 이용하기
718   2)Java Annotation 이용한 DI 설정 방법
719   3)Java Annotation과 XML 을 이용한 DI 설정 방법
720     -XML file에 Java file을 포함시켜 사용하는 방법
721     -Java file에 XML file을 포함시켜 사용하는 방법
722
723
724 11. setter를 이용한 의존주입하기 -> Setter Injection
725   1)setter method를 통해 의존 관계가 있는 bean을 주입하려면 <property> 태그를 사용할 수 있다.
726   2)ref 속성은 사용하면 bean이름을 이용해서 주입할 bean을 찾는다.
727   3)value 속성은 단순 값 또는 bean이 아닌 객체를 주입할 때 사용한다.
728   4)단순 값(문자열이나 숫자)의 주입
729     -setter method를 통해 bean의 레퍼런스가 아니라 단순 값을 주입하려고 할 때는 <property> 태그의 value속
      성을 사용한다.
730
731     -/src/com.example.Hello
732       public class Hello {
733         private String name;
734         private Printer printer;
735
736         public Hello(){}
737
738         public void setName(String name){
739           this.name = name;
740         ...
741
```

```
742     -/src/config/beans.xml
743       <bean id="hello" class="com.example.Hello">
744         <property name="name" value="Spring" />
745         <property name="printer" ref="printer" />
746       </bean>
747
748   5)Collection 타입의 값 주입
749     -Spring은 List, Set, Map, Properties와 같은 Collection 타입을 XML로 작성해서 property에 주입하는 방법
        을 제공한다.
750     -List 타입 : <list>와 <value> 태그를 이용
751     -Set 타입 : <set>과 <value> 태그를 이용
752
753
754       public class Hello{
755         List<String> names;
756         public void setNames(List<String> list){
757           this.names = list;
758         }
759       }
760
761     <bean id="hello" class="com.example">
762       <property name="names">
763         <list>
764           <value>Spring</value>
765           <value>IoC</value>
766           <value>DI</value>
767         </list>
768       </property>
769       <property name="foods">
770         <set>
771           <value>Chicken</value>
772           <value>Pizza</value>
773           <value>Bread</value>
774         </set>
775       </property>
776     </bean>
777
778     -Map 타입 : <map>과 <entry> 태그를 이용
779
780       public class Hello{
781         Map<String, Integer> ages;
782
783         public void setAges(Map<String, Integer> ages){
784           this.ages = ages;
785         }
786       }
787
788       <bean id="hello" class="com.example.Hello">
789         <property name="ages">
790           <map>
791             <entry key="나훈아" value="30" />
792             <entry key="이미자" value="50" />
793             <entry>
794               <key>
```

```
                <value>설운도</value>
              </key>
              <value>60</value>
            </entry>
          </map>
        </property>
      </bean>

    -Properties 타입 : <props>와 <prop>를 이용

      <bean id="hello" class="com.example.Hello">
        <property name="ages">
          <props>
            <prop key="나훈아">서울시 강남구 역삼동</prop>
            <prop key="이미자">경기도 수원시 장안구</prop>
          </props>
        </property>
      </bean>

    -null값 추가
      <set>
        <value>Element 1</value>
        <value>Element 2</value>
        <null />
      </set>

      <map>
        <entry>
          <key>
            <null />
          </key>
          <null />
        </entry>
      </map>

  6)배열의 값 지정
    <property name="">
      <array>
        <value>1</value>
        <value>2</value>
      </array>
    </property>
```

7)실제 application 개발 scenario에서 사용되는 Spring bean의 속성과 생성자 인자 형식은 String 형식, 다른 bean의 참조, 여러 표준 형식(java.util.Date, java.util.Map 등등등) 또는 사용자 지정 형식(예, Address)까지 매우 다양하다.

8)java.util.Date, java.util.Currency, 기본 형식 등의 bean 속성과 생성자 인자를 간편하게 전달하기 위해 Spring 에서는 기본적으로 PropertyEditor를 제공하고 있다.

12. setter를 이용한 의존주입하기 실습

1)In Package Explorer > right-click > New > Java Project
  Project name : DIDemo1

2)src > right-click > New > Package
Package name : com.example

3)POJO class 작성
-com.example > right-click > New > Class
<Hello.java>

```
package com.example;

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());
  }
}
```

-com.example > right-click > New > Interface
interface name : Printer

<Printer.java>

```
package com.example;

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

-com.example > right-click > New > Class
Class Name : StringPrinter

<StringPrinter.java>

```
package com.example;

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

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

```
900
901         public String toString(){
902           return this.buffer.toString();
903         }
904       }
905
906     -com.example > right-click > New > Class
907       Class Name : ConsolePrinter
908
909     <ConsolePrinter.java>
910       package com.example;
911
912       public class ConsolePrinter implements Printer{
913
914         @Override
915         public void print(String message){
916           System.out.println(message);
917         }
918       }
919
920   4)Java Project를 Spring Project로 변환
921     -DIDemo Project > right-click > Configuration > Convert to Maven Project
922       --Project : /DIDemo1
923       --Group Id : DIDemo1
924       --Artifact Id : DIDemo1
925       --version : 0.0.1-SNAPSHOT
926       --Packaging : jar
927       --Finish
928
929     -DIDemo Project > right-click > Spring > Add Spring Project Nature
930
931     -pom.xml file에 Spring Context Dependency 추가하기
932      <version>0.0.1-SNAPSHOT</version>
933       <dependencies>
934         <dependency>
935           <groupId>org.springframework</groupId>
936           <artifactId>spring-context</artifactId>
937           <version>4.3.24.RELEASE</version>
938         </dependency>
939       </dependencies>
940
941     -pom.xml > right-click > Run As > Maven install
942
943   5)src/config folder 생성
944     -/src > right-click > New > Folder
945       Folder name : config
946
947   6)Bean Configuration XML 작성
948     -/src/config > right-click > New > Other > Spring > Spring Bean Configuration File
949       File name : beans.xml > Next
950         Check [beans - http://www.springframework.org/schema/beans]
951         Check [http://www.springframework.org/schema/beans/spring-beans-4.3.xsd]
952         Finish
953
```

```
954         <?xml version="1.0" encoding="UTF-8"?>
955         <beans xmlns="http://www.springframework.org/schema/beans"
956           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
957           xsi:schemaLocation="http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans.xsd">
958
959           <bean id="hello" class="com.example.Hello">
960             <property name="name" value="Spring" />
961             <property name="printer" ref="printer" />
962           </bean>
963           <bean id="printer" class="com.example.StringPrinter" />
964           <bean id="consolePrinter" class="com.example.ConsolePrinter" />
965
966         </beans>
967
968     7)DI Test class 작성
969       -/src/com.example > right-click > New > Package
970         Package Name : test
971       -/src/com.example/test/HelloBeanTest.java
972
973         package com.example.test;
974
975         import org.springframework.context.ApplicationContext;
976         import org.springframework.context.support.GenericXmlApplicationContext;
977
978         import com.example.Hello;
979         import com.example.Printer;
980
981         public class HelloBeanTest {
982           public static void main(String [] args){
983             //1. IoC Container 생성
984             ApplicationContext context =
985                   new GenericXmlApplicationContext("config/beans.xml");
986
987             //2. Hello Beans 가져오기
988             Hello hello = (Hello)context.getBean("hello");
989             System.out.println(hello.sayHello());
990             hello.print();
991
992             //3. SpringPrinter 가져오기
993             Printer printer = (Printer)context.getBean("printer");
994             System.out.println(printer.toString());
995
996             Hello hello2 = context.getBean("hello", Hello.class);
997             hello2.print();
998
999             System.out.println(hello == hello2);
1000          }
1001        }
1002
1003    8)Test
1004      -/src/com.example.test/HelloBeanTest.java > right-click > Run As > Java Application
1005      ---------------------------------------------------------------
1006        Hello Spring
```

```
1007        Hello Spring
1008        true
1009
1010    9)jUnit으로 test
1011      -jUnit Library 설치
1012        --jUnit 4.12 version을 pom.xml에 추가
1013
1014        <dependency>
1015            <groupId>junit</groupId>
1016            <artifactId>junit</artifactId>
1017            <version>4.12</version>
1018            <scope>test</scope>
1019        </dependency>
1020
1021        --pom.xml > right-click > Run As > Maven Install
1022
1023      -jUnit을 사용한 DI test class(HelloBeanJunitTest.java) 작성
1024        --/src/com.example.test/HelloBeanTest.java 복사
1025        --/src/com.example.test/ 붙여넣고 이름변경 -> HelloBeanJunitTest.java
1026
1027        package com.example.test;
1028
1029        import org.junit.Before;
1030        import org.junit.Test;
1031        import org.springframework.context.ApplicationContext;
1032        import org.springframework.context.support.GenericXmlApplicationContext;
1033
1034        import com.example.Hello;
1035        import com.example.Printer;
1036
1037        import static org.junit.Assert.assertEquals;
1038        import static org.junit.Assert.assertSame;
1039
1040        public class HelloBeanJunitTest {
1041          ApplicationContext context;
1042
1043          @Before
1044          public void init(){
1045            context = new GenericXmlApplicationContext("config/beans.xml");
1046          }
1047
1048          @Test
1049          public void test1(){
1050            Hello hello = (Hello)context.getBean("hello");
1051            assertEquals("Hello Spring", hello.sayHello());
1052            hello.print();
1053
1054            Printer printer = (Printer)context.getBean("printer");
1055            assertEquals("Hello Spring", printer.toString());
1056          }
1057
1058          @Test
1059          public void test2(){
1060            Hello hello = (Hello)context.getBean("hello");
```

```
1061
1062            Hello hello2 = context.getBean("hello", Hello.class);
1063            assertSame(hello, hello2);
1064          }
1065        }
1066
1067      -right-click > Run As > Junit Test
1068      -결과 -> Junit View에 초록색 bar
1069
1070    10)Spring-Test를 사용할 DI test class-HelloBeanJunitSpringTest.java 작성하기
1071      -Spring-Test library 설치
1072      -pom.xml code 추가
1073        <dependency>
1074       <groupId>org.springframework</groupId>
1075       <artifactId>spring-test</artifactId>
1076       <version>4.3.9.RELEASE</version>
1077       <scope>test</scope>
1078      </dependency>
1079
1080      -pom.xml > right-click > Maven Install
1081
1082      -Spring-Test를 사용할 DI test class-HelloBeanJunitSpringTest.java 작성하기
1083        --/src/com.example.test/HelloBeanJunitTest.java 복사해서
1084        --/src/com.example.test/HelloBeanJunitSpringTest.java 로 붙여넣기
1085
1086          import org.junit.runner.RunWith;
1087          import org.springframework.beans.factory.annotation.Autowired;
1088          import org.springframework.test.context.ContextConfiguration;
1089          import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
1090          ...
1091          @RunWith(SpringJUnit4ClassRunner.class)
1092          @ContextConfiguration(locations="classpath:config/beans.xml")
1093          public class HelloBeanJunitSpringTest {
1094
1095            @Autowired
1096            ApplicationContext context;
1097
1098      -right-click > Run As > Junit Test
1099      -결과 -> Junit View에 초록색 bar
1100
1101    11)Hello class 수정
1102
1103      ...
1104      private List<String> names;
1105      ...
1106      public void setNames(List<String> list){
1107        this.names = list;
1108      }
1109
1110      public List<String> getNames(){
1111        return this.names;
1112      }
1113      ...
1114
```

12)beans.xml 수정

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

13)HelloBeanJunitTest로 test하기

```
@Test <--@Ignore 붙여서 test하지 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 code 수정하기
public void test2(){
  Hello hello = (Hello)context.getBean("hello");

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

  //아래 code 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

13. setter를 이용한 의존주입하기 실습

1)In Package Explorer > right-click > New > Java Project
-Project Name : SpringDemo
2)src > right-click > New > Package
Package name : com.example

3)POJO class 작성
-com.example > right-click > New > Class
-com.example.BmiCalculator.java

```
1169    package com.example;
1170
1171    public class BmiCalculator {
1172      private double lowWeight;
1173      private double normal;
1174      private double overWeight;
1175      private double obesity;
1176
1177      public void setLowWeight(double lowWeight) {
1178        this.lowWeight = lowWeight;
1179      }
1180
1181      public void setNormal(double normal) {
1182        this.normal = normal;
1183      }
1184
1185      public void setOverWeight(double overWeight) {
1186        this.overWeight = overWeight;
1187      }
1188
1189      public void setObesity(double obesity) {
1190        this.obesity = obesity;
1191      }
1192      public void bmiCalcu(double weight, double height){
1193        double h = height * 0.01;
1194        double result = weight / (h * h);
1195
1196        System.out.println("BMI 지수 : " + (int)result);
1197
1198        if(result > obesity)
1199          System.out.println("비만입니다.");
1200        else if(result > overWeight)
1201          System.out.println("과체중입니다.");
1202        else if(result > normal)
1203          System.out.println("정상입니다.");
1204        else
1205          System.out.println("저체중입니다.");
1206      }
1207    }
1208
1209  4)com.example.MyInfo.java
1210    package com.example;
1211
1212    import java.util.ArrayList;
1213
1214    public class MyInfo {
1215      private String name;
1216      private double height;
1217      private double weight;
1218      private ArrayList<String> hobby;
1219      private BmiCalculator bmiCalculator;
1220
1221      public void setBmiCalculator(BmiCalculator bmiCalculator) {
1222        this.bmiCalculator = bmiCalculator;
```

```
1223        }
1224        public void setName(String name) {
1225          this.name = name;
1226        }
1227        public void setHeight(double height) {
1228          this.height = height;
1229        }
1230        public void setWeight(double weight) {
1231          this.weight = weight;
1232        }
1233        public void setHobby(ArrayList<String> hobby) {
1234          this.hobby = hobby;
1235        }
1236        public void getInfo(){
1237          System.out.println("Name : " + this.name);
1238          System.out.println("Height : " + this.height);
1239          System.out.println("Weight : " + this.weight);
1240          System.out.println("Hobby : " + this.hobby);
1241          this.bmiCalcu();
1242        }
1243        public void bmiCalcu(){
1244          this.bmiCalculator.bmiCalcu(this.weight, this.height);
1245        }
1246      }
1247
1248    5)Java Project를 Spring Project로 변환
1249      -SpringDemo Project > right-click > Configuration > Convert to Maven Project
1250        --Project : /SpringDemo
1251        --Group Id : SpringDemo
1252        --Artifact Id : SpringDemo
1253        --version : 0.0.1-SNAPSHOT
1254        --Packaging : jar
1255        --Finish
1256
1257      -SpringDemo Project > right-click > Spring > Add Spring Project Nature
1258
1259      -pom.xml file에 Spring Context Dependency 추가하기
1260       <version>0.0.1-SNAPSHOT</version>
1261        <dependencies>
1262          <dependency>
1263            <groupId>org.springframework</groupId>
1264            <artifactId>spring-context</artifactId>
1265            <version>4.3.24.RELEASE</version>
1266          </dependency>
1267        </dependencies>
1268
1269      -pom.xml > right-click > Run As > Maven install
1270      [INFO] BUILD SUCCESS 확인
1271
1272    6)SpringDemo/resources folder 생성
1273      -SpringDemo project > right-click > Build Path > Configure Build Path
1274      -Source Tab > Add Folder
1275      -SpringDemo click
1276      -Create New Folder > Folder name : resources > Finish > OK
```

-SpringDemo/resources(new) 확인
-Apply and Close

7)Bean Configuration XML 작성
-SpringDemo/resources > right-click > New > Other > Spring > Spring Bean Configuration File
-File name : applicationContext.xml > Finish

```
<bean id="bmiCalculator" class="com.example.BmiCalculator">
  <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="com.example.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>
```

8)com.example.MainClass.java

```
package com.example;

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();
  }
}
```

9)결과

```
    Name : 한지민
    Height : 170.5
    Weight : 67.0
    Hobby : [수영, 요리, 독서]
    BMI 지수 : 23
    정상입니다.


14. 생성자 이용하여 의존 주입하기 -> Constructor Injection
  1)Constructor를 통해 의존관계가 있는 Bean을 주입하려면 <constructor-arg> tag를 사용할 수 있다.
  2)Constructor 주입방식은 생성자의 parameter를 이용하기 때문에 한번에 여러 개의 객체를 주입할 수 있다.


15. 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 : com.example

  3)POJO class 작성
    -com.example > right-click > New > Class
    <Hello.java>
      package com.example;

      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());
        }
      }

    -com.example > right-click > New > Interface
      interface name : Printer

    <Printer.java>
      package com.example;
```

```
1384        public interface Printer{
1385          void print(String message);
1386        }
1387
1388      -com.example > right-click > New > Class
1389        Class Name : StringPrinter
1390
1391      <StringPrinter.java>
1392        package com.example;
1393
1394        public class StringPrinter implements Printer{
1395          private StringBuffer buffer = new StringBuffer();
1396
1397          @Override
1398          public void print(String message){
1399            this.buffer.append(message);
1400          }
1401
1402          public String toString(){
1403            return this.buffer.toString();
1404          }
1405        }
1406
1407      -com.example > right-click > New > Class
1408        Class Name : ConsolePrinter
1409
1410      <ConsolePrinter.java>
1411        package com.example;
1412
1413        public class ConsolePrinter implements Printer{
1414
1415          @Override
1416          public void print(String message){
1417            System.out.println(message);
1418          }
1419        }
1420
1421    4)Java Project를 Spring Project로 변환
1422      -DIDemo2 Project > right-click > Configuration > Convert to Maven Project
1423        --Project : /DIDemo2
1424        --Group Id : DIDemo2
1425        --Artifact Id : DIDemo2
1426        --version : 0.0.1-SNAPSHOT
1427        --Packaging : jar
1428        --Finish
1429
1430      -DIDemo2 Project > right-click > Spring > Add Spring Project Nature
1431
1432      -pom.xml file에 Spring Context Dependency 추가하기
1433        <version>0.0.1-SNAPSHOT</version>
1434        <dependencies>
1435          <dependency>
1436            <groupId>org.springframework</groupId>
1437            <artifactId>spring-context</artifactId>
```

```
            <version>4.3.24.RELEASE</version>
          </dependency>
        </dependencies>

    -pom.xml > right-click > Run As > Maven install
    [INFO] BUILD SUCCESS 확인

  5)DIDemo2/resources folder 생성
    -DIDemo2 project > right-click > Build Path > Configure Build Path
    -Source Tab > Add Folder
    -DIDemo2 click
    -Create New Folder > Folder name : resources > Finish > OK
    -DIDemo2/resources(new) 확인
    -Apply and Close

  6)Bean Configuration XML 작성
    -DIDemo2/resources > right-click > New > Other > Spring > Spring Bean Configuration File
    -File name : beans.xml > 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="com.example.Hello">
          <property name="name" value="Spring" />
          <property name="printer" ref="printer" />
        </bean>
        <bean id="printer" class="com.example.StringPrinter" />
        <bean id="consolePrinter" class="com.example.ConsolePrinter" />

      </beans>

  7)Test class 작성
    -/src/com.example > right-click > New > Package
      Package Name : test
    -/src/com.example/test/HelloBeanTest.java

      package com.example.test;

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

      import com.example.Hello;
      import com.example.Printer;

      public class HelloBeanTest {
        public static void main(String [] args){
          //1. IoC Container 생성
          ApplicationContext context =
              new GenericXmlApplicationContext("classpath: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/com.example.test/HelloBeanTest.java > right-click > Run As > Java Application

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

9)/src/com.example.Hello 생성자 추가

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

10)/resources/beans.xml에 추가

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

11)/src/com.example.test/HelloBeanTest.java 수정

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

12)Test

-/src/com.example.test/HelloBeanTest.java > right-click > Run As > Java Application

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

16. 생성자 이용하여 의존 주입하기 실습

1)In Package Explorer > right-click > New > Java Project
-Project Name : SpringDemo1

2)src > right-click > New > Package
-Package name : com.example

3)com.example.Student.java

```
package com.example;

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)com.example.StudentInfo.java

```
package com.example;

public class StudentInfo {
  private Student student;

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

```
1599        }
1600
1601        public void printInfo(){
1602          if(this.student != null){
1603            System.out.println("Name : " + this.student.getName());
1604            System.out.println("Age : " + this.student.getAge());
1605            System.out.println("Grade : " + this.student.getGrade());
1606            System.out.println("Class : " + this.student.getClassNum());
1607            System.out.println("------------------------");
1608          }
1609        }
1610
1611        public void setStudent(Student student){
1612          this.student = student;
1613        }
1614      }
1615
1616    5)Java Project를 Spring Project로 변환
1617      -SpringDemo1 Project > right-click > Configuration > Convert to Maven Project
1618        --Project : /SpringDemo1
1619        --Group Id : SpringDemo1
1620        --Artifact Id : SpringDemo1
1621        --version : 0.0.1-SNAPSHOT
1622        --Packaging : jar
1623        --Finish
1624
1625      -SpringDemo1 Project > right-click > Spring > Add Spring Project Nature
1626
1627      -pom.xml file에 Spring Context Dependency 추가하기
1628       <version>0.0.1-SNAPSHOT</version>
1629        <dependencies>
1630          <dependency>
1631            <groupId>org.springframework</groupId>
1632            <artifactId>spring-context</artifactId>
1633            <version>4.3.24.RELEASE</version>
1634          </dependency>
1635        </dependencies>
1636
1637      -pom.xml > right-click > Run As > Maven install
1638      [INFO] BUILD SUCCESS 확인
1639
1640    6)SpringDemo1/resources folder 생성
1641      -SpringDemo1 project > right-click > Build Path > Configure Build Path
1642      -Source Tab > Add Folder
1643      -SpringDemo1 click
1644      -Create New Folder > Folder name : resources > Finish > OK
1645      -SpringDemo1/resources(new) 확인
1646      -Apply and Close
1647
1648    7)Bean Configuration XML 작성
1649      -SpringDemo1/resources > right-click > New > Other > Spring > Spring Bean Configuration
          File
1650      -File name : applicationContext.xml > Finish
1651
```

```
<?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="com.example.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="com.example.Student">
    <constructor-arg value="설운도" />
    <constructor-arg value="16" />
    <constructor-arg value="3" />
    <constructor-arg value="7" />
  </bean>

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

8)com.example.MainClass.java

```
package com.example;

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();
  }
```

```
    }
```

9)결과

```
Name : 한지민
Age : 15
Grade : 2
Class : 5
-----------------------
Name : 설운도
Age : 16
Grade : 3
Class : 7
-----------------------
```

17. 표준 Java 형식 및 사용자 지정 형식을 지정하는 생성자 인자

1)생성자 인자 형식이 기본 형식(int, long, boolean 등...), String 형식 또는 사용자 지정 형식(Address 같은...)인 경우 <constructor-arg> 요소의 value 속성으로 값을 지정한다.

2)그런데, value 속성으로 지정한 문자열 값을 두 개 이상의 생성자 인자로 변환할 수 있는 경우 Spring container가 생성자 인자의 형식을 유추할 수 없다.

3)예를 들어, 값이 int, long 또는 String 중 어떤 형식인지 알 수 없는 경우가 있다.

4)이를 경우 type 속성을 사용해 생성자 인자의 형식을 명시적으로 지정할 수 있다.

5)위의 Lab에서 Student class의 생성자를 보면

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

6)Student class에 주입되는 bean 설정 file의 형식은 아래와 같다.

```
<bean id="student2" class="com.example.Student">
  <constructor-arg value="설운도" />
  <constructor-arg value="16" />
  <constructor-arg value="3" />
  <constructor-arg value="7" />
</bean>
```

7)Spring container는 Student bean 정의에서 <constructor-arg> 요소가 나온 순서대로 생성자에 적용한다.

8)이러한 모호함을 해결하기 위해서 다음 예제처럼 type속성으로 생성자 인자의 형식을 지정할 수 있다.

```
<bean id="student2" class="com.example.Student">
  <constructor-arg type="java.lang.String" value="설운도" />
  <constructor-arg type="int" value="16" />
  <constructor-arg type="int" value="3" />
  <constructor-arg type="int" value="7" />
</bean>
```

18. 이름을 기준으로 한 생성자 인자 연결

1)<constructor-arg>요소의 name 속성에는 <constructor-arg> 요소가 적용되는 생성자 인자의 이름을 지정한다.

2)이전의 Lab에서 Student class의 생성자를 보면

```
1756      public Student(String name, int age, int grade, int classNum) {
1757        this.name = name;
1758        this.age = age;
1759        this.grade = grade;
1760        this.classNum = classNum;
1761      }
1762
```

3)<constructor-arg> 요소의 name 속성으로 이 요소가 적용될 생성자 인자의 이름을 지정한다.

```
1763
1764      <bean id="student2" class="com.example.Student">
1765        <constructor-arg name="name" value="설운도" />
1766        <constructor-arg name="age" value="16" />
1767        <constructor-arg name="grade" value="3" />
1768        <constructor-arg name="classNum" value="7" />
1769      </bean>
1770
```

4)이 구성은 해당 class를 compile할 때 debug flag를 활성화해야 제대로 작동한다.
5)Debug flag를 활성화하면 생성된 .class file에 생성자 인자 이름이 유지된다.
6)만일 debug flag를 활성화하지 않고 compile하면 compile 중에 생성자 이름이 손실되므로 Spring이 <constructor-arg> 요소의 name 속성에 지정된 생성자 인자 이름에 해당하는 생성자 인자를 찾을 수 없다.
7)만일 debug flag를 활성화하고 class를 compile하고 싶지 않은 경우, 다음 예제처럼 @ConstructorProperties annotation(Java SE 6에서 추가됨)을 사용해서 생성자 인자 이름을 명시적으로 지정하면 된다.

```
1771
1772      package com.example;
1773
1774      public class Student {
1775        private String name;
1776        private int age;
1777        private int grade;
1778        private int classNum;
1779
1780        @ConstructorProperties({"name", "age", "grade", "classNum"})
1781        public Student(String name, int age, int grade, int classNum) {
1782          this.name = name;
1783          this.age = age;
1784          this.grade = grade;
1785          this.classNum = classNum;
1786        }
1787
```

8)이 예제에서는 @ConstructorProperties annotation에 생성자 인자의 이름이 bean class의 생성자에 나오는 순서대로 지정돼 있다.
9)이름을 지정할 때는 <constructor-arg> 요소의 생성자 인자 이름과 완전히 동일하게 지정해야 한다.

```
1788
1789
```

19. @ConstructorProperties annotation과 bean 정의 상속

1)부모 bean 정의에 해당하는 class의 생성자에 @ConstructorProperties annotation을 지정한 경우 자식 bean 정의에 해당하는 bean class에도 @ConstructorProperties annotation을 지정해야 한다.

```
1790
1791      <bean id="human" class="com.example.Human">
1792        <constructor-arg name="name" value="설운도" />
1793      </bean>
1794
1795      <bean id="student" class="com.example.Student" parent="human">
1796        <constructor-arg name="age" value="16" />
```

```
        <constructor-arg name="grade" value="3" />
        <constructor-arg name="classNum" value="7" />
      </bean>
```

2)이 예제에서 human bean 정의는 추상이 아니므로(만일 추상이었다면 abstract="true"라고 선언해야 함), Spring container는 이 bean의 instance를 생성한다.
3)human bean의 <constructor-arg> 구성은 자식 bean 정의인 student으로 상속된다.
4)자식 생성자에 @ConstructorProperties annotation을 지정하지 않으면 Spring container가 상속된 <constructor-arg> 요소와 자식 class의 생성자에 지정된 생성자 인자를 연결할 수 없다.
5)하지만, static 또는 instance factory method의 인자를 이름으로 전달하는 데는 @ConstructorProperties annotation을 사용할 수 없다.

20. @ConstructorProperties annotation과 factory method
1)static factory method를 이용한 bean instance화하기
-만일 Calendar class가 instance를 생성하기 위해 Calendar.getInstance()를 사용하는 것처럼 static factory method를 이용한 bean instance화에 대해 살펴보자.
-이럴 경우에 Spring 에서는 아래처럼 bean 정의를 해야 한다.

```
public class CalendarFactory{
  private CalendarFactory(){}

  public static Calendar getInstance(String country){
    Calendar calendar = null;
    ...

    return calendar;
  }
}

<bean id="myCalendar" class="java.util.Calendar" factory-method="getInstance">
  <constructor-arg index="0" value="kor" />
</bean>
```

2)만일 이때, <constructor-arg>요소의 name 속성을 지정하고 factory method에 @ConstructorProperties annotation을 지정하면 static 및 instance factory method에 인자를 이름으로 전달할 수 있지 않을까 생각해본다.
3)하지만, @ConstructorProperties annotation은 생성자에만 지정할 수 있으며, method에는 지정할 수 없다.
4)즉, static 또는 instance factory method에 인자를 이름으로 전달하려면 debug flag를 활성화하고 class를 compile하는 것이 유일한 방법이다.
5)Debug flag를 활성화하고 class를 compile하면, .class file의 크기는 커지지만 application의 실행 성능에는 영향을 주지 않으며, class를 loading하는 시간만 약간 늘어난다.

※Eclipse IDE에서 Debug flag 활성화/비활성하는 방법
1. Window > Preferences에서 Java > Compiler option을 선택
2. [Classfile Generation]이라는 절이 표시된다.
3. 이 절에서 [Add variable attributes to generated class files (used by the debugger] 확인란을 선택하면 debug flag가 활성화되며, 이 확인란을 선택 취소하면 debug flag가 비활성화된다.

21. DI의 장점
1)Javafile의 수정 없이 Spring 설정 file만을 수정하여 부품들을 생성/조립할 수 있다.

2)com.example.Car.java Interface

```
1851    package com.example;
1852
1853    public interface Car {
1854      void drive();
1855    }
1856
1857  3)com.example.Sonata.java
1858    package com.example;
1859
1860    public class Sonata implements Car {
1861
1862      @Override
1863      public void drive() {
1864        System.out.println("Drive a Sonata");
1865      }
1866    }
1867
1868  4)com.example.Carnival.javaa
1869    package com.example;
1870
1871    public class Carnival implements Car {
1872
1873      @Override
1874      public void drive() {
1875        System.out.println("Drive a Carnival");
1876      }
1877    }
1878
1879  5)com.example.HybridCar.java
1880    package com.example;
1881
1882    public class HybridCar extends Sonata implements Car {
1883      @Override
1884      public void drive(){
1885        System.out.println("Drive a HybridCar with Sonata");
1886      }
1887    }
1888
1889  6)CarContext.xml
1890    <?xml version="1.0" encoding="UTF-8"?>
1891    <beans xmlns="http://www.springframework.org/schema/beans"
1892      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1893      xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
1894
1895      <!-- <bean id="car" class="com.example.Sonata" /> -->
1896      <!-- <bean id="car" class="com.example.Carnival" /> -->
1897      <bean id="car" class="com.example.HybridCar" />
1898      //CarMainClass를 변경하지 않고, CarContext.xml만 변경해도 여러 class를 이용할 수 있다.
1899    </beans>
1900
1901  7)com.example.CarMainClass.java
1902    package com.example;
1903
```

```
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 file 여러개 사용하기

1)In Package Explorer > right-click > New > Java Project
-Project Name : SpringDemo2

2)src > right-click > New > Package
-Package name : com.example

3)com.example.Student.java

```
package com.example;

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
```

```
1958        public String toString() {
1959          return String.format("Student [name=%s, age=%s, hobbys=%s, height=%s,
              weight=%s]", name, age, hobbys, height,
1960              weight);
1961        }
1962      }
1963
1964    4)com.example.StudentInfo.java
1965      package com.example;
1966      public class StudentInfo {
1967        private Student student;
1968
1969        public Student getStudent() {
1970          return student;
1971        }
1972
1973        public void setStudent(Student student) {
1974          this.student = student;
1975        }
1976      }
1977
1978    5)com.example.Product.java
1979      package com.example;
1980      public class Product {
1981        private String pName;
1982        private int pPrice;
1983        private String maker;
1984        private String color;
1985        public Product(String pName, int pPrice) {
1986          this.pName = pName;
1987          this.pPrice = pPrice;
1988        }
1989        public void setpName(String pName) {
1990          this.pName = pName;
1991        }
1992        public void setpPrice(int pPrice) {
1993          this.pPrice = pPrice;
1994        }
1995        public void setMaker(String maker) {
1996          this.maker = maker;
1997        }
1998        public void setColor(String color) {
1999          this.color = color;
2000        }
2001        @Override
2002        public String toString() {
2003          return String.format("Product [pName=%s, pPrice=%s, maker=%s, color=%s]",
              pName, pPrice, maker, color);
2004        }
2005      }
2006
2007    6)Java Project를 Spring Project로 변환
2008      -SpringDemo2 Project > right-click > Configuration > Convert to Maven Project
2009        --Project : /SpringDemo2
```

```
2010        --Group Id : SpringDemo2
2011        --Artifact Id : SpringDemo2
2012        --version : 0.0.1-SNAPSHOT
2013        --Packaging : jar
2014        --Finish
2015
2016      -SpringDemo2 Project > right-click > Spring > Add Spring Project Nature
2017
2018      -pom.xml file에 Spring Context Dependency 추가하기
2019       <version>0.0.1-SNAPSHOT</version>
2020        <dependencies>
2021          <dependency>
2022            <groupId>org.springframework</groupId>
2023            <artifactId>spring-context</artifactId>
2024            <version>4.3.24.RELEASE</version>
2025          </dependency>
2026        </dependencies>
2027
2028      -pom.xml > right-click > Run As > Maven install
2029      [INFO] BUILD SUCCESS 확인
2030
2031    7)SpringDemo2/resources folder 생성
2032      -SpringDemo2 project > right-click > Build Path > Configure Build Path
2033      -Source Tab > Add Folder
2034      -SpringDemo2 click
2035      -Create New Folder > Folder name : resources > Finish > OK
2036      -SpringDemo2/resources(new) 확인
2037      -Apply and Close
2038
2039    8)Bean Configuration XML 작성
2040      -SpringDemo2/resources > right-click > New > Other > Spring > Spring Bean Configuration
          File
2041      -File name : applicationContext.xml > Finish
2042
2043    9)applicationContext.xml
2044      <?xml version="1.0" encoding="UTF-8"?>
2045      <beans xmlns="http://www.springframework.org/schema/beans"
2046        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2047        xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
2048
2049        <bean id="student1" class="com.example.Student">
2050          <constructor-arg value="한지민" />
2051          <constructor-arg value="25" />
2052          <constructor-arg>
2053            <list>
2054              <value>독서</value>
2055              <value>영화감상</value>
2056              <value>요리</value>
2057            </list>
2058          </constructor-arg>
2059          <property name="height" value="165" />
2060          <property name="weight">
2061            <value>45</value>
```

```
2062          </property>
2063        </bean>
2064
2065        <bean id="studentInfo1" class="com.example.StudentInfo">
2066          <property name="student">
2067            <ref bean="student1" />
2068          </property>
2069        </bean>
2070      </beans>
2071
2072    10)/resources/applicationContext2.xml
2073      -또 하나의 file을 생성한다.
2074      -위의 applicationContext.xml을 복사하여 붙여 넣기 한다.
2075      -Names tab을 선택하여 c, p를 선택한다.
2076      <?xml version="1.0" encoding="UTF-8"?>
2077      <beans xmlns="http://www.springframework.org/schema/beans"
2078        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2079        xmlns:c="http://www.springframework.org/schema/c"
2080        xmlns:p="http://www.springframework.org/schema/p"
2081        xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
2082
2083        <bean id="student3" class="com.example.Student">
2084          <constructor-arg value="설운도" />
2085          <constructor-arg value="50" />
2086          <constructor-arg>
2087            <list>
2088              <value>노래부르기</value>
2089              <value>게임</value>
2090            </list>
2091          </constructor-arg>
2092          <property name="height" value="175" />
2093          <property name="weight">
2094            <value>75</value>
2095          </property>
2096        </bean>
2097
2098        <bean id="product" class="com.example.Product" c:pName="Computer"
          c:pPrice="2000000" p:maker="Samsung">
2099          <property name="color" value="Yellow" />
2100        </bean>
2101      </beans>
2102
2103    11)com.example.MainClass
2104      package com.example;
2105
2106      import org.springframework.context.support.AbstractApplicationContext;
2107      import org.springframework.context.support.GenericXmlApplicationContext;
2108
2109      public class MainClass {
2110        public static void main(String[] args) {
2111          String configFile = "classpath:applicationContext.xml";
2112          String configFile1 = "classpath:applicationContext2.xml";
2113          AbstractApplicationContext context = new GenericXmlApplicationContext(configFile,
```

```
            configFile1);
2114        Student student1 = context.getBean("student1", Student.class);
2115        System.out.println(student1);
2116
2117        StudentInfo studentInfo = context.getBean("studentInfo1", StudentInfo.class);
2118        Student student2 = studentInfo.getStudent();
2119        System.out.println(student2);
2120        if(student1.equals(student2)) System.out.println("Equals");
2121        else System.out.println("Different");
2122
2123        Student student3 = context.getBean("student3", Student.class);
2124        System.out.println(student3);
2125
2126        if(student1.equals(student3)) System.out.println("Equals");
2127        else System.out.println("Different");
2128
2129        Product product = context.getBean("product", Product.class);
2130        System.out.println(product);
2131        context.close();
2132      }
2133    }
2134
2135  12)결과
2136    Student [name=한지민, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
2137    Student [name=한지민, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
2138    Equals
2139    Student [name=설운도, age=50, hobbys=[노래부르기, 게임], height=175.0,weight=75.0]
2140    Different
2141    Product [pName=Computer, pPrice=2000000, maker=Samsung, color=Yellow]
2142
2143
2144 23. p 및 c namespace를 이용해 간결하게 bean 정의 작성하기
2145   1)Spring은 각각 bean 속성과 생성자 인자의 값을 지정할 수 있는 p 및 c namespace를 제공함으로써 application
       context xml file에서 bean 정의를 간결하게 작성하도록 돕는다.
2146   2)p 및 c namespace는 각각 <property> 및 <constructor-arg> 요소 대신 사용할 수 있다.
2147   3)p namespace
2148     -p namespace를 사용해서 bean 속성을 설정하려면 bean 속성을 <bean> 요소의 속성으로 지정하고 p
         namespace 안에서 각 bean 속성을 지정한다.
2149       <bean id="student2" class="com.example.Student"
2150         p:name="홍길동" p:age="24"
2151         p:height="179.4" p:weight="68.4" />
2152     -bean 속성이 bean 참조가 아닌 경우 다음 구분을 사용해 지정한다.
2153
2154       p:<속성이름>="<속성 값>"
2155
2156     -bean 속성이 bean 참조인 경우 다음 구분을 사용해 지정한다.
2157
2158       p:<속성이름>-ref="<bean 참조>"
2159
2160   4)c namespace
2161     -c namespace를 사용해 생성자 인자의 값을 제공하려면 생성자 인자를 <bean> 요소의 속성으로 지정하고 c
         namespace 안에서 각 생성자 인자를 지정한다.
2162
2163       public class Student{
```

```
        ...
        @ConstructorProperties({"name", "age", "height", "weight" })
        public Student(String name, int age, double height, double weight){
          ...
        }
      }

      <bean id="student2" class="com.example.Student"
        c:name="한지민" c:age="24" c:height="158.3" c:weight="52.4" />
```

-생성자 인자가 bean 참조가 아닌 경우 가음 구문을 사용해 지정한다.

```
      c:<생성자 인자 이름>="<생성자 인자 값>"
```

-생성자 인자가 bean 참조인 경우 다음 구문을 사용해 지정한다.

```
      c:<생성자 인자 이름>-ref="<bean 참조>"
```

-Debug flag를 활성화하고 class를 compile하면 생성된 .class file에 생성자 인자 이름이 보존된다.
-만일 debug flag를 활성화하지 않고 class를 compile하면 예제는 작동되지 않는다.
-이 경우에는 다음과 같이 index를 사용해서 생성자 인자의 값을 지정하면 된다.

```
      <bean id="student2" class="com.example.Student"
        c:_0="한지민" c:_1="24" c:_2="158.3" c:_3="52.4" />
```

-XML에서는 속성 이름을 숫자 값으로 시작할 수 없기 때문에 생성자 인자의 index 앞에 밑줄을 붙인다.
-생성자 인자가 다른 bean의 참조인 경우 생성자 인자의 index 뒤에 -ref를 붙여야 한다.

```
      c:_0-ref
```

24. Java Annotation을 이용한 DI 설정하기

1)@Configuration
public class ApplicationConfig{} //@Configuration : 이 class는 Spring 설정에 사용되는 class 입니다'라고 명시해 주는 annotation

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

3)In Package Explorer > right-click > New > Java Project
-Project Name : SpringDemo3

4)src > right-click > New > Package
-Package name : com.example

5)com.example.Student.java

```
package com.example;

import java.util.ArrayList;

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

```
2217        private double height;
2218        private double weight;
2219        public Student(String name, int age, ArrayList<String> hobbys) {
2220          this.name = name;
2221          this.age = age;
2222          this.hobbys = hobbys;
2223        }
2224        public void setName(String name) {
2225          this.name = name;
2226        }
2227        public void setAge(int age) {
2228          this.age = age;
2229        }
2230        public void setHobbys(ArrayList<String> hobbys) {
2231          this.hobbys = hobbys;
2232        }
2233        public void setHeight(double height) {
2234          this.height = height;
2235        }
2236        public void setWeight(double weight) {
2237          this.weight = weight;
2238        }
2239        @Override
2240        public String toString() {
2241          return String.format("Student [name=%s, age=%s, hobbys=%s, height=%s,
           weight=%s]", name, age, hobbys, height,
2242              weight);
2243        }
2244      }
2245
2246    6)Java Project를 Spring Project로 변환
2247      -SpringDemo3 Project > right-click > Configuration > Convert to Maven Project
2248        --Project : /SpringDemo3
2249        --Group Id : SpringDemo3
2250        --Artifact Id : SpringDemo3
2251        --version : 0.0.1-SNAPSHOT
2252        --Packaging : jar
2253        --Finish
2254        --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
2255
2256      -SpringDemo3 Project > right-click > Spring > Add Spring Project Nature
2257        --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
2258
2259      -pom.xml file에 Spring Context Dependency 추가하기
2260        <version>0.0.1-SNAPSHOT</version>
2261        <dependencies>   <--- dependencies element 추가
2262          <dependency>   <---여기에 paste
2263            <groupId>org.springframework</groupId>
2264            <artifactId>spring-context</artifactId>
2265            <version>4.3.24.RELEASE</version>
2266          </dependency>
2267        </dependencies>
2268
2269      -pom.xml > right-click > Run As > Maven install
```

```
2270    [INFO] BUILD SUCCESS 확인
2271
2272  7)com.example.ApplicationConfig.java
2273    import org.springframework.context.annotation.Bean;
2274    import org.springframework.context.annotation.Configuration;
2275
2276    @Configuration
2277    public class ApplicationConfig {
2278
2279      @Bean
2280      public Student student1(){
2281        ArrayList<String> hobbys = new ArrayList<String>();
2282        hobbys.add("독서");
2283        hobbys.add("영화감상");
2284        hobbys.add("요리");
2285
2286        Student student = new Student("한지민", 25, hobbys);
2287        student.setHeight(165);
2288        student.setWeight(45);
2289
2290        return student;
2291      }
2292
2293      @Bean
2294      public Student student2(){
2295        ArrayList<String> hobbys = new ArrayList<String>();
2296        hobbys.add("노래부르기");
2297        hobbys.add("게임");
2298        Student student = new Student("설운도", 50, hobbys);
2299        student.setHeight(175);
2300        student.setWeight(75);
2301
2302        return student;
2303      }
2304    }
2305
2306  8)com.example.MainClass.java
2307    package com.example;
2308
2309    import org.springframework.context.annotation.AnnotationConfigApplicationContext;
2310
2311    public class MainClass {
2312      public static void main(String[] args) {
2313        AnnotationConfigApplicationContext context = new
          AnnotationConfigApplicationContext(ApplicationConfig.class);
2314        Student student1 = context.getBean("student1", Student.class);
2315        System.out.println(student1);
2316
2317        Student student2 = context.getBean("student2", Student.class);
2318        System.out.println(student2);
2319
2320        context.close();
2321      }
2322    }
```

9)결과
Student [name=한지민, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
Student [name=설운도, age=50, hobbys=[노래부르기, 게임], height=175.0,weight=75.0]

25. Java Annotation과 XML 을 이용한 DI 설정 방법 : XML file에 Java file을 포함시켜 사용하는 방법
1)In Package Explorer > right-click > New > Java Project
-Project Name : SpringDemo4

2)src > right-click > New > Package
-Package name : com.example

3)com.example.Student.java

```
package com.example;

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)Java Project를 Spring Project로 변환
-SpringDemo4 Project > right-click > Configuration > Convert to Maven Project

```
2376        --Project : /SpringDemo4
2377        --Group Id : SpringDemo4
2378        --Artifact Id : SpringDemo4
2379        --version : 0.0.1-SNAPSHOT
2380        --Packaging : jar
2381        --Finish
2382        --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
2383
2384      -SpringDemo4 Project > right-click > Spring > Add Spring Project Nature
2385        --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
2386
2387      -pom.xml file에 Spring Context Dependency 추가하기
2388       <version>0.0.1-SNAPSHOT</version>
2389        <dependencies>   <--- dependencies element 추가
2390          <dependency>   <---여기에 paste
2391            <groupId>org.springframework</groupId>
2392            <artifactId>spring-context</artifactId>
2393            <version>4.3.24.RELEASE</version>
2394          </dependency>
2395        </dependencies>
2396
2397      -pom.xml > right-click > Run As > Maven install
2398      [INFO] BUILD SUCCESS 확인
2399
2400
2401    5)com.example.ApplicationConfig.java
2402      package com.example;
2403
2404      import java.util.ArrayList;
2405
2406      import org.springframework.context.annotation.Bean;
2407      import org.springframework.context.annotation.Configuration;
2408
2409      @Configuration
2410      public class ApplicationConfig {
2411        @Bean
2412        public Student student1(){
2413          ArrayList<String> hobbys = new ArrayList<String>();
2414          hobbys.add("독서");
2415          hobbys.add("영화감상");
2416          hobbys.add("요리");
2417
2418          Student student = new Student("한지민", 25, hobbys);
2419          student.setHeight(165);
2420          student.setWeight(45);
2421
2422          return student;
2423        }
2424      }
2425
2426    6)SpringDemo4/resources folder 생성
2427      -SpringDemo4 project > right-click > Build Path > Configure Build Path
2428      -Source Tab > Add Folder
2429      -SpringDemo4 click
```

```
   -Create New Folder > Folder name : resources > Finish > OK
   -SpringDemo4/resources(new) 확인
   -Apply and Close

7)Bean Configuration XML 작성
   -SpringDemo4/resources > right-click > New > Other > Spring > Spring Bean Configuration File
   -File name : applicationContext.xml > Finish

8)/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="com.example.ApplicationConfig" />
     <bean id="student3" class="com.example.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>

9)com.example.MainClass.java
   package com.example;

   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);
     }
   }
```

10)result
Student [name=한지민, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
Student [name=설운도, age=50, hobbys=[노래부르기, 게임], height=175.0,weight=75.0]

26. Java Annotation과 XML 을 이용한 DI 설정 방법 : Java file에 XML file을 포함시켜 사용하는 방법
1)In Package Explorer > right-click > New > Java Projectn
-Project Name : SpringDemo5

2)src > right-click > New > Package
-Package name : com.example

3)com.example.Student.java

```
package com.example;

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)Java Project를 Spring Project로 변환
-SpringDemo5 Project > right-click > Configuration > Convert to Maven Project
--Project : /SpringDemo5

```
2534        --Group Id : SpringDemo5
2535        --Artifact Id : SpringDemo5
2536        --version : 0.0.1-SNAPSHOT
2537        --Packaging : jar
2538        --Finish
2539        --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
2540
2541      -SpringDemo5 Project > right-click > Spring > Add Spring Project Nature
2542        --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
2543
2544      -pom.xml file에 Spring Context Dependency 추가하기
2545       <version>0.0.1-SNAPSHOT</version>
2546        <dependencies>   <--- dependencies element 추가
2547          <dependency>   <---여기에 paste
2548            <groupId>org.springframework</groupId>
2549            <artifactId>spring-context</artifactId>
2550            <version>4.3.24.RELEASE</version>
2551          </dependency>
2552        </dependencies>
2553
2554      -pom.xml > right-click > Run As > Maven install
2555      [INFO] BUILD SUCCESS 확인
2556
2557    5)SpringDemo5/resources folder 생성
2558      -SpringDemo5 project > right-click > Build Path > Configure Build Path
2559      -Source Tab > Add Folder
2560      -SpringDemo5 click
2561      -Create New Folder > Folder name : resources > Finish > OK
2562      -SpringDemo5/resources(new) 확인
2563      -Apply and Close
2564
2565    6)Bean Configuration XML 작성
2566      -SpringDemo4/resources > right-click > New > Other > Spring > Spring Bean Configuration
          File
2567      -File name : applicationContext.xml > Finish
2568
2569    7)/resources/applicationContext.xml
2570      <?xml version="1.0" encoding="UTF-8"?>
2571      <beans xmlns="http://www.springframework.org/schema/beans"
2572        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2573        xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
2574
2575        <bean id="student3" class="com.example.Student">
2576          <constructor-arg value="설운도" />
2577          <constructor-arg value="50" />
2578          <constructor-arg>
2579            <list>
2580              <value>노래부르기</value>
2581              <value>게임</value>
2582            </list>
2583          </constructor-arg>
2584          <property name="height" value="175" />
2585          <property name="weight">
```

```
            <value>75</value>
          </property>
        </bean>
      </beans>

  8)com.example.ApplicationConfig.java
      package com.example;

      import java.util.ArrayList;

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

      @Configuration
      @ImportResource("classpath:ApplicationContext.xml")
      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;
        }
      }

  9)com.example.MainClass.java
      package com.example;

      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();
        }
      }

  10)result
      Student [name=한지민, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
```

Student [name=설운도, age=50, hobbys=[노래부르기, 게임], height=175.0,weight=75.0]

27. 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 file을 관리하기 번거로울 수 있다.
-여러 개발자가 같은 설정file을 공유해서 개발하다보면 설정file을 동시에 수정하다가 충돌이 일어나는 경우도 적지 않다.
-DI에 필요한 적절한 setter method 또는 constructor가 code 내에 반드시 존재해야 한다.
-개발 중에는 annotation 설정방법을 사용했지만, 운영중에는 관리의 편의성을 위해 XML 설정으로 변경하는 전략을 쓸 수도 있다.

2)전략 2 - XML과 Bean Scanning의 혼용
-Bean으로 사용될 class에 특별한 annotation을 부여해주면 이런 class를 자동으로 찾아서 Bean으로 등록한다.
-특정 annotation이 붙은 class를 자동으로 찾아서 Bean으로 등록해 주는 방식을 Bean Scanning을 통한 자동인식 Bean 등록기능이라고 한다.
-annotation을 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.

3)주의 사항
-library형태로 제공되는 Class는 반드시 XML 설정을 통해서만 사용할 수 있다.
-예를 들면, Apache에서 제공하는 BasicDataSource class를 사용하여 DB 연동을 처리한다면 commons-dbcp-1.4.jar file에 있는 BasicDataSource Class에 관련된 Annotation을 추가할 수 없다.

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close">
  <property name="driverClassName" value="org.h2.Driver" />
  <property name="url"  value="jdbc:h2:tcp://localhost/~/test" />
  <property name="username" value="sa" />
  <property name="password" value="" />
</bean>
```

28. Bean등록 및 의존관계 설정하는 Annotation
1)Bean 등록 Annotation
-@Component
--Component를 나타내는 일반적인 스테레오 타입으로 <bean> 태그와 동일한 역할을 함
--Spring component class를 지정하는 형식 단계의 annotation
--이 annotation보다는 아래에 소개된 각각의 annotation을 사용하는 것이 좋다.
-@Repository : Persistence 레이어, 영속성을 가지는 속성(file, 데이터베이스)을 가진 class
-@Service : service 레이어, 비즈니스 로직을 가진 class
-@Controller : 프리젠테이션 레이어, 웹 어플리케이션에서 웹 요청과 응답을 처리하는 class
-@Repository, @Service, @Controller는 더 특정한 유즈케이스에 대한 @Component의 구체화된 형태이다.

```
@Service(value = "TestService")
public class TestServiceImpl implements Service {...}
```

-위의 코드처럼 @Service annotation은 bean을 Spring container에 component로 등록하는 데 사용할 이름을 지정하는 value 속성을 받는다.
-즉, TestServiceImpl class를 TestService라는 이름의 bean으로 Spring container에 등록했다.
-value 속성은 <bean> 요소의 id 속성과 동일한 역할을 한다.

-@Component, @Controller, @Repository annotation에서도 value 속성으로 component의 이름을 지정할 수 있지만, value 속성 없이도 Spring component의 이름을 지정할 수 있다.
-즉, @Service(value = "TestService") 및 @Service("TestService")는 동일하다.
-Component의 이름을 지정하지 않으면 Spring은 component의 이름이 component class와 같다고 가정한다.
-다만 component의 이름은 소문자로 시작한다.
-Component에 사용자 지정 이름을 지정하면 '이름을 기준으로' 자동 연결을 수행할 때 특히 유용하다.
-Spring의 class 경로 검사 기능을 활성화한 경우 @Component, @Controller, @Service 또는 @Repository annotation으로 지정한 bean class가 자동으로 Spring container에 등록된다.
-Class 경로 검사 기능은 Spring context schema의 <component-scn> 요소를 사용해 활성화할 수 있다.

```
<context:component-scan base-package="com.example" />
```

2)Bean 의존관계 주입 Annotation
-@Autowired, @Resource annotation은 의존하는 객체를 자동으로 주입해 주는 annotation이다.
-@Autowired는 타입으로, @Resource는 이름으로 연결한다는 점이 다르다.
-@Autowired
--정밀한 의존관계 주입(Depencdency Injection)이 필요한 경우에 유용하다.
--property, setter method, 생성자, 일반 method에 적용 가능하다.
--의존하는 객체를 주입할 때 주로 Type을 이용하게 된다.
--<property>, <constructor-arg>태그와 동일한 역할을 한다.
--아래 Component Scan을 지원하는 태그부분에서 언급했듯이 Bean 설정file에서 <context:component-scan>를 설정해 주면 DI container는 해당 패키지에서 @Autowired가 붙은 인스턴스 변수의 형에 대입할 수 있는 class를 @Component가 붙은 class 중에서 찾아내 그 인스턴스를 injection해 준다.
--그리고 인스턴스 변수로 인젝션은 access modifier가 private이라도 injection할 수 있다.
--만일 @Autowired가 붙은 객체가 메모리에 없다면 Container는 NoSuchBeanDefinitionException을 발생시킨다.
--이 Annotation은 settermethod를 필요로 하지 않게 한다.

```
@Autowired
private String name;
```

--이 annotatio을 사용할 때 형식과 일치하는 bean이 발견되지 않으면 예외가 발생한다.
--이 annotation의 required 속성은 자동 연결 의존성이 필수 인지 또는 선택사항인지 지정한다.
--만일 required 속성을 false로 지정하면 의존성 자동 연결을 선택 사항으로 취급한다.
--즉, 필수 형식과 일치하는 bean이 발견되지 않아도 예외가 발생하지 않는다는 것을 의미한다.
--기본적으로 required 속성은 true이므로 Spring container가 의존성을 충족할 수 있어야 한다.

-@Resource
--Spring은 JSR 250의 이 annotation을 통해 field와 method를 이름을 기준으로 자동 연결할 때 사용된다.
--프로퍼티, setter method에 적용 가능하다.
--생성자 인자를 자동 연결하는 데는 사용할 수 없다.
--의존하는 객체를 주입할 때 주로 name을 이용하게 된다.
--이름을 기준으로 의존성을 자동 연결할 때는 @Autowired 및 @Qualifier annotation보다는 @Resource annotation을 사용하는 것이 좋다.
--왜냐하면, Spring은 @Autowired-@Qualifier 조합을 사용하면 먼저 field의 형식(또는 method 인자나 생성자 인자의 형식)을 기준으로 bean을 찾은 다음, @Qualifier annotation으로 지정된 bean이름으로 범위를 좁히는 과정을 거친다.
--반면, @Resource annotation을 사용하면 @Resource annotation으로 지정된 bean 이름으로 곧바로 고유한 bean을 찾는다.
--즉 @Resource annotation을 사용하면 자동 연결할 field(또는 setter method 인자)의 형식은 관계가 없다.

```
@Resource(name="stringPrinter")
```

```
2728        private Printer printer;
2729
2730      -@Value
2731        --단순한 값을 주입할 때 사용되는 annotation이다.
2732        --@Value("Spring")은 <property ... value="Spring" />와 동일한 역할을 한다.
2733        --field, method, method 매개변수 및 생성자 인자 단계에서 사용 가능하다.
2734
2735          @Value("#{configuration.environment}")
2736          private String environment;
2737
2738          @Value("#{configuration.getCountry()}")
2739          private String country;
2740
2741        --SpEL(Spring Expression Language) 식을 @Value annotation의 값으로 사용할 수도 있다.
2742        --SpEL은 실행 시 객체를 조회 및 처리할 수 있는 식 언어다.
2743        --@Value annotation으로 지정한 SpEL 식은 BeanPostProcessor가 처리한다.
2744        --SpEL 식은 <bean 이름>.<field 또는 속성 또는 method> 형식을 사용해 값을 얻을 수 있다.
2745
2746      -@Qualifier
2747        --@Autowired annotation과 같이 사용되어 진다.
2748        --field, method parameter 및 생성자 인자 단계에 사용가능하다.
2749        --@Autowired는 타입으로 찾아서 주입하므로, 동일한 타입의 Bean객체가 여러 개 존재할 때 특정 Bean을 찾기
            위해서는 @Qualifier를 같이 사용해야 한다.
2750        --의존성 주입 대상이 되는 객체가 두 개 이상일 때 Container는 어떤 객체를 할당할지 스스로 판단할 수 없어서 에
            러를 발생한다.
2751        --NoUniqueBeanDefinitionException이 발생한다.
2752        --아래의 코드처럼, Spring은 먼저 @Autowired annotation이 지정된 field, 생성자, method에서 type을
            기준으로 자동 연결 후보를 찾을 다음, @Qualifier annotation으로 지정된 bean 이름을 사용해서 자동 연결 후보
            목록에서 고유한 bean을 찾는다.
2753
2754        @Autowired
2755        @Qualifier("stringPrinter")
2756        private Printer printer;
2757
2758      -@Inject와 @Named
2759        --@Autowired와 동일한 기능을 제공
2760        --method, 생성자, field 단계에서 사용 가능.
2761        --@Autowired와 달리 required 속성이 없으므로 의존성 자동 연결이 필수인지 선택인지를 지정할 수 없다.
2762        --JSR 330에서 제안
2763        --JSR 330(Java 의존성 주입)은 Java platform을 위한 의존성 주입 annotation을 표준화해서 Spring의
            @Autowired 및 @Qualifier annotation과 비슷한 @Inject 및 @Named annotation을 정의하고 있다.
2764        --@Named annotation은 타입 단계에 지정된 경우 Spring @Component annotation과 비슷하게 작동하
            며, method 매개변수나 생성자 인자 단계에 지정된 경우에는 Spring의 @Qualifier annotation과 비슷하게 작
            동한다.
2765        --만일 @Named annotation을 class에 지정한 경우 Spring context schema의 <component-scan>
            요소는 이 class를 @Component annotation이 지정된 component class와 같이 취급한다.
2766        --@Named 및 @Inject annotation을 사용하려면 JSR 330 JAR file을 project에 포함시켜야 한다.
2767        --pom.xml에 다음과 같이 <dependency> 요소를 사용해서 JSR 330 JAR file을 포함시킨다.
2768
2769          <dependency>
2770            <groupId>javax.injdect</groupId>
2771            <artifactId>javax.inject</artifactId.
2772            <version>1</version>
2773          </dependency>
```

-@Scope
--Spring component의 범위(prototype or singleton)를 지정
--기본적으로 singleton 범위를 가지며, prototype 범위로 지정하기 위해 사용
--<bean> 요소의 scope 속성과 같은 역할
--value 속성 값으로 SCOPE_SINGLETON과 SCOPE_PROTOTYPE 상수를 지정할 수 있다.

```
import org.springframework.beans.factory.config.ConfigurableBeanFactory;
import org.springframework.context.annotation.Scope;

@Scope(value=ConfigurableBeanFactory.SCOPE_PROTOTYPE)
public class Test {...}
```

-@Lazy
--기본적으로 singleton 범위 Spring component는 사전 초기화된다.
--즉, Spring container가 생성될 때 함께 instance화 된다.
--Singleton 범위 구성 요소를 지연 생성하려면 singleton 범위 component의 component class에 @Lazy annotation을 지정하면 된다.
--<bean> 요소의 lazy-init 속성과 같은 역할을 한다.

```
@Lazy(value=true)
@Component
public class Sample {...}
```

-@DependsOn
--암시적 bean 의존성을 지정하는데 사용

```
@DependsOn(value = {"beanA", "beanB"})
@Component
public class Sample {...}
```

--위 코드는 Sample class의 @DependsOn annotation은 Sample class의 instance를 생성하기 전에 beanA 및 beanB bean을 먼저 생성하도록 Spring container에 지시한다.
--<bean> 요소의 depends-on 속성가 같은 역할을 한다.

-@Primary
--의존성을 자동 연결을 후보가 여럿인 경우 특정한 bean을 자동 연결의 기본 후보로 지정할 수 있다.
--<bean> 요소의 primary 속성과 같은 역할을 한다.

```
@Primary
@Component
public class Sample {...}
```

3)객체의 유효성 검사를 위한 annotation
-JSR 303(bean 유효성 검사)의 annotation을 사용해서 JavaBeans component에 제약 조건을 지정할 수 있다.
-bean 속성에 JSR 303 annotation을 지정하면 Spring이 bean 유효성 검사를 수행하고 결과를 제공한다.
-@NotNull
--Annotation을 지정한 field가 null일 수 없다.

```
@NotNull
private long id;
```

-@Min

2825 --Annotation을 지정한 field가 지정된 값 이상이어야 한다.
2826
2827 @Min(1000)
2828 @Max(500000)
2829 private float depositAmount;
2830
2831 -@Max
2832 --Annotation을 지정한 field가 지정된 값 이하여야 한다.
2833
2834 -@NotBlank
2835 --Annotation을 지정한 field가 null이거나 비어 있을 수 없다.
2836
2837 @NotBlank
2838 @Size(min=5, max=100)
2839 private String email;
2840
2841 -@Size
2842 --Annotation을 지정한 field의 크기가 지정된 min 및 max 특성 사이여야 한다.
2843
2844
2845 4)Component Scan을 지원하는 태그
2846 -<context:component-scan> 태그
2847 --@Component를 통해 자동으로 Bean을 등록하고, @Autowired로 의존관계를 주입받는 annotation을 class에서 선언하여 사용했을 경우에는 해당 class가 위치한 특정 패키지를 Scan하기 위한 설정을 XML에 해주어야 한다.
2848
2849 <context:component-scan base-package="com.example" />
2850
2851 -Spring container에 자동 등록할 component class를 filtering 하려면 <component-scan> 요소의 resource-pattern 속성을 사용한다.
2852 -resource-pattern 속성의 기본값은 **/*.class이므로 base-package 속성으로 지정한 package 이하의 모든 component class가 자동 등록된다.
2853 -<context:include-filter> 태그와 <context:exclude-filter> 태그를 같이 사용하면 자동 스캔 대상에 포함시킬 class와 포함시키지 않을 class를 구체적으로 명시할 수 있다.
2854
2855 <beans ...>
2856 <context:component-scan base-package="com.example">
2857 <context:include-filter type="annotation"
expression="com.example.annotation.MyAnnotation" />
2858 <context:exclude-filter type="regex" expression=".*Details" />
2859 </context:component-scan>
2860 </beans>
2861
2862 -<exclude-filter> 및 <include-filter> 요소의 type 속성에는 component class를 filtering하는 전략을 지정하며, expression 속성은 해당하는 filter 식을 지정한다.
2863 -위의 코드에서 보면, <include-filter> 요소에서는 MyAnnotation 형식 단계 annotation을 지정한 component class를 Spring container에 자동으로 등록하도록 지정했으며, <exclude-filter> 요소에서는 <component-scan> 요소에서 이름이 Detail로 끝나는 component class를 무시하도록 지정했다.
2864 -다음 표를 참조하자.
2865
2866 type 속성 값 설명
2867 annotation 이 경우 expression 속성에는 component class에 사용해야 하는 annotation의 정규화된 class의 이름을 지정한다. 예를 들어 expression 속성값이 com.example.annotation.MyAnnotation인 경우 MyAnnotation annotation을 지정한 component class가 포함(<include-filter>요소에 사용된 경우)

또는 제외(<exclude-filter>요소에 사용된 경우)된다.
2868 assignable 이 경우 expression 속성에는 component class를 할당할 수 있어야 하는 class 또는 interface의 정규화된 이름을 지정한다.
2869 aspectj 이 경우 expression 속성에는 component class를 filtering 하는데 사용되는 AspectJ 식을 지정한다.
2870 regex 이 경우 expression 속성에는 component class를 이름을 기준으로 filtering 하는데 사용되는 정규식을 지정한다.
2871 custom 이 경우 expression 속성에는 component class를 filtering 하기 위한 org.springframework.core.type.TypeFilter interface의 구현을 지정한다.
2872
2873 5)사용 예
2874 <SpringPrinter.java>
2875
```
package com.example;

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();
  }
}
```
2891
2892
2893 29. Lab
2894 1)In Package Explorer > right-click > New > Java Project
2895 -Project name : DIDemo3
2896
2897 2)src > right-click > New > Package
2898 -Package name : com.example
2899
2900 3)POJO class 작성
2901 -com.example > right-click > New > Class
2902 <Hello.java>
2903
```
package com.example;

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;
```

```
2917          }
2918
2919          public String sayHello(){
2920            return "Hello " + name;
2921          }
2922
2923          public void print(){
2924            this.printer.print(sayHello());
2925          }
2926        }
2927
2928      -com.example > right-click > New > Interface
2929        interface name : Printer
2930
2931      <Printer.java>
2932        package com.example;
2933
2934        public interface Printer{
2935          void print(String message);
2936        }
2937
2938      -com.example > right-click > New > Class
2939        Class Name : StringPrinter
2940
2941      <StringPrinter.java>
2942        package com.example;
2943
2944        public class StringPrinter implements Printer{
2945          private StringBuffer buffer = new StringBuffer();
2946
2947          @Override
2948          public void print(String message){
2949            this.buffer.append(message);
2950          }
2951
2952          public String toString(){
2953            return this.buffer.toString();
2954          }
2955        }
2956
2957      -com.example > right-click > New > Class
2958        Class Name : ConsolePrinter
2959
2960      <ConsolePrinter.java>
2961        package com.example;
2962
2963        public class ConsolePrinter implements Printer{
2964
2965          @Override
2966          public void print(String message){
2967            System.out.println(message);
2968          }
2969        }
2970
```

```
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

  -DIDemo3 Project > right-click > Spring > Add Spring Project Nature

  -pom.xml file에 Spring Context Dependency 추가하기
   <version>0.0.1-SNAPSHOT</version>
    <dependencies>
      <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>4.3.24.RELEASE</version>
      </dependency>
    </dependencies>

  -pom.xml > right-click > Run As > Maven install
  [INFO] BUILD SUCCESS 확인

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="com.example.Hello">
        <property name="name" value="Spring" />
        <property name="printer" ref="printer" />
      </bean>
      <bean id="printer" class="com.example.StringPrinter" />
      <bean id="consolePrinter" class="com.example.ConsolePrinter" />

    </beans>

7)DI Test class 작성
  -/src/com.example > right-click > New > Package
    Package Name : test
  -/src/com.example/test/HelloBeanTest.java
```

```
3024
3025        package com.example.test;
3026
3027        import org.springframework.context.ApplicationContext;
3028        import org.springframework.context.support.GenericXmlApplicationContext;
3029
3030        import com.example.Hello;
3031        import com.example.Printer;
3032
3033        public class HelloBeanTest {
3034          public static void main(String [] args){
3035            //1. IoC Container 생성
3036            ApplicationContext context =
3037                 new GenericXmlApplicationContext("config/beans.xml");
3038
3039            //2. Hello Beans 가져오기
3040            Hello hello = (Hello)context.getBean("hello");
3041            System.out.println(hello.sayHello());
3042            hello.print();
3043
3044            //3. SpringPrinter 가져오기
3045            Printer printer = (Printer)context.getBean("printer");
3046            System.out.println(printer.toString());
3047
3048            Hello hello2 = context.getBean("hello", Hello.class);
3049            hello2.print();
3050
3051            System.out.println(hello == hello2);  //Singleton Pattern
3052          }
3053        }
3054
3055        ------------------------------------------------------------------
3056        Hello Spring
3057        Hello Spring
3058        true
3059
3060    8)jUnit Library 설치
3061      -jUnit 4.12 버전을 pom.xml에 추가
3062
3063        <dependency>
3064            <groupId>junit</groupId>
3065            <artifactId>junit</artifactId>
3066            <version>4.12</version>
3067            <scope>test</scope>
3068        </dependency>
3069
3070      -pom.xml > right-click > Run As > Maven Install
3071
3072    9)jUnit을 사용한 DI test class(HelloBeanJunitTest.java) 작성
3073      -/src/com.example.test/HelloBeanTest.java 복사
3074      -/src/com.example.test/ 붙여넣고 이름변경 -> HelloBeanJunitTest.java
3075
3076        package com.example.test;
3077
```

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

import com.example.Hello;
import com.example.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>
```

```
3132      </dependency>
3133
3134    -pom.xml > right-click > Maven Install
3135
3136    -Spring-Test를 사용할 DI test class-HelloBeanJunitSpringTest.java 작성하기
3137      --/src/com.example.test/HelloBeanJunitTest.java 복사해서
3138      --/src/com.example.test/HelloBeanJunitSpringTest.java 로 붙여넣기
3139
3140        import org.junit.runner.RunWith;
3141        import org.springframework.beans.factory.annotation.Autowired;
3142        import org.springframework.test.context.ContextConfiguration;
3143        import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
3144        ...
3145        @RunWith(SpringJUnit4ClassRunner.class)
3146        @ContextConfiguration(locations="classpath:config/beans.xml")
3147        public class HelloBeanJunitSpringTest {
3148
3149          @Autowired
3150          ApplicationContext context;
3151
3152    -right-click > Run As > Junit Test
3153    -결과 -> Junit View에 초록색 bar
3154
3155  11)src/com.example/StringPrinter.java 수정
3156    package com.example;
3157
3158    import org.springframework.stereotype.Component;
3159
3160    @Component("stringPrinter")
3161    public class StringPrinter implements Printer{
3162      private StringBuffer buffer = new StringBuffer();
3163    ...
3164
3165  12)src/com.example/ConsolePrinter.java 수정
3166
3167    package com.example;
3168
3169    import org.springframework.stereotype.Component;
3170
3171    @Component("consolePrinter")
3172    public class ConsolePrinter implements Printer{
3173    ...
3174
3175  13)/src/com.example/Hello.java 수정
3176    package com.example;
3177
3178    import org.springframework.beans.factory.annotation.Autowired;
3179    import org.springframework.beans.factory.annotation.Qualifier;
3180    import org.springframework.beans.factory.annotation.Value;
3181    import org.springframework.stereotype.Component;
3182
3183    @Component
3184    public class Hello {
3185      @Value("Spring")
```

```
3186        private String name;
3187
3188        @Autowired
3189        @Qualifier("stringPrinter")
3190        private Printer printer;
3191
3192        //setter method가 필요 없음.
3193
3194        public String sayHello(){
3195          return "Hello " + name;
3196        }
3197
3198        public void print(){
3199          this.printer.print(sayHello());
3200        }
3201      }
3202
3203    14)기존의 설정file과 충돌이 발생하기 때문에 /src/config/beans.xml 삭제
3204
3205    15)새로운 설정 file 생성
3206      -/src/config/beans.xml 새로 생성
3207      -/src/config > right-click > New > Spring Bean Configuration File
3208        File name : annos.xml
3209      - Next > context - http://www.springframework.org/schema/context Check
3210      - Check http://www.springframework.org/schema/context/spring-context-4.3.xsd
3211      - Finish
3212
3213      <?xml version="1.0" encoding="UTF-8"?>
3214      <beans xmlns="http://www.springframework.org/schema/beans"
3215        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3216        xmlns:context="http://www.springframework.org/schema/context"
3217        xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd
3218          http://www.springframework.org/schema/context
              http://www.springframework.org/schema/context/spring-context-4.3.xsd">
3219
3220        <context:component-scan base-package="com.example" />
3221      </beans>
3222
3223    16)/src/com.example.test/HelloBeanJunitSpringTest.java 수정하기
3224        package com.example.test;
3225
3226        import static org.junit.Assert.assertEquals;
3227
3228        import org.junit.Test;
3229        import org.junit.runner.RunWith;
3230        import org.springframework.beans.factory.annotation.Autowired;
3231        import org.springframework.context.ApplicationContext;
3232        import org.springframework.test.context.ContextConfiguration;
3233        import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
3234
3235        import com.example.Hello;
3236
3237        @RunWith(SpringJUnit4ClassRunner.class)
```

```
3238        @ContextConfiguration(locations="classpath:config/beans.xml")
3239        public class HelloBeanJunitSpringTest {
3240          @Autowired
3241          ApplicationContext context;
3242
3243          @Test
3244          public void test(){
3245            Hello hello = context.getBean("hello", Hello.class);
3246            assertEquals("Hello Spring", hello.sayHello());
3247          }
3248        }
3249
3250      -right-click > Run As > Junit Test
3251      -결과 -> Junit View에 초록색 bar
```