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

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

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

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 아이콘에 Maven의 'M'자가 보임.

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

    -pom.xml 파일에 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
      [INFO] BUILD SUCCESS 확인
```

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

-장점
--코드가 단순해진다.
--컴포넌트 간의 결합도가 제거된다.

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

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

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

-Setter Injection
<beans.xml>

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

<Hello.java>

```
package bean;

import java.util.List;

public class Hello{
  String name;
  Printer printer;

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

-Constructor Injection
<beans.xml>

```
        <bean id="hello" class="bean.Hello">   <--bean은 Srping이 관리해주는 객체라는 뜻
          <constructor-arg index="0" value="Spring" />
          <constructor-arg index="1" ref="printer" />
        </bean>
        <bean id="printer" class="bean.StringPrinter" />
        <bean id="consolePrinter" class="bean.ConsolePrinter" />

      <Hello.java>
        package bean;

        import java.util.List;

        public class Hello{
          String name;
          Printer printer;

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


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

     BeanFactory<interface>
             |
             |
   ApplicationContext<interface>

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

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

XmlWebApplicationContext

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

7. 간단한 DI 프로젝트
  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(){}

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

        public void setPrinter(Printer printer){

```
373           this.printer = printer;
374         }
375
376         public String sayHello(){
377           return "Hello " + name;
378         }
379
380         public void print(){
381           this.printer.print(sayHello());
382         }
383       }
384
385   5)Printer interface의 child class 작성하기
386     -com.example > right-click > New > Class
387       --Class Name : StringPrinter
388       --Interfaces : com.example.Printer
389
390     <StringPrinter.java>
391       package com.example;
392
393       public class StringPrinter implements Printer{
394         private StringBuffer buffer = new StringBuffer();
395
396         @Override
397         public void print(String message){
398           this.buffer.append(message);
399         }
400
401         public String toString(){
402           return this.buffer.toString();
403         }
404       }
405
406     -com.example > right-click > New > Class
407       --Class Name : ConsolePrinter
408       --Interface : com.example.Printer
409
410     <ConsolePrinter.java>
411       package com.example;
412
413       public class ConsolePrinter implements Printer{
414
415         @Override
416         public void print(String message){
417           System.out.println(message);
418         }
419       }
420
421   6)Java Project를 Spring Project로 변환
422     -DIDemo Project > right-click > Configuration > Convert to Maven Project
423       --Project : /DIDemo
424       --Group Id : DIDemo
425       --Artifact Id : DIDemo
426       --version : 0.0.1-SNAPSHOT
```

```
427       --Packaging : jar
428       --Finish
429       --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
430
431     -DIDemo Project > right-click > Spring > Add Spring Project Nature
432       --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
433
434     -pom.xml 파일에 Spring Context Dependency 추가하기
435      <version>0.0.1-SNAPSHOT</version>
436       <dependencies>   <--- dependencies element 추가
437         <dependency>   <---여기에 paste
438           <groupId>org.springframework</groupId>
439           <artifactId>spring-context</artifactId>
440           <version>4.3.24.RELEASE</version>
441         </dependency>
442       </dependencies>
443
444     -pom.xml > right-click > Run As > Maven install
445     [INFO] BUILD SUCCESS 확인
446
447   7)src/config folder 생성
448     -/src > right-click > New > Folder
449       Folder name : config    <--설정 Meta 정보 XML 작성
450
451   8)Bean Configuration XML 작성
452     -/src/config > right-click > New > Other > Spring > Spring Bean Configuration File
453       File name : beans.xml > Next
454         Check [beans - http://www.springframework.org/schema/beans]
455         Check [http://www.springframework.org/schema/beans/spring-beans-4.3.xsd]
456         Finish
457
458       <?xml version="1.0" encoding="UTF-8"?>
459       <beans xmlns="http://www.springframework.org/schema/beans"
460         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
461         xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
462
463         <bean id="hello" class="com.example.Hello">
464           <property name="name" value="Spring" />
465           <property name="printer" ref="printer" />
466         </bean>
467         <bean id="printer" class="com.example.StringPrinter" />
468         <bean id="consolePrinter" class="com.example.ConsolePrinter" />
469
470       </beans>
471
472   9)Beans Graph 사용하기
473     -Windows menu > Show View > Other > Spring > Spring Explorer
474     -In Spring Explorer
475       --DIDemo > Beans > beans.xml > right-click > Open Beans Graphs
476
477   10)DI Test 클래스 작성
478     -/src/com.example > right-click > New > Package
479       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("config/beans.xml");

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

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

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

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

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

8. jUnit의 개요와 특징

1)jUnit의 특징

-TDD의 창시자인 Kent Beck과 디자인 패턴 책의 저자인 Erich Gamma가 작성
-단정(Assert) 메소드로 테스트 케이스의 수행 결과를 판별 --> assertEquals(예상 값, 실제 값)
-jUnit4부터는 테스트를 지원하는 어노테이션 제공, @Test, @Before, @After
-각 @Test 메소드가 호출할 때마다 새로운 인스턴스를 생성하여 독립적인 테스트가 이루어지도록 한다.

2)jUnit

-jUnit Library 설치
--http://mvnrepository.com에 접근
--jUnit으로 검색
--jUnit 4.12 버전을 pom.xml에 추가

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
```

```
            <version>4.12</version>
            <scope>test</scope>
        </dependency>

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

    -jUnit에서 테스트를 지원하는 어노테이션
        --@Test
            ---이것이 선언된 메소드는 테스트를 수행하는 메소드가 된다.
            ---jUnit은 각각의 테스트가 서로 영향을 주지 않고 독립적으로 실행됨을 원칙으로 하므로 @Test 마다 객체를 생성한다.

        --@Ignore
            ---이것이 선언된 메소드는 테스트를 실행하지 않게 한다.

        --@Before
            ---이것이 선언된 메소드는 @Test가 실행되기 전에 반드시 실행된다.
            ---@Test 메소드에서 공통으로 사용하는 코드를 @Before 메소드에 선언하여 사용하면 된다.

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

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

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

    -테스트 결과를 확인하는 단정(Assert) 메소드 종류
        --org.junit.Assert
            +assertArrayEquals(expected, actual)
            +assertEquals(expected, actual)
            +assertNotNull(object)
            +assertSame(expected, actual)
            +assertTrue(object)

    -assertEquals(a, b)
        --객체 a와 b가 일치함을 확인
    -assertArrayEquals(a, b)
        --배열 a, b가 일치함을 확인
    -assertSame(a, b)
        --객체 a, b가 같은 객체임을 확인
        --assertEquals() 메소드는 값이 같은지를 확인하는 것이고, assertSame() 메소든느 두 객체의 레퍼런스가 같은지를 확인한다.(==연산자)
    -assertTrue(a)
        --조건 a가 참인가를 확인
    -assertNotNull(a)
        --객체 a가 null이 아님을 확인한다.
    -이외에도 다양한 assert 메소드가 존재함
        http://junit.sourceforge.net/javadoc/org/junit/Assert.html

3)jUnit을 사용한 DI 테스트 클래스 작성하기
    -jUnit을 사용한 DI 테스트 클래스(HelloBeanJunitTest.java) 작성
        --/src/com.example.test/HelloBeanTest.java 복사
```

```
--/src/com.example.test/ 붙여넣고 이름변경 -> HelloBeanJunitTest.java

package com.example.test;

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);
  }
}

-@Before에 마우스를 올려놓으면 Fix project setup... click
  --Add archive 'junit-4.12.jar ... > OK
  --import org.junit...에 마우스를 올려놓으면 Fix project setup... click
  --Add JUnit 4 library to the build path > OK
-right-click > Run As > Junit Test
-결과 -> Junit View에 초록색 bar
-만일, test1() 메소드를 jUnit에서 제외하고 싶을 때에는 @Test 옆에 @Ignore를 선언한다.
```

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

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


9. Spring TestContext Framework
  1)Spring-Test library 설치
    -http://mvnrepository.com에서 'spring-test'로 검색
    -검색 결과 목록에서 'Spring TestContext Framework' 클릭
    -버전 목록에서 4.3.24.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 테스트를 지원하는 어노테이션
    -@RunWith(SpringJUnit4ClassRunner.class)
        --jUnit 프레임워크의 테스트 실행방법을 확장할 때 사용하는 어노테이션
        --SpringJUnit4ClassRunner라는 클래스를 지정해주면 jUnit이 테스트를 진행하는 중에
        ApplicationContext를 만들고 관리하는 작업을 진행해 준다.
        --이 어노테이션은 각각의 테스트 별로 객체가 생성되더라도 Singleton의 ApplicationContext를 보장한다.

      -@ContextConfiguration
        --Spring bean 설정 파일의 위치를 지정할 때 사용되는 어노테이션

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

  3)Spring-Test를 사용할 DI 테스트 클래스-HelloBeanJunitSpringTest.java 작성하기
    -/src/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;
          import org.springframework.test.context.ContextConfiguration;
          import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
          ...
          @RunWith(SpringJUnit4ClassRunner.class)
```

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

```
745         </bean>
746
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>
795                   <value>설운도</value>
796                 </key>
797                 <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>


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
```

```
852
853       <Printer.java>
854         package com.example;
855
856         public interface Printer{
857           void print(String message);
858         }
859
860       -com.example > right-click > New > Class
861         Class Name : StringPrinter
862
863       <StringPrinter.java>
864         package com.example;
865
866         public class StringPrinter implements Printer{
867           private StringBuffer buffer = new StringBuffer();
868
869           @Override
870           public void print(String message){
871             this.buffer.append(message);
872           }
873
874           public String toString(){
875             return this.buffer.toString();
876           }
877         }
878
879       -com.example > right-click > New > Class
880         Class Name : ConsolePrinter
881
882       <ConsolePrinter.java>
883         package com.example;
884
885         public class ConsolePrinter implements Printer{
886
887           @Override
888           public void print(String message){
889             System.out.println(message);
890           }
891         }
892
893     4)Java Project를 Spring Project로 변환
894       -DIDemo Project > right-click > Configuration > Convert to Maven Project
895         --Project : /DIDemo1
896         --Group Id : DIDemo1
897         --Artifact Id : DIDemo1
898         --version : 0.0.1-SNAPSHOT
899         --Packaging : jar
900         --Finish
901
902       -DIDemo Project > right-click > Spring > Add Spring Project Nature
903
904       -pom.xml 파일에 Spring Context Dependency 추가하기
905         <version>0.0.1-SNAPSHOT</version>
```

```
906         <dependencies>   <--- dependencies element 추가
907           <dependency>   <---여기에 paste
908             <groupId>org.springframework</groupId>
909             <artifactId>spring-context</artifactId>
910             <version>4.3.24.RELEASE</version>
911           </dependency>
912         </dependencies>
913
914       -pom.xml > right-click > Run As > Maven install
915
916     5)src/config folder 생성
917       -/src > right-click > New > Folder
918         Folder name : config
919
920     6)Bean Configuration XML 작성
921       -/src/config > right-click > New > Other > Spring > Spring Bean Configuration File
922         File name : beans.xml > Next
923           Check [beans - http://www.springframework.org/schema/beans]
924           Check [http://www.springframework.org/schema/beans/spring-beans-4.3.xsd]
925           Finish
926
927         <?xml version="1.0" encoding="UTF-8"?>
928         <beans xmlns="http://www.springframework.org/schema/beans"
929           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
930           xsi:schemaLocation="http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans.xsd">
931
932           <bean id="hello" class="com.example.Hello">
933             <property name="name" value="Spring" />
934             <property name="printer" ref="printer" />
935           </bean>
936           <bean id="printer" class="com.example.StringPrinter" />
937           <bean id="consolePrinter" class="com.example.ConsolePrinter" />
938
939         </beans>
940
941     7)DI Test 클래스 작성
942       -/src/com.example > right-click > New > Package
943         Package Name : test
944       -/src/com.example/test/HelloBeanTest.java
945
946         package com.example.test;
947
948         import org.springframework.context.ApplicationContext;
949         import org.springframework.context.support.GenericXmlApplicationContext;
950
951         import com.example.Hello;
952         import com.example.Printer;
953
954         public class HelloBeanTest {
955           public static void main(String [] args){
956             //1. IoC Container 생성
957             ApplicationContext context =
958                 new GenericXmlApplicationContext("config/beans.xml");
```

```
959
960             //2. Hello Beans 가져오기
961             Hello hello = (Hello)context.getBean("hello");
962             System.out.println(hello.sayHello());
963             hello.print();
964
965             //3. SpringPrinter 가져오기
966             Printer printer = (Printer)context.getBean("printer");
967             System.out.println(printer.toString());
968
969             Hello hello2 = context.getBean("hello", Hello.class);
970             hello2.print();
971
972             System.out.println(hello == hello2);
973           }
974         }
975
976     8)Test
977       -/src/com.example.test/HelloBeanTest.java > right-click > Run As > Java Application
978       ------------------------------------------------------------------
979         Hello Spring
980         Hello Spring
981         true
982
983     9)jUnit으로 테스트
984       -jUnit Library 설치
985         --jUnit 4.12 버전을 pom.xml에 추가
986
987         <dependency>
988             <groupId>junit</groupId>
989             <artifactId>junit</artifactId>
990             <version>4.12</version>
991             <scope>test</scope>
992         </dependency>
993
994         --pom.xml > right-click > Run As > Maven Install
995
996       -jUnit을 사용한 DI 테스트 클래스(HelloBeanJunitTest.java) 작성
997         --/src/com.example.test/HelloBeanTest.java 복사
998         --/src/com.example.test/ 붙여넣고 이름변경 -> HelloBeanJunitTest.java
999
1000        package com.example.test;
1001
1002        import org.junit.Before;
1003        import org.junit.Test;
1004        import org.springframework.context.ApplicationContext;
1005        import org.springframework.context.support.GenericXmlApplicationContext;
1006
1007        import com.example.Hello;
1008        import com.example.Printer;
1009
1010        import static org.junit.Assert.assertEquals;
1011        import static org.junit.Assert.assertSame;
1012
```

```
1013        public class HelloBeanJunitTest {
1014          ApplicationContext context;
1015
1016          @Before
1017          public void init(){
1018            context = new GenericXmlApplicationContext("config/beans.xml");
1019          }
1020
1021          @Test
1022          public void test1(){
1023            Hello hello = (Hello)context.getBean("hello");
1024            assertEquals("Hello Spring", hello.sayHello());
1025            hello.print();
1026
1027            Printer printer = (Printer)context.getBean("printer");
1028            assertEquals("Hello Spring", printer.toString());
1029          }
1030
1031          @Test
1032          public void test2(){
1033            Hello hello = (Hello)context.getBean("hello");
1034
1035            Hello hello2 = context.getBean("hello", Hello.class);
1036            assertSame(hello, hello2);
1037          }
1038        }
1039
1040      -right-click > Run As > Junit Test
1041      -결과 -> Junit View에 초록색 bar
1042
1043    10)Spring-Test를 사용할 DI 테스트 클래스-HelloBeanJunitSpringTest.java 작성하기
1044      -Spring-Test library 설치
1045      -pom.xml 코드 추가
1046         <dependency>
1047        <groupId>org.springframework</groupId>
1048        <artifactId>spring-test</artifactId>
1049        <version>4.3.9.RELEASE</version>
1050        <scope>test</scope>
1051       </dependency>
1052
1053      -pom.xml > right-click > Maven Install
1054
1055      -Spring-Test를 사용할 DI 테스트 클래스-HelloBeanJunitSpringTest.java 작성하기
1056        --/src/com.example.test/HelloBeanJunitTest.java 복사해서
1057        --/src/com.example.test/HelloBeanJunitSpringTest.java 로 붙여넣기
1058
1059          import org.junit.runner.RunWith;
1060          import org.springframework.beans.factory.annotation.Autowired;
1061          import org.springframework.test.context.ContextConfiguration;
1062          import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
1063          ...
1064          @RunWith(SpringJUnit4ClassRunner.class)
1065          @ContextConfiguration(locations="classpath:config/beans.xml")
1066          public class HelloBeanJunitSpringTest {
```

```
1067
1068            @Autowired
1069            ApplicationContext context;
1070
1071      -right-click > Run As > Junit Test
1072      -결과 -> Junit View에 초록색 bar
1073
1074   11)Hello class 수정
1075
1076      ...
1077      private List<String> names;
1078      ...
1079      public void setNames(List<String> list){
1080        this.names = list;
1081      }
1082
1083      public List<String> getNames(){
1084        return this.names;
1085      }
1086      ...
1087
1088   12)beans.xml 수정
1089      <bean id="hello2" class="com.example.Hello">
1090        <property name="names">
1091          <list>
1092            <value>AOP</value>
1093            <value>Spring</value>
1094            <value>DI</values>
1095          </list>
1096        </property>
1097      </bean>
1098
1099   13)HelloBeanJunitTest로 테스트하기
1100
1101        @Test <--@Ignore 붙여서 테스트하지 않고
1102        public void test1(){
1103          //2. Hello Beans 가져오기
1104          Hello hello = (Hello)context.getBean("hello");
1105          assertEquals("Hello Spring", hello.sayHello());
1106          hello.print();
1107
1108          //3. SpringPrinter 가져오기
1109          Printer printer = (Printer)context.getBean("printer");
1110          assertEquals("Hello Spring", printer.toString());
1111        }
1112
1113        @Test @Ignore <-- @Ignore를 해제하여 코드 수정하기
1114        public void test2(){
1115          Hello hello = (Hello)context.getBean("hello");
1116
1117          Hello hello2 = context.getBean("hello", Hello.class);
1118          assertSame(hello, hello2);
1119
1120          //아래 코드 추가
```

```
1121          assertEquals(3, hello2.getNames().size());
1122          List<String> list = hello.getNames();
1123          for(String value : list){
1124            System.out.println(value);
1125          }
1126        }
1127
1128      -right-click > Run As > Junit Test
1129      -결과 -> Junit View에 초록색 bar
1130
1131
1132  13. setter를 이용한 의존주입하기 실습
1133    1)In Package Explorer > right-click > New > Java Project
1134      -Project Name : SpringDemo
1135    2)src > right-click > New > Package
1136      Package name : com.example
1137
1138    3)POJO class 작성
1139      -com.example > right-click > New > Class
1140      -com.example.BmiCalculator.java
1141
1142      package com.example;
1143
1144      public class BmiCalculator {
1145        private double lowWeight;
1146        private double normal;
1147        private double overWeight;
1148        private double obesity;
1149
1150        public void setLowWeight(double lowWeight) {
1151          this.lowWeight = lowWeight;
1152        }
1153
1154        public void setNormal(double normal) {
1155          this.normal = normal;
1156        }
1157
1158        public void setOverWeight(double overWeight) {
1159          this.overWeight = overWeight;
1160        }
1161
1162        public void setObesity(double obesity) {
1163          this.obesity = obesity;
1164        }
1165        public void bmiCalcu(double weight, double height){
1166          double h = height * 0.01;
1167          double result = weight / (h * h);
1168
1169          System.out.println("BMI 지수 : " + (int)result);
1170
1171          if(result > obesity)
1172            System.out.println("비만입니다.");
1173          else if(result > overWeight)
1174            System.out.println("과체중입니다.");
```

```
        else if(result > normal)
          System.out.println("정상입니다.");
        else
          System.out.println("저체중입니다.");
      }
    }

4)com.example.MyInfo.java
  package com.example;

  import java.util.ArrayList;

  public class MyInfo {
    private String name;
    private double height;
    private double weight;
    private ArrayList<String> hobby;
    private BmiCalculator bmiCalculator;

    public void setBmiCalculator(BmiCalculator bmiCalculator) {
      this.bmiCalculator = bmiCalculator;
    }
    public void setName(String name) {
      this.name = name;
    }
    public void setHeight(double height) {
      this.height = height;
    }
    public void setWeight(double weight) {
      this.weight = weight;
    }
    public void setHobby(ArrayList<String> hobby) {
      this.hobby = hobby;
    }
    public void getInfo(){
      System.out.println("Name : " + this.name);
      System.out.println("Height : " + this.height);
      System.out.println("Weight : " + this.weight);
      System.out.println("Hobby : " + this.hobby);
      this.bmiCalcu();
    }
    public void bmiCalcu(){
      this.bmiCalculator.bmiCalcu(this.weight, this.height);
    }
  }

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

```
1229        --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
1230
1231      -SpringDemo Project > right-click > Spring > Add Spring Project Nature
1232        --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
1233
1234      -pom.xml 파일에 Spring Context Dependency 추가하기
1235       <version>0.0.1-SNAPSHOT</version>
1236        <dependencies>   <--- dependencies element 추가
1237          <dependency>   <---여기에 paste
1238            <groupId>org.springframework</groupId>
1239            <artifactId>spring-context</artifactId>
1240            <version>4.3.24.RELEASE</version>
1241          </dependency>
1242        </dependencies>
1243
1244      -pom.xml > right-click > Run As > Maven install
1245      [INFO] BUILD SUCCESS 확인
1246
1247    6)SpringDemo/resources folder 생성
1248      -SpringDemo project > right-click > Build Path > Configure Build Path
1249      -Source Tab > Add Folder
1250      -SpringDemo click
1251      -Create New Folder > Folder name : resources > Finish > OK
1252      -SpringDemo/resources(new) 확인
1253      -Apply and Close
1254
1255    7)Bean Configuration XML 작성
1256      -SpringDemo/resources > right-click > New > Other > Spring > Spring Bean Configuration
          File
1257      -File name : applicationContext.xml > Finish
1258
1259      <bean id="bmiCalculator" class="com.example.BmiCalculator">
1260        <property name="lowWeight" value="18.5" />
1261        <property name="normal" value="23" />
1262        <property name="overWeight" value="25" />
1263        <property name="obesity">
1264          <value>30</value>
1265        </property>
1266      </bean>
1267      <bean id="myInfo" class="com.example.MyInfo">
1268        <property name="name" value="한지민" />
1269        <property name="height" value="170.5" />
1270        <property name="weight" value="67" />
1271        <property name="hobby">
1272          <list>
1273            <value>수영</value>
1274            <value>요리</value>
1275            <value>독서</value>
1276          </list>
1277        </property>
1278        <property name="bmiCalculator">
1279          <ref bean="bmiCalculator" />
1280        </property>
1281      </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> 태그를 사용할 수 있다.

2)Constructor 주입방식은 생성자의 파라미터를 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(){}
```

```
1336          public void setName(String name){
1337            this.name = name;
1338          }
1339
1340          public void setPrinter(Printer printer){
1341            this.printer = printer;
1342          }
1343
1344          public String sayHello(){
1345            return "Hello " + name;
1346          }
1347
1348          public void print(){
1349            this.printer.print(sayHello());
1350          }
1351        }
1352
1353      -com.example > right-click > New > Interface
1354        interface name : Printer
1355
1356      <Printer.java>
1357        package com.example;
1358
1359        public interface Printer{
1360          void print(String message);
1361        }
1362
1363      -com.example > right-click > New > Class
1364        Class Name : StringPrinter
1365
1366      <StringPrinter.java>
1367        package com.example;
1368
1369        public class StringPrinter implements Printer{
1370          private StringBuffer buffer = new StringBuffer();
1371
1372          @Override
1373          public void print(String message){
1374            this.buffer.append(message);
1375          }
1376
1377          public String toString(){
1378            return this.buffer.toString();
1379          }
1380        }
1381
1382      -com.example > right-click > New > Class
1383        Class Name : ConsolePrinter
1384
1385      <ConsolePrinter.java>
1386        package com.example;
1387
1388        public class ConsolePrinter implements Printer{
1389
```

```
1390          @Override
1391          public void print(String message){
1392            System.out.println(message);
1393          }
1394        }
1395
1396    4)Java Project를 Spring Project로 변환
1397      -DIDemo2 Project > right-click > Configuration > Convert to Maven Project
1398        --Project : /DIDemo2
1399        --Group Id : DIDemo2
1400        --Artifact Id : DIDemo2
1401        --version : 0.0.1-SNAPSHOT
1402        --Packaging : jar
1403        --Finish
1404        --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
1405
1406      -DIDemo2 Project > right-click > Spring > Add Spring Project Nature
1407        --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
1408
1409      -pom.xml 파일에 Spring Context Dependency 추가하기
1410       <version>0.0.1-SNAPSHOT</version>
1411        <dependencies>   <--- dependencies element 추가
1412          <dependency>   <---여기에 paste
1413            <groupId>org.springframework</groupId>
1414            <artifactId>spring-context</artifactId>
1415            <version>4.3.24.RELEASE</version>
1416          </dependency>
1417        </dependencies>
1418
1419      -pom.xml > right-click > Run As > Maven install
1420      [INFO] BUILD SUCCESS 확인
1421
1422    5)DIDemo2/resources folder 생성
1423      -DIDemo2 project > right-click > Build Path > Configure Build Path
1424      -Source Tab > Add Folder
1425      -DIDemo2 click
1426      -Create New Folder > Folder name : resources > Finish > OK
1427      -DIDemo2/resources(new) 확인
1428      -Apply and Close
1429
1430    6)Bean Configuration XML 작성
1431      -DIDemo2/resources > right-click > New > Other > Spring > Spring Bean Configuration File
1432      -File name : beans.xml > Finish
1433
1434        <?xml version="1.0" encoding="UTF-8"?>
1435        <beans xmlns="http://www.springframework.org/schema/beans"
1436          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1437          xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
1438
1439          <bean id="hello" class="com.example.Hello">
1440            <property name="name" value="Spring" />
1441            <property name="printer" ref="printer" />
1442          </bean>
```

```
1443          <bean id="printer" class="com.example.StringPrinter" />
1444          <bean id="consolePrinter" class="com.example.ConsolePrinter" />
1445
1446       </beans>
1447
1448    7)Test 클래스 작성
1449      -/src/com.example > right-click > New > Package
1450        Package Name : test
1451      -/src/com.example/test/HelloBeanTest.java
1452
1453        package com.example.test;
1454
1455        import org.springframework.context.ApplicationContext;
1456        import org.springframework.context.support.GenericXmlApplicationContext;
1457
1458        import com.example.Hello;
1459        import com.example.Printer;
1460
1461        public class HelloBeanTest {
1462          public static void main(String [] args){
1463            //1. IoC Container 생성
1464            ApplicationContext context =
1465                 new GenericXmlApplicationContext("classpath:beans.xml");
1466
1467            //2. Hello Beans 가져오기
1468            Hello hello = (Hello)context.getBean("hello");
1469            System.out.println(hello.sayHello());
1470            hello.print();
1471
1472            //3. SpringPrinter 가져오기
1473            Printer printer = (Printer)context.getBean("printer");
1474            System.out.println(printer.toString());
1475
1476            Hello hello2 = context.getBean("hello", Hello.class);
1477            hello2.print();
1478
1479            System.out.println(hello == hello2);  //Singleton Pattern
1480          }
1481        }
1482
1483    8)Test
1484      -/src/com.example.test/HelloBeanTest.java > right-click > Run As > Java Application
1485      ------------------------------------------------------------------
1486        Hello Spring
1487        Hello Spring
1488        true
1489
1490    9)/src/com.example.Hello 생성자 추가
1491
1492      public Hello(String name, Printer printer) {
1493        this.name = name;
1494        this.printer = printer;
1495      }
1496
```

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;
  }
```

```
1551        public void setAge(int age) {
1552          this.age = age;
1553        }
1554        public int getGrade() {
1555          return grade;
1556        }
1557        public void setGrade(int grade) {
1558          this.grade = grade;
1559        }
1560        public int getClassNum() {
1561          return classNum;
1562        }
1563        public void setClassNum(int classNum) {
1564          this.classNum = classNum;
1565        }
1566      }
1567
1568    4)com.example.StudentInfo.java
1569      package com.example;
1570
1571      public class StudentInfo {
1572        private Student student;
1573
1574        public StudentInfo(Student student) {
1575          this.student = student;
1576        }
1577
1578        public void printInfo(){
1579          if(this.student != null){
1580            System.out.println("Name : " + this.student.getName());
1581            System.out.println("Age : " + this.student.getAge());
1582            System.out.println("Grade : " + this.student.getGrade());
1583            System.out.println("Class : " + this.student.getClassNum());
1584            System.out.println("------------------------");
1585          }
1586        }
1587
1588        public void setStudent(Student student){
1589          this.student = student;
1590        }
1591      }
1592
1593    5)Java Project를 Spring Project로 변환
1594      -SpringDemo1 Project > right-click > Configuration > Convert to Maven Project
1595        --Project : /SpringDemo1
1596        --Group Id : SpringDemo1
1597        --Artifact Id : SpringDemo1
1598        --version : 0.0.1-SNAPSHOT
1599        --Packaging : jar
1600        --Finish
1601        --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
1602
1603      -SpringDemo1 Project > right-click > Spring > Add Spring Project Nature
1604        --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
```

-pom.xml 파일에 Spring Context Dependency 추가하기
<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
[INFO] BUILD SUCCESS 확인

6)SpringDemo1/resources folder 생성
-SpringDemo1 project > right-click > Build Path > Configure Build Path
-Source Tab > Add Folder
-SpringDemo1 click
-Create New Folder > Folder name : resources > Finish > OK
-SpringDemo1/resources(new) 확인
-Apply and Close

7)Bean Configuration XML 작성
-SpringDemo1/resources > right-click > New > Other > Spring > Spring Bean Configuration File
-File name : applicationContext.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="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>
```

```
1657
1658        <bean id="studentInfo" class="com.example.StudentInfo">
1659          <constructor-arg>
1660            <ref bean="student1"/>
1661          </constructor-arg>
1662        </bean>
1663      </beans>
1664
1665    8)com.example.MainClass.java
1666      package com.example;
1667
1668      import org.springframework.context.support.AbstractApplicationContext;
1669      import org.springframework.context.support.GenericXmlApplicationContext;
1670
1671      public class MainClass {
1672        public static void main(String[] args) {
1673          String configFile = "classpath:applicationContext.xml";
1674          AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);
1675          StudentInfo studentInfo = context.getBean("studentInfo", StudentInfo.class);
1676          studentInfo.printInfo();
1677
1678          Student student2 = context.getBean("student2", Student.class);
1679          studentInfo.setStudent(student2);
1680          studentInfo.printInfo();
1681
1682          context.close();
1683        }
1684      }
1685
1686    9)결과
1687      Name : 한지민
1688      Age : 15
1689      Grade : 2
1690      Class : 5
1691      ------------------------
1692      Name : 설운도
1693      Age : 16
1694      Grade : 3
1695      Class : 7
1696      ------------------------
1697
1698
1699  17. DI의 장점
1700    1)Java파일의 수정 없이 스프링 설정 파일만을 수정하여 부품들을 생성/조립할 수 있다.
1701
1702    2)com.example.Car.java Interface
1703      package com.example;
1704
1705      public interface Car {
1706        void drive();
1707      }
1708
1709    3)com.example.Sonata.java
1710      package com.example;
```

```
1711
1712      public class Sonata implements Car {
1713
1714        @Override
1715        public void drive() {
1716          System.out.println("Drive a Sonata");
1717        }
1718      }
1719
1720    4)com.example.Carnival.javaa
1721      package com.example;
1722
1723      public class Carnival implements Car {
1724
1725        @Override
1726        public void drive() {
1727          System.out.println("Drive a Carnival");
1728        }
1729      }
1730
1731    5)com.example.HybridCar.java
1732      package com.example;
1733
1734      public class HybridCar extends Sonata implements Car {
1735        @Override
1736        public void drive(){
1737          System.out.println("Drive a HybridCar with Sonata");
1738        }
1739      }
1740
1741    6)CarContext.xml
1742      <?xml version="1.0" encoding="UTF-8"?>
1743      <beans xmlns="http://www.springframework.org/schema/beans"
1744        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1745        xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
1746
1747        <!-- <bean id="car" class="com.example.Sonata" /> -->
1748        <!-- <bean id="car" class="com.example.Carnival" /> -->
1749        <bean id="car" class="com.example.HybridCar" />
1750        //CarMainClass를 변경하지 않고, CarContext.xml만 변경해도 여러 클래스를 이용할 수 있다.
1751      </beans>
1752
1753    7)com.example.CarMainClass.java
1754      package com.example;
1755
1756      import org.springframework.context.support.AbstractApplicationContext;
1757      import org.springframework.context.support.GenericXmlApplicationContext;
1758
1759      public class CarMainClass {
1760        public static void main(String[] args) {
1761          String configFile = "classpath:CarContext.xml";
1762          AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);
1763          Car car = context.getBean("car", Car.class);
```

```
        car.drive();

        context.close();
      }
    }


18. Context 파일 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
      public String toString() {
        return String.format("Student [name=%s, age=%s, hobbys=%s, height=%s,
        weight=%s]", name, age, hobbys, height,
            weight);
      }
    }

  4)com.example.StudentInfo.java
```

```
1817    package com.example;
1818    public class StudentInfo {
1819      private Student student;
1820
1821      public Student getStudent() {
1822        return student;
1823      }
1824
1825      public void setStudent(Student student) {
1826        this.student = student;
1827      }
1828    }
1829
1830  5)com.example.Product.java
1831    package com.example;
1832    public class Product {
1833      private String pName;
1834      private int pPrice;
1835      private String maker;
1836      private String color;
1837      public Product(String pName, int pPrice) {
1838        this.pName = pName;
1839        this.pPrice = pPrice;
1840      }
1841      public void setpName(String pName) {
1842        this.pName = pName;
1843      }
1844      public void setpPrice(int pPrice) {
1845        this.pPrice = pPrice;
1846      }
1847      public void setMaker(String maker) {
1848        this.maker = maker;
1849      }
1850      public void setColor(String color) {
1851        this.color = color;
1852      }
1853      @Override
1854      public String toString() {
1855        return String.format("Product [pName=%s, pPrice=%s, maker=%s, color=%s]",
          pName, pPrice, maker, color);
1856      }
1857    }
1858
1859  6)Java Project를 Spring Project로 변환
1860    -SpringDemo2 Project > right-click > Configuration > Convert to Maven Project
1861      --Project : /SpringDemo2
1862      --Group Id : SpringDemo2
1863      --Artifact Id : SpringDemo2
1864      --version : 0.0.1-SNAPSHOT
1865      --Packaging : jar
1866      --Finish
1867      --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
1868
1869    -SpringDemo2 Project > right-click > Spring > Add Spring Project Nature
```

```
1870        --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
1871
1872      -pom.xml 파일에 Spring Context Dependency 추가하기
1873       <version>0.0.1-SNAPSHOT</version>
1874        <dependencies>   <--- dependencies element 추가
1875          <dependency>   <---여기에 paste
1876            <groupId>org.springframework</groupId>
1877            <artifactId>spring-context</artifactId>
1878            <version>4.3.24.RELEASE</version>
1879          </dependency>
1880        </dependencies>
1881
1882      -pom.xml > right-click > Run As > Maven install
1883      [INFO] BUILD SUCCESS 확인
1884
1885    7)SpringDemo2/resources folder 생성
1886      -SpringDemo2 project > right-click > Build Path > Configure Build Path
1887      -Source Tab > Add Folder
1888      -SpringDemo2 click
1889      -Create New Folder > Folder name : resources > Finish > OK
1890      -SpringDemo2/resources(new) 확인
1891      -Apply and Close
1892
1893    8)Bean Configuration XML 작성
1894      -SpringDemo2/resources > right-click > New > Other > Spring > Spring Bean Configuration
          File
1895      -File name : applicationContext.xml > Finish
1896
1897    9)applicationContext.xml
1898      <?xml version="1.0" encoding="UTF-8"?>
1899      <beans xmlns="http://www.springframework.org/schema/beans"
1900        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1901        xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
1902
1903        <bean id="student1" class="com.example.Student">
1904          <constructor-arg value="한지민" />
1905          <constructor-arg value="25" />
1906          <constructor-arg>
1907            <list>
1908              <value>독서</value>
1909              <value>영화감상</value>
1910              <value>요리</value>
1911            </list>
1912          </constructor-arg>
1913          <property name="height" value="165" />
1914          <property name="weight">
1915            <value>45</value>
1916          </property>
1917        </bean>
1918
1919        <bean id="studentInfo1" class="com.example.StudentInfo">
1920          <property name="student">
1921            <ref bean="student1" />
```

```
        </property>
      </bean>
    </beans>

10)/resources/applicationContext2.xml
  -또 하나의 file을 생성한다.
  -위의 applicationContext.xml을 복사하여 붙여 넣기 한다.
  -Names tab을 선택하여 c, p를 선택한다.
  <?xml version="1.0" encoding="UTF-8"?>
  <beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:c="http://www.springframework.org/schema/c"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="student3" class="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>

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

11)com.example.MainClass
  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";
      String configFile1 = "classpath:applicationContext2.xml";
      AbstractApplicationContext context = new GenericXmlApplicationContext(configFile,
      configFile1);
      Student student1 = context.getBean("student1", Student.class);
      System.out.println(student1);

      StudentInfo studentInfo = context.getBean("studentInfo1", StudentInfo.class);
      Student student2 = studentInfo.getStudent();
```

```
1973            System.out.println(student2);
1974            if(student1.equals(student2)) System.out.println("Equals");
1975            else System.out.println("Different");
1976
1977            Student student3 = context.getBean("student3", Student.class);
1978            System.out.println(student3);
1979
1980            if(student1.equals(student3)) System.out.println("Equals");
1981            else System.out.println("Different");
1982
1983            Product product = context.getBean("product", Product.class);
1984            System.out.println(product);
1985            context.close();
1986         }
1987      }
1988
1989      12)결과
1990        Student [name=한지민, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
1991        Student [name=한지민, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
1992        Equals
1993        Student [name=설운도, age=50, hobbys=[노래부르기, 게임], height=175.0,weight=75.0]
1994        Different
1995        Product [pName=Computer, pPrice=2000000, maker=Samsung, color=Yellow]
1996
1997
1998    19. Java Annotation을 이용한 DI 설정하기
1999      1)@Configuration
2000        public class ApplicationConfig{}   //@Configuration : 이 클래스는 스프링 설정에 사용되는 클래스 입니다'
            라고 명시해 주는 어노테이션
2001
2002      2)@Bean
2003        public class Student1(){}   //@Bean : 객체생성
2004
2005      3)In Package Explorer > right-click > New > Java Project
2006        -Project Name : SpringDemo3
2007
2008      4)src > right-click > New > Package
2009        -Package name : com.example
2010
2011      5)com.example.Student.java
2012        package com.example;
2013
2014        import java.util.ArrayList;
2015
2016        public class Student {
2017          private String name;
2018          private int age;
2019          private ArrayList<String> hobbys;
2020          private double height;
2021          private double weight;
2022          public Student(String name, int age, ArrayList<String> hobbys) {
2023            this.name = name;
2024            this.age = age;
2025            this.hobbys = hobbys;
```

```
2026        }
2027        public void setName(String name) {
2028          this.name = name;
2029        }
2030        public void setAge(int age) {
2031          this.age = age;
2032        }
2033        public void setHobbys(ArrayList<String> hobbys) {
2034          this.hobbys = hobbys;
2035        }
2036        public void setHeight(double height) {
2037          this.height = height;
2038        }
2039        public void setWeight(double weight) {
2040          this.weight = weight;
2041        }
2042        @Override
2043        public String toString() {
2044          return String.format("Student [name=%s, age=%s, hobbys=%s, height=%s,
            weight=%s]", name, age, hobbys, height,
2045              weight);
2046        }
2047      }
2048
2049    6)Java Project를 Spring Project로 변환
2050      -SpringDemo3 Project > right-click > Configuration > Convert to Maven Project
2051        --Project : /SpringDemo3
2052        --Group Id : SpringDemo3
2053        --Artifact Id : SpringDemo3
2054        --version : 0.0.1-SNAPSHOT
2055        --Packaging : jar
2056        --Finish
2057        --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
2058
2059      -SpringDemo3 Project > right-click > Spring > Add Spring Project Nature
2060        --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
2061
2062      -pom.xml 파일에 Spring Context Dependency 추가하기
2063       <version>0.0.1-SNAPSHOT</version>
2064        <dependencies>   <--- dependencies element 추가
2065          <dependency>   <---여기에 paste
2066            <groupId>org.springframework</groupId>
2067            <artifactId>spring-context</artifactId>
2068            <version>4.3.24.RELEASE</version>
2069          </dependency>
2070        </dependencies>
2071
2072      -pom.xml > right-click > Run As > Maven install
2073      [INFO] BUILD SUCCESS 확인
2074
2075    7)com.example.ApplicationConfig.java
2076      import org.springframework.context.annotation.Bean;
2077      import org.springframework.context.annotation.Configuration;
2078
```

```
@Configuration
public class ApplicationConfig {

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

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

    return student;
  }

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

    return student;
  }
}
```

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

    context.close();
  }
}
```

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]
```

20. Java Annotation과 XML 을 이용한 DI 설정 방법 : XML 파일에 Java 파일을 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
      --Project : /SpringDemo4
      --Group Id : SpringDemo4
      --Artifact Id : SpringDemo4
      --version : 0.0.1-SNAPSHOT
      --Packaging : jar
      --Finish

```
2185        --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
2186
2187      -SpringDemo4 Project > right-click > Spring > Add Spring Project Nature
2188        --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
2189
2190      -pom.xml 파일에 Spring Context Dependency 추가하기
2191       <version>0.0.1-SNAPSHOT</version>
2192        <dependencies>   <--- dependencies element 추가
2193          <dependency>   <---여기에 paste
2194            <groupId>org.springframework</groupId>
2195            <artifactId>spring-context</artifactId>
2196            <version>4.3.24.RELEASE</version>
2197          </dependency>
2198        </dependencies>
2199
2200      -pom.xml > right-click > Run As > Maven install
2201      [INFO] BUILD SUCCESS 확인
2202
2203
2204    5)com.example.ApplicationConfig.java
2205      package com.example;
2206
2207      import java.util.ArrayList;
2208
2209      import org.springframework.context.annotation.Bean;
2210      import org.springframework.context.annotation.Configuration;
2211
2212      @Configuration
2213      public class ApplicationConfig {
2214        @Bean
2215        public Student student1(){
2216          ArrayList<String> hobbys = new ArrayList<String>();
2217          hobbys.add("독서");
2218          hobbys.add("영화감상");
2219          hobbys.add("요리");
2220
2221          Student student = new Student("한지민", 25, hobbys);
2222          student.setHeight(165);
2223          student.setWeight(45);
2224
2225          return student;
2226        }
2227      }
2228
2229    6)SpringDemo4/resources folder 생성
2230      -SpringDemo4 project > right-click > Build Path > Configure Build Path
2231      -Source Tab > Add Folder
2232      -SpringDemo4 click
2233      -Create New Folder > Folder name : resources > Finish > OK
2234      -SpringDemo4/resources(new) 확인
2235      -Apply and Close
2236
2237    7)Bean Configuration XML 작성
2238      -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]
```

21. Java Annotation과 XML 을 이용한 DI 설정 방법 : Java 파일에 XML 파일을 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
--Group Id : SpringDemo5
--Artifact Id : SpringDemo5
--version : 0.0.1-SNAPSHOT
--Packaging : jar
--Finish
--Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.

-SpringDemo5 Project > right-click > Spring > Add Spring Project Nature
--Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.

-pom.xml 파일에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<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
[INFO] BUILD SUCCESS 확인

5)SpringDemo5/resources folder 생성
-SpringDemo5 project > right-click > Build Path > Configure Build Path
-Source Tab > Add Folder
-SpringDemo5 click
-Create New Folder > Folder name : resources > Finish > OK
-SpringDemo5/resources(new) 확인
-Apply and Close

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

7)/resources/applicationContext.xml

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

  <bean id="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>
```

8)com.example.ApplicationConfig.java

```
2395      package com.example;
2396
2397      import java.util.ArrayList;
2398
2399      import org.springframework.context.annotation.Bean;
2400      import org.springframework.context.annotation.Configuration;
2401      import org.springframework.context.annotation.ImportResource;
2402
2403      @Configuration
2404      @ImportResource("classpath:ApplicationContext.xml")
2405      public class ApplicationConfig {
2406
2407        @Bean
2408        public Student student1(){
2409          ArrayList<String> hobbys = new ArrayList<String>();
2410          hobbys.add("독서");
2411          hobbys.add("영화감상");
2412          hobbys.add("요리");
2413
2414          Student student = new Student("한지민", 25, hobbys);
2415          student.setHeight(165);
2416          student.setWeight(45);
2417
2418          return student;
2419        }
2420      }
2421
2422    9)com.example.MainClass.java
2423      package com.example;
2424
2425      import org.springframework.context.annotation.AnnotationConfigApplicationContext;
2426
2427      public class MainClass {
2428        public static void main(String[] args) {
2429          AnnotationConfigApplicationContext context = new
            AnnotationConfigApplicationContext(ApplicationConfig.class);
2430          Student student1 = context.getBean("student1", Student.class);
2431          System.out.println(student1);
2432
2433          Student student3 = context.getBean("student3", Student.class);
2434          System.out.println(student3);
2435
2436          context.close();
2437        }
2438      }
2439
2440    10)result
2441      Student [name=한지민, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
2442      Student [name=설운도, age=50, hobbys=[노래부르기, 게임], height=175.0,weight=75.0]
2443
2444
2445  22. Bean 등록 메타 정보 구성 전략
2446    1)전략 1 - XML 단독 사용
2447      -모든 Bean을 명시적으로 XML에 등록하는 방법이다.
```

-생성되는 모든 Bean을 XML에서 확인할 수 있다는 장점이 있으나 Bean의 갯수가 많아지면 XML 파일을 관리하기 번거로울 수 있다.
-여러 개발자가 같은 설정파일을 공유해서 개발하다보면 설정파일을 동시에 수정하다가 충돌이 일어나는 경우도 적지 않다.
-DI에 필요한 적절한 setter 메소드 또는 constructor가 코드 내에 반드시 존재해야 한다.
-개발 중에는 어노테이션 설정방법을 사용했지만, 운영중에는 관리의 편의성을 위해 XML 설정으로 변경하는 전략을 쓸 수도 있다.

2)전략 2 - XML과 Bean Scanning의 혼용
-Bean으로 사용될 클래스에 특별한 어노테이션을 부여해주면 이런 클래스를 자동으로 찾아서 Bean으로 등록한다.
-특정 어노테이션이 붙은 클래스를 자동으로 찾아서 Bean으로 등록해 주는 방식을 Bean Scanning을 통한 자동인식 Bean 등록기능이라고 한다.
-어노테이션을 부여하고 자동 스캔으로 빈을 등록하면 XML 문서 생성과 관리에 따른 수고를 덜어주고 개발 속도를 향상시킬 수 있다.
-어플리케이션에 등록될 bean이 어떤 것들이 있고, bean들 간의 의존관계가 어떻게 되는지를 한눈에 파악할 수 없다는 단점이 있다.

3)주의 사항
-library형태로 제공되는 Class는 반드시 XML 설정을 통해서만 사용할 수 있다.
-예를 들면, Apache에서 제공하는 BasicDataSource 클래스를 사용하여 DB 연동을 처리한다면 commons-dbcp-1.4.jar 파일에 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>
```

23. Bean등록 및 의존관계 설정하는 Annotation

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

2)Bean 의존관계 주입 Annotation
-@Autowired, @Resource 어노테이션은 의존하는 객체를 자동으로 주입해 주는 어노테이션이다.
-@Autowired는 타입으로, @Resource는 이름으로 연결한다는 점이 다르다.
-@Autowired
--정밀한 의존관계 주입(Depencdency Injection)이 필요한 경우에 유용하다.
--property, setter 메소드, 생성자, 일반메소드에 적용 가능하다.
--의존하는 객체를 주입할 때 주로 Type을 이용하게 된다.
--<property>, <constructor-arg>태그와 동일한 역할을 한다.
--아래 Component Scan을 지원하는 태그부분에서 언급했듯이 Bean 설정파일에서 <context:component-scan>를 설정해 주면 DI 컨테이너는 해당 패키지에서 @Autowired가 붙은 인스턴스 변수의 형에 대입할 수 있는 클래스를 @Component가 붙은 클래스 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 setter메소드를 필요로 하지 않게 한다.

@Autowired
private String name;

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

@Resource(name="stringPrinter")
private Printer printer;

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

-@Qualifier
--@Autowired 어노테이션과 같이 사용되어 진다.
--@Autowired는 타입으로 찾아서 주입하므로, 동일한 타입의 Bean객체가 여러 개 존재할 때 특정 Bean을 찾기 위해서는 @Qualifier를 같이 사용해야 한다.
--의존성 주입 대상이 되는 객체가 두 개 이상일 때 Container는 어떤 객체를 할당할지 스스로 판단할 수 없어서 에러를 발생한다.
--NoUniqueBeanDefinitionException이 발생한다.

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

-@Injection
--@Autowired와 동일한 기능을 제공

3)Component Scan을 지원하는 태그
-<context:component-scan> 태그
--@Component를 통해 자동으로 Bean을 등록하고, @Autowired로 의존관계를 주입받는 어노테이션을 클래스에서 선언하여 사용했을 경우에는 해당 클래스가 위치한 특정 패키지를 Scan하기 위한 설정을 XML에 해주어야 한다.

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

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

4)사용 예
<SpringPrinter.java>
package 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();
          }
        }


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

  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);
      }

```

```
2594      -com.example > right-click > New > Class
2595        Class Name : StringPrinter
2596
2597      <StringPrinter.java>
2598        package com.example;
2599
2600        public class StringPrinter implements Printer{
2601          private StringBuffer buffer = new StringBuffer();
2602
2603          @Override
2604          public void print(String message){
2605            this.buffer.append(message);
2606          }
2607
2608          public String toString(){
2609            return this.buffer.toString();
2610          }
2611        }
2612
2613      -com.example > right-click > New > Class
2614        Class Name : ConsolePrinter
2615
2616      <ConsolePrinter.java>
2617        package com.example;
2618
2619        public class ConsolePrinter implements Printer{
2620
2621          @Override
2622          public void print(String message){
2623            System.out.println(message);
2624          }
2625        }
2626
2627    4)Java Project를 Spring Project로 변환
2628      -DIDemo3 Project > right-click > Configuration > Convert to Maven Project
2629        --Project : /DIDemo3
2630        --Group Id : DIDemo3
2631        --Artifact Id : DIDemo3
2632        --version : 0.0.1-SNAPSHOT
2633        --Packaging : jar
2634        --Finish
2635        --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
2636
2637      -DIDemo3 Project > right-click > Spring > Add Spring Project Nature
2638        --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
2639
2640      -pom.xml 파일에 Spring Context Dependency 추가하기
2641        <version>0.0.1-SNAPSHOT</version>
2642        <dependencies>   <--- dependencies element 추가
2643          <dependency>   <---여기에 paste
2644            <groupId>org.springframework</groupId>
2645            <artifactId>spring-context</artifactId>
2646            <version>4.3.24.RELEASE</version>
2647          </dependency>
```

```
2648        </dependencies>
2649
2650      -pom.xml > right-click > Run As > Maven install
2651      [INFO] BUILD SUCCESS 확인
2652
2653    5)src/config folder 생성
2654      -/src > right-click > New > Folder
2655        Folder name : config
2656
2657    6)Bean Configuration XML 작성
2658      -/src/config > right-click > New > Other > Spring > Spring Bean Configuration File
2659        File name : beans.xml > Next
2660          Check [beans - http://www.springframework.org/schema/beans]
2661          Check [http://www.springframework.org/schema/beans/spring-beans-4.3.xsd]
2662          Finish
2663
2664        <?xml version="1.0" encoding="UTF-8"?>
2665        <beans xmlns="http://www.springframework.org/schema/beans"
2666          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2667          xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans-4.3.xsd">
2668
2669          <bean id="hello" class="com.example.Hello">
2670            <property name="name" value="Spring" />
2671            <property name="printer" ref="printer" />
2672          </bean>
2673          <bean id="printer" class="com.example.StringPrinter" />
2674          <bean id="consolePrinter" class="com.example.ConsolePrinter" />
2675
2676        </beans>
2677
2678    7)DI Test 클래스 작성
2679      -/src/com.example > right-click > New > Package
2680        Package Name : test
2681      -/src/com.example/test/HelloBeanTest.java
2682
2683        package com.example.test;
2684
2685        import org.springframework.context.ApplicationContext;
2686        import org.springframework.context.support.GenericXmlApplicationContext;
2687
2688        import com.example.Hello;
2689        import com.example.Printer;
2690
2691        public class HelloBeanTest {
2692          public static void main(String [] args){
2693            //1. IoC Container 생성
2694            ApplicationContext context =
2695                new GenericXmlApplicationContext("config/beans.xml");
2696
2697            //2. Hello Beans 가져오기
2698            Hello hello = (Hello)context.getBean("hello");
2699            System.out.println(hello.sayHello());
2700            hello.print();
```

```
2701
2702            //3. SpringPrinter 가져오기
2703            Printer printer = (Printer)context.getBean("printer");
2704            System.out.println(printer.toString());
2705
2706            Hello hello2 = context.getBean("hello", Hello.class);
2707            hello2.print();
2708
2709            System.out.println(hello == hello2);  //Singleton Pattern
2710          }
2711        }
2712
2713        ------------------------------------------------------------------
2714        Hello Spring
2715        Hello Spring
2716        true
2717
2718    8)jUnit Library 설치
2719      -jUnit 4.12 버전을 pom.xml에 추가
2720
2721        <dependency>
2722            <groupId>junit</groupId>
2723            <artifactId>junit</artifactId>
2724            <version>4.12</version>
2725            <scope>test</scope>
2726        </dependency>
2727
2728      -pom.xml > right-click > Run As > Maven Install
2729
2730    9)jUnit을 사용한 DI 테스트 클래스(HelloBeanJunitTest.java) 작성
2731      -/src/com.example.test/HelloBeanTest.java 복사
2732      -/src/com.example.test/ 붙여넣고 이름변경 -> HelloBeanJunitTest.java
2733
2734        package com.example.test;
2735
2736        import org.junit.Before;
2737        import org.junit.Test;
2738        import org.springframework.context.ApplicationContext;
2739        import org.springframework.context.support.GenericXmlApplicationContext;
2740
2741        import com.example.Hello;
2742        import com.example.Printer;
2743
2744        import static org.junit.Assert.assertEquals;
2745        import static org.junit.Assert.assertSame;
2746
2747        public class HelloBeanJunitTest {
2748          ApplicationContext context;
2749
2750          @Before
2751          public void init(){
2752            //항상 먼저 ApplicationContext를 생성해야 하기 때문에
2753            //1. IoC Container 생성
2754            context = new GenericXmlApplicationContext("config/beans.xml");
```

```
2755          }
2756
2757          @Test
2758          public void test1(){
2759            //2. Hello Beans 가져오기
2760            Hello hello = (Hello)context.getBean("hello");
2761            assertEquals("Hello Spring", hello.sayHello());
2762            hello.print();
2763
2764            //3. SpringPrinter 가져오기
2765            Printer printer = (Printer)context.getBean("printer");
2766            assertEquals("Hello Spring", printer.toString());
2767          }
2768
2769          @Test
2770          public void test2(){
2771            Hello hello = (Hello)context.getBean("hello");
2772
2773            Hello hello2 = context.getBean("hello", Hello.class);
2774            assertSame(hello, hello2);
2775          }
2776        }
2777
2778      -right-click > Run As > Junit Test
2779      -결과 -> Junit View에 초록색 bar
2780
2781    10)Spring TestContext Framework
2782      -Spring-Test library 설치
2783       --pom.xml 수정
2784
2785        <dependency>
2786       <groupId>org.springframework</groupId>
2787       <artifactId>spring-test</artifactId>
2788       <version>4.3.9.RELEASE</version>
2789       <scope>test</scope>
2790      </dependency>
2791
2792      -pom.xml > right-click > Maven Install
2793
2794      -Spring-Test를 사용할 DI 테스트 클래스-HelloBeanJunitSpringTest.java 작성하기
2795        --/src/com.example.test/HelloBeanJunitTest.java 복사해서
2796        --/src/com.example.test/HelloBeanJunitSpringTest.java 로 붙여넣기
2797
2798          import org.junit.runner.RunWith;
2799          import org.springframework.beans.factory.annotation.Autowired;
2800          import org.springframework.test.context.ContextConfiguration;
2801          import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
2802          ...
2803          @RunWith(SpringJUnit4ClassRunner.class)
2804          @ContextConfiguration(locations="classpath:config/beans.xml")
2805          public class HelloBeanJunitSpringTest {
2806
2807            @Autowired
2808            ApplicationContext context;
```

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

11)src/com.example/StringPrinter.java 수정

```
package com.example;

import org.springframework.stereotype.Component;

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

12)src/com.example/ConsolePrinter.java 수정

```
package com.example;

import org.springframework.stereotype.Component;

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

13)/src/com.example/Hello.java 수정

```
package com.example;

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

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

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

  //setter 메소드가 필요 없음.

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

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

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

15)새로운 설정 파일 생성
-/src/config/beans.xml 새로 생성
-/src/config > right-click > New > Spring Bean Configuration File
File name : annos.xml
- Next > context - http://www.springframework.org/schema/context Check
- Check http://www.springframework.org/schema/context/spring-context-4.3.xsd
- Finish

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-4.3.xsd">

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

16)/src/com.example.test/HelloBeanJunitSpringTest.java 수정하기

```
package com.example.test;

import static org.junit.Assert.assertEquals;

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

import com.example.Hello;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations="classpath:config/beans.xml")
public class HelloBeanJunitSpringTest {
  @Autowired
  ApplicationContext context;

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

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