**[ 02 ] DI(Dependency Injection) 개념** – 기초단계에서 가장 중요한 개념

1. 스프링을 이용한 객체 생성과 조립(스프링은 객체 생성과 조립을 하는 컨테이너라 볼 수 있다)
   * 지난 예제는 스프링의 특징(사용법)이 적용되지 않은 프로젝트였습니다.
   * 이번 시간에는 스프링의 특징(사용법)을 적용하여 스프링과 좀 더 친근해 질 수 있는 시간을 갖도록 하겠습니다.
   * 스프링을 사용하지 않은 프로젝트

Calculation calculation = **new** Calculation();

calculation.setFirstNum(10);

calculation.setSecondNum(5);

calculation.add();

calculation.sub();

calculation.mult();

calculation.div();

* + 스프링을 적용한 프로젝트

<bean class=*"com.tj.ex.Calculation"* id=*"calculation"*>

<property name=*"firstNum"*>

<value>10</value>

</property>

<property name=*"secondNum"*>

<value>2</value>

</property>

</bean>

String resourceLocation = "classpath:applicationCTX.xml";

AbstractApplicationContext ctx = **new** GenericXmlApplicationContext(resourceLocation);

Calculation calculation = ctx.getBean("calculation", Calculation.**class**);

calculation.setFirstNum(10);

calculation.setSecondNum(5);

calculation.add();

calculation.sub();

calculation.mult();

calculation.div();

1. 스프링 설정 파일의 이해 ; 필드에 대한 setter함수들이 있어서 property의 value값을 줄 수 있다.

MyCalculator.java

**package** com.tj.ex;

**public** **class** MyCalculator {

**private** Calculator calculator;

**private** **int** firstNum;

**private** **int** secondNum;

**public** **void** add(){

calculator.addition(firstNum, secondNum);

}

**public** **void** sub(){

calculator.subtracion(firstNum, secondNum);

}

**public** **void** mul(){

calculator.multiplication(firstNum, secondNum);

}

**public** **void** div(){

calculator.division(firstNum, secondNum);

}

**public** **void** setCalculator(Calculator calculator) {

**this**.calculator = calculator;

}

**public** **void** setFirstNum(**int** firstNum) {

**this**.firstNum = firstNum;

}

**public** **void** setSecondNum(**int** secondNum) {

**this**.secondNum = secondNum;

}

}

Calculator.java

**package** com.tj.ex;

**public** **class** Calculator {

**public** **void** addition(**int** f, **int** s){

System.***out***.println("addition()");

**int** result = f + s;

System.***out***.println(f+"+"+s+"="+result);

}

**public** **void** subtracion(**int** f, **int** s){

System.***out***.println("addition()");

**int** result = f - s;

System.***out***.println(f+"-"+s+"="+result);

}

**public** **void** multiplication(**int** f, **int** s){

System.***out***.println("addition()");

**int** result = f \* s;

System.***out***.println(f+"\*"+s+"="+result);

}

**public** **void** division(**int** f, **int** s){

System.***out***.println("addition()");

**int** result = f / s;

System.***out***.println(f+"/"+s+"="+result);

}

}

<?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=*"calculator"* class=*"com.tj.ex.Calculator"*/>

<!-- 변수 설정 : calculator, myCalculator class는 패키지명을 포함한 fullname을 다 쓴다-->

<bean id=*"myCalculator"* class=*"com.tj.ex.MyCalculator"*>

<!-- 필드설정 -->

<property name=*"calculator"*>

<ref bean=*"calculator"*/>

</property>

<property name=*"firstNum"* value=*"10"*/>

<property name=*"secondNum"* value=*"2"*/>

</bean>

</beans>

MainClass.java

**package** com.tj.ex;

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

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

**public** **class** MainClass {

**public** **static** **void** main(String[] args) {

//MyCalculator myCalculator = new MyCalculator();

//myCalculator.setCalculator(new Calculator());

//myCalculator.setFirstNum(10);

//myCalculator.setSecondNum(2);

String resourceLocation = "classpath:applicationCTX.xml";

AbstractApplicationContext ctx = **new** GenericXmlApplicationContext(resourceLocation);

MyCalculator myCalculator = ctx.getBean("myCalculator", MyCalculator.**class**);

myCalculator.add();

myCalculator.sub();

myCalculator.mul();

myCalculator.div();

ctx.close();

}

}

1. 스프링 프로퍼티 설정

; 기초데이터, list타입, 다른 빈 객체 참조 방법 등을 살펴본다

BMICalculator.java

**package** com.tj.ex;

**public** **class** BMICalculator {

**private** **double** lowWeight;

**private** **double** normal;

**private** **double** overWeight;

**private** **double** obesity;

**public** **void** bmiCalculation(**double** weight, **double** height){

**double** h = height \* 0.01;

**double** result = weight / (h\*h);

System.***out***.println("BMI 지수 : "+(**int**)result);

**if**(result>obesity){

System.***out***.println("비만입니다");

}**else** **if**(result>overWeight){

System.***out***.println("과체중입니다");

}**else** **if**(result>normal){

System.***out***.println("정상입니다");

}**else** **if**(result>lowWeight){

System.***out***.println("저체중입니다");

}**else** {

System.***out***.println("심각하네요. 살 좀 찌세요");

}

}

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

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

**public** **void** setOverWeight(**double** overWeight) { **this**.overWeight = overWeight; }

**public** **void** setObesity(**double** obesity) { **this**.obesity = obesity; }

}

MyInfo.java

**package** com.tj.ex;

**import** java.util.ArrayList;

**public** **class** MyInfo {

**private** String name;

**private** **double** height;

**private** **double** weight;

**private** ArrayList<String> hobbies;

**private** BMICalculator bmiCalculator;

**public** **void** bmiCalculation(){

bmiCalculator.bmiCalculation(weight,height);

}

**public** **void** getInfo(){

System.***out***.println("이름 : "+name);

System.***out***.println("키 : "+height);

System.***out***.println("몸무게 : "+weight);

System.***out***.println("취미:"+ hobbies);

bmiCalculation();

}

**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** setHobbies(ArrayList<String> hobbies) {**this**.hobbies = hobbies;}

**public** **void** setBmiCalculator(BMICalculator bmiCalculator) {**this**.bmiCalculator = bmiCalculator;}

}

applicationCTX.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=*"bmiCalculator"* class=*"com.tj.ex.BMICalculator"*>

<property name=*"lowWeight"*>

<value>18.5</value>

</property>

<property name=*"normal"*>

<value>23</value>

</property>

<property name=*"overWeight"*>

<value>25</value>

</property>

<property name=*"obesity"*>

<value>30</value>

</property>

</bean>

<bean id=*"myInfo"* class=*"com.tj.ex.MyInfo"*>

<property name=*"name"*>

<value>홍길동</value> **<!--기본데이터-->**

</property>

<property name=*"height"*>

<value>164</value>

</property>

<property name=*"weight"*>

<value>52</value>

</property>

<property name=*"hobbys"*> **<!--List 타입데이터-->**

<list>

<value>수영</value>

<value>요리</value>

<value>독서</value>

</list>

</property>

<property name=*"bmiCalculator"*> **<!-- 다른 빈 객체 참조 -->**

<ref bean=*"bmiCalculator"*/>

</property>

</bean>

</beans>

MainClass.java

**package** com.tj.ex;

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

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

**public** **class** MainClass {

**public** **static** **void** main(String[] args) {

String resourceLocation = "classpath:applicationCTX.xml";

AbstractApplicationContext ctx = **new** GenericXmlApplicationContext(resourceLocation);

MyInfo myInfo = ctx.getBean("myInfo", MyInfo.**class**);

myInfo.getInfo();

ctx.close();

}

}

1. 스프링 컨테이너의 이해
   * 스프링 컨테이너를 생성하고 컴포넌트를 사용하는 방법에 대해서 알아보자

String resourceLocation = "classpath:applicationCTX.xml";

AbstractApplicationContext ctx = new GenericXmlApplicationContext(resourceLocation);

//스프링컨테이너 생성

MyInfo myInfo = ctx.getBean(“myInfo”, MyInfo.class);

// 스프링 컨테이너에서 컴포넌트 가져옴

myInfo.getInfo();

ctx.close();

// AbstractApplicationContext 추상클래스 위로는 ApplicationContext 인터페이스. 밑으로 상속받는 얘들중의 하나가 GenericXmlApplicationContext. 매개변수로 위치를 주면 IOC 컨테이너(스프링 컨테이너)가 생성. 즉 부품이 된 거임. 그 부품에서 “myInfo”라는 빈(이 안에도 bmiCalculator 주입됨)을 뽑아서 myInfo라는 변수에 넣음. 객체를 생성하고 조립하는 것을 Spring 컨테이너가 하고 있음