**[ 02 ] DI(Dependency Injection) 개념 및 활용** – 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 cal = **new** Calculation();

cal.setNum1(10);

cal.setNum2(5);

cal.add();

cal.sub();

cal.mul();

cal.div();

* + 스프링 부트는 기본적으로 자바 config를 기반으로 하기 때문에 DI 설정파일을 추가하는 기능이 제공되지 않음. 그래서 sping3을 플러그인 하여야 한다.
  + 프로젝트 오른쪽 마우스 -> Configure -> Convert to Maven Project
  + Pom.xml에 다음을 추가

<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.26</version>

<scope>provided</scope>

</dependency>

<!--https://mvnrepository.com/참조 -->

* + 외부 컨테이너에서 부품조립을 위한 클래스들

ApplicationContext

* + - AbstractApplicationContext
      * ClassPathXmlApplicationContext(GenericXmlApplicationContext)
      * AnnotationConfigApplicationContext

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

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

**public** **void** addition(**int** n1, **int** n2) {

System.***out***.println("더하기");

System.***out***.println(n1 + " + " + n2 + " = " + (n1+n2));

}

**public** **void** subtraction(**int** n1, **int** n2) {

System.***out***.println("빼기");

System.***out***.println(n1 + " - " + n2 + " = " + (n1-n2));

}

**public** **void** multiplication(**int** n1, **int** n2) {

System.***out***.println("곱하기");

System.***out***.println(n1 + " \* " + n2 + " = " + (n1\*n2));

}

**public** **void** division(**int** n1, **int** n2) {

System.***out***.println("나누기");

System.***out***.println(n1 + " / " + n2 + " = " + (n1/n2));

}

}

@Data

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

**private** Calculator calculator;

**private** **int** num1;

**private** **int** num2;

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

calculator.addition(num1, num2);

}

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

calculator.subtraction(num1, num2);

}

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

calculator.multiplication(num1, num2);

}

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

calculator.division(num1, num2);

}

}

**public** **class** TestMain {

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

// MyCalculator myCalculator = new MyCalculator();

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

// myCalculator.setNum1(10);

// myCalculator.setNum2(5);

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();

}

}

<?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.lec.ch02.Calculator"*/>

<bean id=*"myCalculator"* class=*"com.lec.ch02.MyCalculator"*>

<property name=*"calculator"*>

<ref bean=*"calculator"*/>

</property>

<property name=*"num1"* value=*"10"*/>

<property name=*"num2"* value=*"5"*/>

</bean>

</beans>

1. 스프링 프로퍼티 설정

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

BMICalculator.java

**import** lombok.Data;

@Data

**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 \* .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");

}

}

}

MyInfo.java

**import** lombok.Data;

@Data

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

**private** String name;

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

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

**private** ArrayList<String> hobbies;

**private** BMICalculator bmiCalculator;

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

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

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

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

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

bmiCalculator.bmiCalculation(weight,height);

}

}

applicationCTX2.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 name=*"bmiCalculatorMan"* class=*"com.lec.ch02.ex2\_bmi.BMICalculator"*>

<property name=*"lowWeight"* value=*"18.5"*/> <!-- 기초데이터 타입, String -->

<property name=*"normal"* value=*"23"*/>

<property name=*"overWeight"* value=*"25"*/>

<property name=*"obesity"* value=*"30"*/>

</bean>

<bean name=*"bmiCalculatorWoman"* class=*"com.lec.ch02.ex2\_bmi.BMICalculator"*>

<property name=*"lowWeight"* value=*"19"*/> <!-- 기초데이터 타입, String -->

<property name=*"normal"* value=*"23.5"*/>

<property name=*"overWeight"* value=*"25.5"*/>

<property name=*"obesity"* value=*"31"*/>

</bean>

<bean id=*"man"* class=*"com.lec.ch02.ex2\_bmi.MyInfo"*>

<property name=*"name"* value=*"홍길동"*/>

<property name=*"height"* value=*"175"*/>

<property name=*"weight"* value=*"66"*/>

<property name=*"hobby"*>

<list>

<value>수영</value><value>수면</value><value>개발</value>

</list>

</property>

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

</bean>

<bean id=*"woman"* class=*"com.lec.ch02.ex2\_bmi.MyInfo"*>

<property name=*"name"* value=*"김길순"*/>

<property name=*"height"* value=*"163"*/>

<property name=*"weight"* value=*"57"*/>

<property name=*"hobby"*>

<list>

<value>조깅</value><value>요가</value><value>개발</value>

</list>

</property>

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

</bean>

</beans>

**public** **class** MyInfoTestMain {

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

String resourceLocation = "applicationCTX2.xml";

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

//MyInfo man = (MyInfo) ctx.getBean("man");

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

man.getInfo();

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

woman.getInfo();

ctx.close();

}

}

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

String resourceLocation = "classpath:bmi/applicationCTX3.xml";

// 스프링 컨테이너 생성

AbstractApplicationContext ctx = new GenericXmlApplicationContext(resourceLocation);

// 스프링 컨테이너에서 컴포넌트 가져옴

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

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

1. 의존 관계
   * DI는 Dependency Injection의 약자로 우리말로 하면 ‘의존하는 객체를 주입하는 것‘ 입니다.
   * 의존주입은 setter를 통한 주입과 생성자를 통한 주입이 있습니다.
   * xml파일을 이용한 의존주입(아래의 예제는 studentInfo객체는 student객체에 의존하고 있다)

<bean id=*"studentInfo"* class=*"com.lec.ex.StudentInfo"*>

<!-- <constructor-arg ref="student1"/> -->

<constructor-arg>

<ref bean=*"student1"*/>

</constructor-arg>

</bean>

* + java파일을 이용한 의존주입

Student student2 = ctx.getBean(“student2”,Student.class);

studentInfo.setStudent(student2);

studentInfo.getStudentInfo();

1. 의 경우 예제 : 생성자 함수를 통해 주입하고 있다

**Student.java**

@Data

@NoArgsConstructor

@AllArgsConstructor

**public** **class** Student {

**private** String name;

**private** String age;

**private** String gradeNum;

**private** String classNum;

}

**StudentInfo.java**

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

@Data

@AllArgsConstructor

**public** **class** StudentInfo {

**private** Student student;

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

**if**(student!=**null**){

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

System.***out***.println("나이:"+student.getAge());

System.***out***.println("학년:"+student.getGradeNum());

System.***out***.println("반:"+student.getClassNum());

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

}**else** {

System.***out***.println("없어");

}

}

}

**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=*"student1"* class=*"com.lec.ch02.student.Student"*>

<constructor-arg >

<value>홍길동</value>

</constructor-arg>

<constructor-arg>

<value>10</value>

</constructor-arg>

<constructor-arg>

<value>3학년</value>

</constructor-arg>

<constructor-arg>

<value>20반</value>

</constructor-arg>

</bean>

<bean name=*"student2"* class=*"com.lec.ch02.student.Student"*>

<constructor-arg value=*"성춘향"*/>

<constructor-arg value=*"10"*/>

<constructor-arg value=*"3학년"*/>

<constructor-arg value=*"20반"*/>

</bean>

<bean id=*"studentInfo"* class=*"com.lec.ch02.student.StudentInfo"*>

<constructor-arg>

<ref bean=*"student1"*/>

</constructor-arg>

</bean>

</beans>

**MainClass.java**

**public** **class** TestMain {

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

String configLocation = "classpath:applicationStudent.xml";

AbstractApplicationContext ctx = **new** GenericXmlApplicationContext(configLocation);//스트링 설정파일을 파싱해서 컨테이너 생성

StudentInfo studentInfo = ctx.getBean("studentInfo", StudentInfo.**class**); //컴포넌트 가져온다

studentInfo.getInfo();

Student student2 = ctx.getBean("student2", Student.**class**); //컴포넌트 가져온다

studentInfo.setStudent(student2);

studentInfo.getInfo();

ctx.close();

}

}

Student2

Java코드이용

studentInfo 객체

생성자() 또는 setter

student1

Xml이용

Xml에 bean으로 주입해 쓰다가 맘에 안 들면 java 코드에서 다른 bean을 주입해서 쓸 수도 있다

1. DI 사용에 따른 장점
   * 아직은 스프링의 DI사용에 따른 장점을 많이 느끼지 못할 수 있습니다. 또는 DI를 사용하니 더욱 복잡하고 시간이 더 많이 소요된다고 생각 할 수 있습니다. (oop언어를 배울 때, class 나온 후 interface가 나온다. Interface는 구현도 없고 선언만 하는데 굳이 interface를 만들까 생각하지만 interface를 만들므로 타입을 통일시킬 수 있고, 강제로 오버라이드하게 해서 작업을 통일시킬 수 있다는 여러가지 장점이 있다. 이 장점은 규모가 큰 프로젝트를 할수록 느낄 수 있게 된다)
   * 사실 작은 규모의 프로젝트에서는 스프링의 DI사용을 하는 것 보다 일반적인 방법을 사용하여 개발하는 것이 더욱 빠르고, 개발에 따른 스트레스를 줄일 수 있습니다. 하지만 규모가 어느 정도 커지고, 추후 유지보수 업무가 발생시에는 DI를 이용한 개발의 장점을 느낄 수 있습니다.
   * 스프링 DI사용에 따른 장점을 예제를 통해 살펴 봅니다. **자바파일의 수정 없이** 스프링 설정 파일만을 수정하여 부품들을 생성/조립하고 있는 예제입니다

Vehicle.java

**public** **interface** Vehicle {

**public** **void** ride(String name);

}

Bicycle.java

**public** **class** Bicycle **implements** Vehicle {

**public** **void** ride(String name) {

System.***out***.println(name + "은(는) 자전거를 10km/h 속도로 탄다");

}

}

Car.java

**public** **class** Car **implements** Vehicle {

**public** **void** ride(String name) {

System.***out***.println(name + "은(는) 자동차를 70km/h 속도로 탄다");

}

}

Airplane.java

**package** com.ch.ex;

**public** **class** Airplane **implements** Vehicle {

@Override

**public** **void** ride(String name) {

System.***out***.println(name + "은(는) 비행기를 500km/h 속도로 탄다");

}

}

TestMain.java

**public** **class** TestMain {

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

String location = "classpath:applicationVehicleCTX.xml";

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

Vehicle vh = ctx.getBean("vh",Vehicle.**class**);

vh.ride("홍길동");

ctx.close();

}

}

applicationVehicleCTX.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="vh" class="com.lec.ch02.vehicle.Bicycle"/> -->

<!-- <bean id="vh" class="com.lec.ch02.vehicle.Car"/> -->

<bean id=*"vh"* class=*"com.lec.ch02.vehicle.Airplane"*/>

</beans>

DI장점(ex1의 진화버전) ex2.

VehicleImpl.java

**import** com.lec.ch02.vehicle.Vehicle;

**import** lombok.Data;

@Data

**public** **class** VehicleImpl **implements** Vehicle {

**private** String vehicle;

**private** **int** speed;

**public** **void** ride(String name) {

System.***out***.println(name + "은(는) " + vehicle + "을(를) "+speed+ "km/h 속도로 탄다");

}

}

<?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="vh" class="com.lec.ch02.vehicle.Bicycle"/> -->

<!-- <bean id="vh" class="com.lec.ch02.vehicle.Car"/> -->

<!-- <bean id="vh" class="com.lec.ch02.vehicle.Airplane"/> -->

<bean id=*"vh"* class=*"com.lec.ch02.vehicle2.VehicleImpl"*>

<property name=*"vehicle"* value=*"멋진차"*/>

<property name=*"speed"* value=*"70"*/>

</bean>

</beans>

**Quiz : 아래의 소스에서 기본의 자바구현을 수정하지 않고, 시험 과목을 조정할 경우 xml과 class를 추가하여 구현하세요**

아래의 소스 : 이름, 국어, 영어, 수학, 컴퓨터의 이름과 총점, 평균을 출력

(수정후 – 과목이 축소)

이름, 국어, 영어, 수학을 이용해서 이름과 총점, 평균을 출력

--------------------------

이름 : 홍길동

--------------------------

총점 : xxx

평균 : xx.x

---------------------------

**public** **interface** Exam {

**public** String getName();

**public** **int** total();

**public** **double** avg();

}

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

**public** **class** ExamImpl **implements** Exam {

**private** String name;

**private** **int** kor;

**private** **int** eng;

**private** **int** mat;

**private** **int** computer;

**public** **int** total() {

**return** kor+eng+mat+computer;

}

**public** **double** avg() {

**return** total()/4.0;

}

}

**public** **interface** ExamPrint {

**public** **void** print();

}

**import** lombok.AllArgsConstructor;

@AllArgsConstructor

**public** **class** ExamPrintImpl **implements** ExamPrint {

**private** Exam exam;

**public** **void** print() {

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

System.***out***.println("총점 : " + exam.total());

System.***out***.printf("평균 : %5.1f", exam.avg());

}

}

// 아래와 같이 구현하면 과목이 변경될 경우 자바소스를 수정

**public** **class** ExamPrintTestMain {

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

Exam exam = **new** ExamImpl("홍길동", 100,90,95,100);

ExamPrint examPrint = **new** ExamPrintImpl(exam);

examPrint.print();

}

}

**// DI 적용**

<?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=*"exam"* class=*"com.lec.ch02.ex5\_exam.ExamImpl"*>

<property name=*"name"* value=*"홍길동"*/>

<property name=*"kor"* value=*"100"*/>

<property name=*"eng"* value=*"90"*/>

<property name=*"mat"* value=*"95"*/>

<property name=*"computer"* value=*"100"*/>

</bean>

<bean id=*"examPrint"* class=*"com.lec.ch02.ex5\_exam.ExamPrintImpl"*>

<constructor-arg ref=*"exam"*/>

</bean>

</beans>

**public** **class** ExamPrintTestMain {

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

// Exam exam = new ExamImpl("홍길동", 100,99,98,95);

// ExamPrint examPrint = new ExamPrintImpl(exam);

String location = "classpath:applicationCTX5.xml";

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

ExamPrint examPrint = ctx.getBean("examPrint", ExamPrint.**class**);

examPrint.print();

}

}

**Quiz : 기존의 자바구현을 수정하지 않고, 시험 과목을 조정할 경우 xml과 class를 추가하여 구현하세요**

기존의 소스 : 이름, 국어, 영어, 수학, 컴퓨터의 이름과 총점, 평균을 출력

(수정후 내용)

1. .과목이 축소 : 이름, 국,영,수만. 컴퓨터 제외
2. 출력 폼 변경

--------------------------

이름 : 홍길동

--------------------------

총점 : xxx

평균 : xx.x

---------------------------

1. 제출파일 : ExamImpl2.java와 ExamPrintImpl2.java, 변경된 applicationCTX5.xml