[ 03 ] DI(Dependency Injection) 활용

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

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

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

<constructor-arg>

<ref bean=*"student1"*/>

</constructor-arg>

</bean>

* 1. 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**

**package** com.tj.ex;

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

**private** String name;

**private** String age;

**private** String gradeNum;

**private** String classNum;

**public** Student(String name, String age, String gradeNum, String classNum){

**this**.name = name;

**this**.age = age;

**this**.gradeNum = gradeNum;

**this**.classNum = classNum;

}

**public** String getName() {**return** name;}

**public** String getAge() {**return** age;}

**public** String getGradeNum() {**return** gradeNum;}

**public** String getClassNum() {**return** classNum;}

}

**StudentInfo.java**

**package** com.tj.ex;

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

**private** Student student;

**public** StudentInfo(Student student) {

**this**.student = student;

}

**public** **void** getStudent(){

**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("======================");

}

}

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

}

**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.tj.ex.Student"*>

<constructor-arg>

<value>홍길동</value>

</constructor-arg>

<constructor-arg>

<value>10살</value>

</constructor-arg>

<constructor-arg>

<value>3학년</value>

</constructor-arg>

<constructor-arg>

<value>20반</value>

</constructor-arg>

</bean>

<bean id=*"student2"* class=*"com.tj.ex.Student"*>

<constructor-arg value=*"홍길동"*/>

<constructor-arg value=*"9살"*/>

<constructor-arg value=*"2학년"*/>

<constructor-arg value=*"10반"*/>

</bean>

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

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

<constructor-arg>

<ref bean=*"student1"*/>

</constructor-arg>

</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 configLocation = "classpath:applicationCTX.xml";//스프링설정화일을 가져와서

AbstractApplicationContext ctx = **new** GenericXmlApplicationContext(configLocation);//파싱하고

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

studentInfo.getStudent();

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

studentInfo.setStudent(student2);

studentInfo.getStudent();

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
   1. 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)
   2. 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.
   3. 스프링 DI사용에 따른 장점을 예제를 통해 살펴 봅니다. **자바파일의 수정 없이** 스프링 설정 파일만을 수정하여 부품들을 생성/조립하고 있는 예제입니다

Vehicle.java

**package** com.ch.ex;

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

**void** ride(String name);

}

Bicycle.java

**package** com.ch.ex;

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

@Override

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

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

}

}

Car.java

**package** com.ch.ex;

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

@Override

**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");

}

}

MainClass.java

**package** com.ch.ex;

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

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

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

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

String location = "classpath:applicationCTX1.xml";

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

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

vh.ride("홍길동");

ctx.close();

}

}

applicationCTX1.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.ch.ex.Bicycle"/> -->

<!-- <bean id="vh" class="com.ch.ex.Car"/> -->

<bean id=*"vh"* class=*"com.ch.ex.Airplane"*/>

</beans>

DI장점(ex1의 진화버전) ex2.

VehicleImpl.java

**package** com.ch.ex2;

**import** com.ch.ex1.Vehicle;

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

**private** String vehicle;

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

@Override

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

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

}

**Setter 추가**

}

MainClass.java

**package** com.ch.ex;

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

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

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

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

String location = "classpath:applicationCTX2.xml";

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

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

vh.ride("홍길동");

ctx.close();

}

}

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 id=*"vh"* class=*"com.ch.ex2.VehicleImpl"*>

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

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

</bean>

</beans>