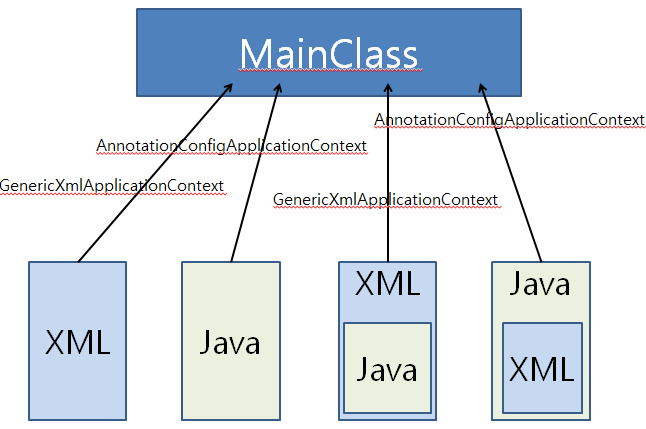
[ 03 ] DI(Dependency Injection) 설정방법



|  |  |  |
| --- | --- | --- |
| ApplicationContext종류 | XML | JAVA config |
| Non-Web | GenericXmlApplicationContext | AnnotationConfigApplicationContext |
| Web | XmlWebApplicationContext | AnnotationConfigWebApplicationContext |

1. XML 파일을 이용한 DI 설정방법 ; XML파일을 이용한 DI설정 방법은 그 동안 우리가 살펴본 방식 입니다. 이미 학습한 기본적인 사항들과 추가적인 새로운 사항들에 대해서 알아봅니다.

<bean id=”s1” class=”com.tj.ex.Student”>

<constructor-arg value=”홍”/> <!—생성자 설정. 기초데이터 -- >

<constructor-arg>

<list> <!—생성자 설정. 객체데이터 -- >

<value>수영</value>

<value>요리</value>

</list>

</constructor-arg>

<property name=”height”> <!—setter()설정 -- >

<value>173</value>

</property>

</bean>

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

<property name=”students”> <!—다른 빈 객체 설정-- >

<ref bean=”s1”/>

</property>

</bean>

<bean id=*"family"* class=*"com.tj.ex.Family"* c:papaName=*"홍아빠"* c:mamiName=*"홍엄마"* p:sisterName=*"홍딸램"*>

<property name=*"brotherName"* value=*"홍아들"*/>

</bean> <!—- c네임스페이스와 p네임스페이스를 쓰기 위해서는 아래에 두라인 추가 -->

xmlns:c=*"http://www.springframework.org/schema/c"*

xmlns:p=*"http://www.springframework.org/schema/p"*

String configLocation1 = “classpath:applicationCTX.xml”;

String configLocation2 = “classpath:applicationCTX1.xml”;

AbstractApplicationContext ctx = new GenericXmlApplicationContext(configLocation1, configLocation2); // 스프링컨테이너 생성. 스프링 설정화일이 다수인 경우

Student student1 = ctx.getBean(“s1”, Student.class); //스프링 컨테이너에서 객체 생성

(예제)

Student.java

@Data

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

**private** String name;

**private** **int** age;

**private** ArrayList<String> hobbies;

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

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

**public** Student(String name, **int** age, ArrayList<String> hobbies) {

**this**.name = name;

**this**.age = age;

**this**.hobbies = hobbies;

}

}

StudentInfo.java

@Data

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

**private** Student student;

}

Family.java

@Data

**public** **class** Family {

**private** String papaName;

**private** String mamiName;

**private** String sisterName;

**private** String brotherName;

**public** Family(String papaName, String mamiName){

**this**.papaName = papaName;

**this**.mamiName = mamiName;

}

}

method1\_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=*"student1"* class=*"com.lec.ch03.method1.Student"*>

<constructor-arg value=*"홍길동"*/>

<constructor-arg value=*"18"*/>

<constructor-arg>

<list>

<value>공기놀이</value>

<value>태권도</value>

</list>

</constructor-arg>

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

<property name=*"weight"* value=*"60"*/>

</bean>

<bean id=*"studentInfo"* class=*"com.lec.ch03.method1.StudentInfo"*>

<property name=*"student"* ref=*"student1"*/>

</bean>

</beans>

method1\_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"*

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.lec.ch03.method1.Student"* p:height=*"165.0"* p:weight=*"55"*>

<constructor-arg value=*"홍길숙"*/>

<constructor-arg value=*"22"*/>

<constructor-arg>

<list>

<value>윷놀이</value>

<value>댄스</value>

</list>

</constructor-arg>

</bean>

<bean id=*"family"* class=*"com.lec.ch03.method1.Family"* p:brotherName=*"홍아들"*>

<constructor-arg value=*"홍아빠"*/>

<constructor-arg value=*"김엄마"*/>

<property name=*"sisterName"* value=*"홍딸램"*/>

</bean>

</beans>

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

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

String resourceLocation1 = "method1\_applicationCTX1.xml";

String resourceLocation2 = "method1\_applicationCTX2.xml";

AbstractApplicationContext ctx = **new**

GenericXmlApplicationContext(resourceLocation1, resourceLocation2);

Student student1 = ctx.getBean("student1",Student.**class**);

System.***out***.println(student1.getName()+"의 취미 : "+student1.getHobbies());

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

Student student2 = studentInfo.getStudent();

System.***out***.println(student2.getName()+"의 취미 : "+student2.getHobbies());

**if**(student1.equals(student2)) {

System.***out***.println("student1 객체와 student2 객체는 같은 객체");

}**else** {

System.***out***.println("student1 객체와 student2 객체는 다른 객체");

}

Student student3 = ctx.getBean("student3", Student.**class**);

System.***out***.println(student3.getName()+"의 취미 : "+student3.getHobbies());

**if**(student1.equals(student3)) {

System.***out***.println("student1 객체와 student3 객체는 같은 객체");

}**else** {

System.***out***.println("student1 객체와 student3 객체는 다른 객체");

}

Family family = ctx.getBean("family", Family.**class**);

System.***out***.println(family);

ctx.close();

}

}

1. Java(어노테이션)를 이용한 DI 설정방법

예전(xml이용시)에는 AbstractApplicationContext를 통해서 GenericXmlApplicationContext를 이용해서 썼는데, 자바파일에서는 어노테이션을 이용하고 있다. 그래서 AnnotationConfigApplicationContext라는 클래스를 이용한다. 근데

AbstractApplicationContext

->GenericXmlApplicationContext

--> AnnotationConfigApplicationContext

결국 config 자바 파일로 내부적으로는 XML로 바뀌어서 다시 GenericXmlApplicationContext로 들어간다는 얘기. 사실 잘 안 쓰인다. 뭐 쓰일수도 있지만

pom.xml에 의존추가

<dependency>

<groupId>cglib</groupId>

<artifactId>cglib</artifactId>

<version>2.2.2</version>

</dependency>

ApplicationConfig.java

@Configuration

**public** **class** ApplicationConfig {

@Bean

**public** Student student1() {

ArrayList<String> hobbies = **new** ArrayList<String>();

hobbies.add("수영");hobbies.add("요가");

Student student = **new** Student("홍길동", 20, hobbies);

student.setHeight(160);

student.setWeight(52);

**return** student;

}

@Bean

**public** Student student2() {

ArrayList<String> hobbies = **new** ArrayList<String>();

hobbies.add("수영");hobbies.add("마라톤");

Student student = **new** Student("홍길순", 20, hobbies);

student.setHeight(160);

student.setWeight(52);

**return** student;

}

}

MainClass.java

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

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

AnnotationConfigApplicationContext ctx =

**new** AnnotationConfigApplicationContext(ApplicationConfig.**class**);

Student student1 = ctx.getBean("student1", Student.**class**);

System.***out***.println("이름 : "+student1.getName());

System.***out***.println("나이 : "+student1.getAge());

System.***out***.println("취미 : "+student1.getHobbies());

System.***out***.println("키 : "+student1.getHeight());

System.***out***.println("몸무게 : "+student1.getWeight());

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

System.***out***.println("이름 : "+student2.getName());

System.***out***.println("나이 : "+student2.getAge());

System.***out***.println("취미 : "+student2.getHobbies());

System.***out***.println("키 : "+student2.getHeight());

System.***out***.println("몸무게 : "+student2.getWeight());

ctx.close();

}

}

1. XML과 Java(어노테이션)를 같이 사용하여 스프링을 설정하고 컨테이너를 만들고 컴포넌트를 생성한다
   * 1. xml 파일에 Java파일을 포함시켜 사용하는 방법

ApplicationConfig.java

import java.util.ArrayList;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class ApplicationConfig {

@Bean

public Student student1(){

ArrayList<String> hobbys = new ArrayList<String>();

hobbys.add("수영");

hobbys.add("요리");

Student student = new Student("홍길동", 22, hobbys);

student.setHeight(180);

student.setWeight(80);

return 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"*

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-3.2.xsd"*>

<context:annotation-config />

<bean class=*"com.ch.ex.ApplicationConfig"* />

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

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

<constructor-arg value=*"30"*></constructor-arg>

<constructor-arg >

<list>

<value>마라톤</value>

<value>요리</value>

</list>

</constructor-arg>

<property name=*"height"* value=*"190"* />

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

</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) {

AbstractApplicationContext ctx = **new** GenericXmlApplicationContext("classpath:applicationCTX.xml");

Student student1 = ctx.getBean("student1",Student.**class**);

System.***out***.println("이름:"+student1.getName());

System.***out***.println("나이:"+student1.getAge());

System.***out***.println("취미:"+student1.getHobbys());

System.***out***.println("키:"+student1.getHeight());

System.***out***.println("몸무게:"+student1.getWeight());

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

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

System.***out***.println("이름:"+student2.getName());

System.***out***.println("나이:"+student2.getAge());

System.***out***.println("취미:"+student2.getHobbys());

System.***out***.println("키:"+student2.getHeight());

System.***out***.println("몸무게:"+student2.getWeight());

ctx.close();

}

}

* + 1. Java파일에 xml 파일을 포함시켜 사용하는 방법

위의 예제에 바뀐 부분만

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=*"student2"* class=*"com.tj.ex.Student"*>

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

<constructor-arg value=*"30"*></constructor-arg>

<constructor-arg >

<list>

<value>마라톤</value>

<value>요리</value>

</list>

</constructor-arg>

<property name=*"height"* value=*"190"* />

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

</bean>

</beans>

ApplicationConfig.java

package com.tj.ex;

import java.util.ArrayList;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.ImportResource;

@Configuration

@ImportResource("classpath:applicationCTX.xml")

public class ApplicationConfig {

@Bean

public Student student1(){

ArrayList<String> hobbys = new ArrayList<String>();

hobbys.add("수영");

hobbys.add("요리");

Student student = new Student("홍길동", 22, hobbys);

student.setHeight(180);

student.setWeight(80);

return student;

}

}

MainClass.java

**package** com.tj.ex;

**import** org.springframework.context.annotation.AnnotationConfigApplicationContext;

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

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

AnnotationConfigApplicationContext ctx = **new** AnnotationConfigApplicationContext(ApplicationConfig.**class**);

Student student1 = ctx.getBean("student1",Student.**class**);

System.***out***.println("이름:"+student1.getName());

System.***out***.println("나이:"+student1.getAge());

System.***out***.println("취미:"+student1.getHobbys());

System.***out***.println("키:"+student1.getHeight());

System.***out***.println("몸무게:"+student1.getWeight());

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

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

System.***out***.println("이름:"+student2.getName());

System.***out***.println("나이:"+student2.getAge());

System.***out***.println("취미:"+student2.getHobbys());

System.***out***.println("키:"+student2.getHeight());

System.***out***.println("몸무게:"+student2.getWeight());

ctx.close();

}

}

**@Autowired 어노테이션을 이용한 객체간 의존 자동 연결**

프로젝트의 규모가 조금만 커져도 한 개의 어플리케이션에서 생성하는 스프링 빈 객체는 수백개 이상으로 증가하게 되는데 이 경우 스프링 빈 간의 의존관계를 xml 설정이나 자바 기반 설정을 관리하는데 시간을 뺏길 수 있다. 또는 특정 타입의 빈 객체가 한 개밖에 존재하지 않는 경우가 많아서 의존 객체가 너무 뻔할 때가 있다. 만약 일일이 의존관계를 설정할 필요없이 자동으로 프로퍼티나 생성자 파라미터 값으로 동일 타입의 빈 객체를 전달해 주는 기능이 있다면 설정 코드가 많이 줄어들 것이다.

Student.java

**package** com.ch.ex;

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

**private** String name;

**private** String age;

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

**this**.name = name;

**this**.age = age;

}

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

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

}

StudentInfo.java

**package** com.ch.ex;

**import** org.springframework.beans.factory.annotation.Autowired;

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

**private** Student student;

@Autowired

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

**public** Student getStudent() {**return** 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"*

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-3.2.xsd"*>

<context:annotation-config/>

<bean id=*"student"* class=*"com.ch.ex.Student"*>

<constructor-arg value=*"홍길동"*/>

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

</bean>

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

</beans>

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 configLocation = "classpath:applicationCTX.xml";

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

StudentInfo stInfo = ctx.getBean("studentInfo",StudentInfo.**class**);

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

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

ctx.close();

}

}