[ 06 ] 외부파일을 이용한 설정

1. Environment 객체를 이용한 스프링 빈 설정
   1. Context 객체를 얻는다.

ConfigurableApplicationContext ctx = new GenericXmlApplicationContext();

* 1. Context 객체로부터 ctx.getEnvironment()를 하여 Environment 객체를 얻는다.

ConfigurableEnvironment env = ctx.getEnvironment();

* 1. Environment객체안에는 여러 가지 정보들(환경설정을 위한)이 있다. 이런 값들을 관리하기 위해 이런 정보들이 바로 들어있지 않고, PropertySource라고 하는 객체가 있고, PropertySource안에 정보가 있다. Environment 객체로부터 env.getPropertySources()를 하여 데이터를 관리해 주는 PropertySources 객체를 얻는다.

MutablePropertySources propertySources = env.getPropertySources();

* 1. 추가 : propertySources.addLast()

propertySources.addLast(new ResourcePropertySource("classpath:admin.properties"));

// 외부 설정파일을 ResourcePropertySource로 만들고 그것을 propertySources에 추가한다

// 내가 만든 설정 환경설정이 따라 붙게 추가한 것

* 1. 추출 : env.getProperty() 내가 필요한 정보가 있을 때까지 계속 검색하고 있으면 정보값을 리턴.

System.out.println(env.getProperty("admin.id"));

(예제)

admin.properties

admin.id=abcde

admin.pw=12345

**MainClass.java**

**package** com.tj.ex1env;

**import** java.io.IOException;

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

**import** org.springframework.core.env.ConfigurableEnvironment;

**import** org.springframework.core.env.MutablePropertySources;

**import** org.springframework.core.io.support.ResourcePropertySource;

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

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

GenericXmlApplicationContext ctx =

**new** GenericXmlApplicationContext();

ConfigurableEnvironment env = ctx.getEnvironment();

MutablePropertySources propertySource = env.getPropertySources();

**try** {

propertySource.addLast(

**new** ResourcePropertySource("classpath:admin.properties"));

// 외부 설정파일을 ResourcePropertySource로 만들고 그것을 propertySources에 추가한다

// 내가 만든 설정 환경설정이 따라 붙게 추가한 것

} **catch** (IOException e) {

e.printStackTrace();

}

System.***out***.println("admin.ID:"+env.getProperty("admin.id"));

System.***out***.println("admin.pw:"+env.getProperty("admin.pw"));

ctx.load("classpath:ex1envCTX.xml");

ctx.refresh();

AdminInfo adminInfo = ctx.getBean("adminInfo", AdminInfo.**class**);

System.***out***.println("adminInfo 빈이 생성시점에 외부설정파일 내용받음");

System.***out***.println("adminId : "+adminInfo.getAdminId());

System.***out***.println("adminPw : "+adminInfo.getAdminPw());

}

}

**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=*"adminInfo"* class=*"com.tj.ex1env.AdminInfo"*/>

</beans>

AdminInfo.java

**package** com.tj.ex1env;

**import** org.springframework.beans.factory.InitializingBean;

**import** org.springframework.context.EnvironmentAware;

**import** org.springframework.core.env.Environment;

**public** **class** AdminInfo **implements** EnvironmentAware, InitializingBean{

**private** Environment env;

**private** String adminId;

**private** String adminPw;

**public** String getAdminId() {**return** adminId;}

**public** **void** setAdminId(String adminId) {**this**.adminId = adminId;}

**public** String getAdminPw() {**return** adminPw;}

**public** **void** setAdminPw(String adminPw) {**this**.adminPw = adminPw;}

@Override

**public** **void** setEnvironment(Environment environment) {

// EnvironmentAware 땜시 오버라이드 - 최초로 실행

**this**.env = environment;

}

@Override

**public** **void** afterPropertiesSet() **throws** Exception {

// InitializingBean 땜시 오버라이드 - 두번째 초기화 작업 실행 부분

adminId = env.getProperty("admin.id");

adminPw = env.getProperty("admin.pw");

}

}

1. 프로퍼티 파일을 이용한 설정 ; Environment 객체를 사용하지 않고 프로퍼티 파일을 직접 이용하여 스프링 빈을 설정하는 방법

applicationCTX.xml 안에 admin.properties와 sub\_admin.properties

(예제)

admin.properties

admin.id=aaaa

admin.pw=1111

sub\_admin.properties

sub\_admin.id=bbbb

sub\_admin.pw=2222

**public** **class** AdminInfo {

**private** String adminId;

**private** String adminPw;

**private** String sub\_adminId;

**private** String sub\_adminPw;

**public** String getAdminId() {**return** adminId;}

**public** **void** setAdminId(String adminId) {**this**.adminId = adminId;}

**public** String getAdminPw() {**return** adminPw;}

**public** **void** setAdminPw(String adminPw) {**this**.adminPw = adminPw;}

**public** String getSub\_adminId() {**return** sub\_adminId;}

**public** **void** setSub\_adminId(String sub\_adminId) {**this**.sub\_adminId = sub\_adminId;}

**public** String getSub\_adminPw() {**return** sub\_adminPw;}

**public** **void** setSub\_adminPw(String sub\_adminPw) {**this**.sub\_adminPw = sub\_adminPw;}

}

**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:property-placeholder location=*"classpath:admin.properties, classpath:sub\_admin.properties"*/>

<bean id=*"adminInfo"* class=*"com.ch.ex.AdminInfo"*>

<property name=*"adminId"*>

<value>${admin.id}</value>

</property>

<property name=*"adminPw"*>

<value>${admin.pw}</value>

</property>

<property name=*"sub\_adminId"*>

<value>${sub\_admin.id}</value>

</property>

<property name=*"sub\_adminPw"*>

<value>${sub\_admin.pw}</value>

</property>

</bean>

</beans>

**public** **class** TestMain2 {

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

String configLocation = "classpath:applicationCTX2.xml";

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

AdminInfo adminInfo = ctx.getBean("adminInfo", AdminInfo.**class**);

System.***out***.println("adminId : "+adminInfo.getAdminId());

System.***out***.println("adminPW : "+adminInfo.getAdminPw());

System.***out***.println("subadminId : "+adminInfo.getSub\_adminId());

System.***out***.println("subadminPW : "+adminInfo.getSub\_adminPw());

ctx.close();

}

}

1. **프로파일(profile) 속성을 이용한 설정** ; 동일한 스프링 빈을 여러 개 만들어 놓고 상황(환경)에 따라서 적절한 스프링 빈을 사용할 수 있게 합니다. profile 속성을 사용하면 됩니다.
   1. Xml 설정 파일을 이용하는 경우

스프링 컨테이너에서 어떤 설정 파일을 사용할지 결정 함 setActiveProfiles(config)

XML설정파일1 profile = “dev” or XML설정파일2 profile = “run”

(예제)

ServerInfo.java

**package** com.ch.ex;

**public** **class** ServerInfo {

**private** String ipNum;

**private** String portNum;

**public** String getIpNum() {**return** ipNum;}

**public** **void** setIpNum(String ipNum) {**this**.ipNum = ipNum;}

**public** String getPortNum() {**return** portNum;}

**public** **void** setPortNum(String portNum) {**this**.portNum = portNum;}

}

applicationCTX\_dev.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"*

profile=*"dev"*>

<bean id=*"serverInfo"* class=*"com.ch.ex.ServerInfo"*>

<property name=*"ipNum"* value=*"192.168.2.10"*/>

<property name=*"portNum"* value=*"8181"*/>

</bean>

</beans>

applicationCTX\_run.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"*

profile=*"run"*>

<bean id=*"serverInfo"* class=*"com.ch.ex.ServerInfo"*>

<property name=*"ipNum"* value=*"127.0.0.1"*/>

<property name=*"portNum"* value=*"80"*/>

</bean>

</beans>

**MainClass.java**

**package** com.ch.ex;

**import** java.util.Scanner;

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

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

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

String config = **null**;

Scanner scanner = **new** Scanner(System.***in***);

String str = scanner.next();

**if**(str.equals("dev")){

config = "dev";

}**else** **if** (str.equals("run")){

config = "run";

}

scanner.close();

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

ctx.getEnvironment().setActiveProfiles(config);

ctx.load("applicationCTX\_dev.xml","applicationCTX\_run.xml");

ServerInfo info = ctx.getBean("serverInfo",ServerInfo.**class**);

System.***out***.println("ip:"+info.getIpNum());

System.***out***.println("ip:"+info.getPortNum());

}

}

* 1. Java 설정 파일을 이용하는 경우

스프링 컨테이너에서 어떤 설정 파일을 사용할지 결정 함 setActiveProfiles(config)

XML설정파일1 profile = “dev” or XML설정파일2 profile = “run”

(예제)

ServerInfo.java

**package** com.ch.ex;

**public** **class** ServerInfo {

**private** String ipNum;

**private** String portNum;

**public** String getIpNum() {**return** ipNum;}

**public** **void** setIpNum(String ipNum) {**this**.ipNum = ipNum;}

**public** String getPortNum() {**return** portNum;}

**public** **void** setPortNum(String portNum) {**this**.portNum = portNum;}

}

ApplicaionConfigDev.java

**package** com.ch.ex;

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

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

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

@Configuration

@Profile("dev")

**public** **class** ApplicaionConfigDev {

@Bean

**public** ServerInfo serverInfo(){

ServerInfo info = **new** ServerInfo();

info.setIpNum("127.0.0.1");

info.setPortNum("8181");

**return** info;

}

}

ApplicaionConfigRun.java

**package** com.ch.ex;

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

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

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

@Configuration

@Profile("run")

**public** **class** ApplicaionConfigRun {

@Bean

**public** ServerInfo serverInfo(){

ServerInfo info = **new** ServerInfo();

info.setIpNum("192.168.2.10");

info.setPortNum("80");

**return** info;

}

}

**MainClass.java**

**package** com.ch.ex;

**import** java.util.Scanner;

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

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

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

String config = **null**;

Scanner scanner = **new** Scanner(System.***in***);

String str = scanner.next();

**if**(str.equals("dev")){

config = "dev";

}**else** **if**(str.equals("run")){

config = "run";

}

scanner.close();

AnnotationConfigApplicationContext ctx = **new** AnnotationConfigApplicationContext();

ctx.getEnvironment().setActiveProfiles(config);

ctx.register(ApplicaionConfigDev.**class**, ApplicaionConfigRun.**class**);

ctx.refresh();

ServerInfo info = ctx.getBean("serverInfo",ServerInfo.**class**);

System.***out***.println("IP : "+info.getIpNum());

System.***out***.println("port:"+info.getPortNum());

ctx.close();

}

}