<http://blog.csdn.net/flygoa/article/details/58075398>

配置文件配置

直接配置

在src/main/resources下添加配置文件application.properties   
例如修改端口号

#端口号

server.port=8089

分环境配置

在src/main/resources下添加，application-pro.properties，application-dev.properties和application.properties三个文件   
application.propertie

spring.profiles.active=dev

application-pro.properties

#端口号

server.port=80

#自定义端口号读取

my.name=pzr.dev

application-dev.properties

#端口号

server.port=8089

#自定义端口号读取

my.name=pzr.pro

当application.propertie设置spring.profiles.active=dev时，则说明是指定使用application-dev.properties文件进行配置

配置文件参数读取

注解方式读取

1. @PropertySource配置文件路径设置，在类上添加注解，如果在默认路径下可以不添加该注解   
   classpath:config/my.properties指的是src/main/resources目录下config目录下的my.properties文件

多配置文件引用，若取两个配置文件中有相同属性名的值，则取值为最后一个配置文件中的值

@PropertySource({"classpath:config/my.properties","classpath:config/config.properties"})

public class TestController

1. @Value属性名，在属性名上添加该注解

@Value("${my.name}")

private String myName;

对象映射方式读取

1. 首先建立对象与配置文件映射关系
2. 方法中使用自动注入方式，将对象注入，调用get方法获取属性值
3. 注意：新版本的@ConfigurationProperties没有了location属性，使用@PropertySource来指定配置文件位置
4. prefix=”obj”指的是配置文件中的前缀，如obj.name，在定义对象属性名时为private String name;
5. 读取配置文件中的集合时，使用List来接收数据，但List必须先实例化

TestController.java

package com.gyqc.ms.controller;

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

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.PropertySource;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseBody;

import com.gyqc.ms.entity.config.ObjectProperties;

/\*\*

\* 测试接口

\* @author pzr

\*

\*/

@Controller

@RequestMapping("/testController")

@PropertySource({"classpath:config/my.properties","classpath:config/config.properties"})

public class TestController {

//在application.properties中的文件，直接使用@Value读取即可，applicarion的读取优先级最高

@Value("${my.name}")

private String myName;

//如果多个文件有重复的名称的属性话，最后一个文件中的属性生效

@Value("${my.name1}")

private String myName1;

@Value("${my.name2}")

private String myName2;

@Autowired

ObjectProperties objectProperties;

@RequestMapping("/test")

@ResponseBody

String test() {

return "myName:"+myName+" myName1:"+myName1+" name:"+myName2;

}

/\*\*

\* 使用对象映射的方式读取配置文件

\* @return

\*/

@RequestMapping("/test1")

@ResponseBody

String test1(){

StringBuffer sb = new StringBuffer();

sb.append(" name:");

sb.append(objectProperties.getName());

sb.append(" age:");

sb.append(objectProperties.getAge());

sb.append(" className:");

for(String str : objectProperties.getClassName()){

sb.append(str);

sb.append(",");

}

return sb.toString();

}

}

ObjectProperties.java

package com.gyqc.ms.entity.config;

import java.util.ArrayList;

import java.util.List;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.context.annotation.PropertySource;

import org.springframework.stereotype.Component;

/\*\*

\* 配置文件映射对象

\* @author pzr

\*

\*/

@Component

@PropertySource("classpath:config/object.properties")

@ConfigurationProperties(prefix="obj")

public class ObjectProperties {

private String name;

private String age;

//集合必须初始化，如果找不到就是空集合，会报错

private List<String> className = new ArrayList<String>();

public List<String> getClassName() {

return className;

}

public void setClassName(List<String> className) {

this.className = className;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAge() {

return age;

}

public void setAge(String age) {

this.age = age;

}

}

object.properties

#自定义属性读取

obj.name=obj.name

obj.age=obj.age

obj.className[0]=obj.className[0]

obj.className[1]=obj.className[1]