

## @Value（“#{}“）和 @Value（”${}“）

作用，为标注的字段赋值。

@Value(“${ 配置文件}”) 将外部配置文件里的properties赋值给字段

@Value(“#{将实体类的属性赋值}”) object的属性赋值给字段。

## Spring加载配置文件

1、config/application.properties（项目根目录中config目录下）

2、config/application.yml

3、application.properties（项目根目录下）

4、application.yml

5、resources/config/application.properties（项目resources目录中config目录下）

6、resources/config/application.yml

7、resources/application.properties（项目的resources目录下）

8、resources/application.yml

Config>根目录

项目>类路径

.propertiriy>yml

如果是对于同一个属性进行设置，优先级高的，也就是先读取的，会将优先级低的覆盖掉

Application.yml application.propertities 这是默认的命名格式。Application-xxx.propertities或者application-xxx.yml

**类路径：就是存放.class文件的路径**，在target项目下

## Profile

背景：定义了多套application-xxx.yml或者application-xxx.propertities同一时间只用一套能生效。

同时Javabean也可以点定义profile，只有在相应的profile被激活的时候，相应的bean才会被创建。没有被指定profile的bean始终都会被创建。[Spring3.0@Profile](mailto:Spring3.0@Profile)只能作用在类级别上，[Spring3.1后@Profile](mailto:Spring3.1后@Profile)可以作用与方法上。

设置的方法spring.profile.active和spring.profile.default(其中spring.profile.active的优先级高点，如果两个都设置了，active会把default覆盖，如果都没设置，那标注了@Profile的bean就不会实例化，相应的配置文件也不会加载，生效)

指定激活的profile方法，在application.yml或者是application.propertities文件中设置属性

Spring:

Profiles:

active: dev

在application.propertities文件中使用spring.profiles.active=dev

如果有冲突，根据优先级进行覆盖。

在idea中也可以进行设置。