**MAVEN 属性定义与使用**

Maven内置了三大特性：属性、Profile和资源过滤来支持构建的灵活性。

**MAVEN属性**

事实上有六种类型的Maven属性：

* 内置属性：主要有两个常用内置属性——${basedir}表示项目根目录，即包含pom.xml文件的目录;${version}表示项目版本。
* POM属性：pom中对应元素的值。例如${project.artifactId}对应了<project><artifactId>元素的值。具体有哪些POM属性可以用，可以查看本页末的附件——超级POM
* 自定义属性：在pom中<properties>元素下自定义的Maven属性。例如：

<project>

<properties>

<my.prop>hello</my.prop>

</properties>

</project>

* Settings属性：与POM属性同理。如${settings.localRepository}指向用户本地仓库的地址。
* Java系统属性：所有Java系统属性都可以使用Maven属性引用，例如${user.home}指向了用户目录。可以通过命令行mvn help:system查看所有的Java系统属性
* 环境变量属性：所有环境变量都可以使用以env.开头的Maven属性引用。例如${env.JAVA\_HOME}指代了JAVA\_HOME环境变量的值。也可以通过命令行mvn help:system查看所有环境变量。

**资源过滤**

maven的**properties filter**功能可以帮你自动替换配置文件中以$｛｝包裹的变量。

为了方便构建不同的环境，我们通常将不同的配置以properties形式配置在pom 中。

默认情况下，Maven属性只有在POM中才会被解析。资源过滤就是指让Maven属性在资源文件(src/main/resources、src/test/resources)中也能被解析。

在POM中添加下面的配置便可以开启资源过滤

<build>

<resources>

<resource>

<directory>${project.basedir}/src/main/resources</directory>

<filtering>true</filtering>

</resource>

</resources>

<testResources>

<testResource>

<directory>${project.basedir}/src/test/resources</directory>

<filtering>true</filtering>

</testResource>

</testResources>

</build>

Maven除了可以对主资源目录、测试资源目录过滤外，还能对Web项目的资源目录(如css、js目录)进行过滤。这时需要对maven-war-plugin插件进行配置

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<version>2.1-beta-1</version>

<configuration>

<webResources>

<resource>

<filtering>true</filtering>

<directory>src/main/webapp</directory>

<includes>

<include>\*\*/\*.css</include>

<include>\*\*/\*.js</include>

</includes>

</resource>

</webResources>

</configuration>

</plugin>

**MAVEN PROFILE**

每个Profile可以看作是POM的一部分配置，我们可以根据不同的环境应用不同的Profile，从而达到不同环境使用不同的POM配置的目的。

profile可以声明在以下这三个文件中：

* pom.xml：很显然，这里声明的profile只对当前项目有效
* 用户settings.xml：.m2/settings.xml中的profile对该用户的Maven项目有效
* 全局settings.xml：conf/settings.xml，对本机上所有Maven项目有效

非常值得注意的一点是，profile在pom.xml中可声明的元素在settings.xml中可声明的元素是不一样的：

* profile在pom.xml中可声明的元素：

<project>

<repositories></repositories>

<pluginRepositories></pluginRepositories>

<distributionManagement></distributionManagement>

<dependencies></dependencies>

<dependencyManagement></dependencyManagement>

<modules></modules>

<properties></properties>

<reporting></reporting>

<build>

<plugins></plugins>

<defaultGoal></defaultGoal>

<resources></resources>

<testResources></testResources>

<finalName></finalName>

</build>

</project>

profile在settings.xml中可声明的元素：

<project>

<repositories></repositories>

<pluginRepositories></pluginRepositories>

<properties></properties>

</project>

**profile 配置实例：**

**<project>**

**<modelVersion>4.0.0</modelVersion>**

**<groupId>cc.mzone</groupId>**

**<artifactId>myjar</artifactId>**

**<version>0.1</version>**

**<packaging>jar</packaging>**

**<build>**

**<resources>**

**<resource>**

**<directory>src/main/resources</directory>**

**<includes>**

**<include>\*.\*</include>**

**</includes>**

**<filtering>true</filtering>**

**</resource>**

**</resources>**

**</build>**

**<properties>**

**<jdbc.url>jdbc:mysql://localhost:3306/abc</jdbc.url>**

**<jdbc.username>root</jdbc.username>**

**<jdbc.password>root</jdbc.password>**

**</properties>**

**<profiles>**

**<profile>**

**<id>product</id>**

**<properties>**

**<jdbc.url>jdbc:mysql://localhost:3306/abc123</jdbc.url>**

**<jdbc.username>rootuser</jdbc.username>**

**<jdbc.password>rootpwd</jdbc.password>**

**</properties>**

**</profile>**

**</profiles>**

**</project>**

**这里我们在pom文件中定义了数据库的相关配置，同时定义了一个profile，其id为product，同时在这个profile中也定义了数据库的相关配置。这样我们使用mvn package命令时就可以使用默认的jdbc设置，当我们使用mvn package -P product时maven就会自动使用id为product的profile中的数据库配置，这个是maven读取属性配置文件的覆盖。**

**激活PROFILE**

有多种激活Profile的方式：

1. 命令行方式激活，如有两个profile id为devx和devy的profile：

mvn clean install  -Pdevx,devy

1. settings文件显式激活
2. <settings>
3. ...
4. <activeProfiles>
5. <activeProfile>devx</activeProfile>
6. <activeProfile>devy</activeProfile>
7. </activeProfiles>
8. ...

</settings>

1. 系统属性激活，用户可以配置当某系统属性存在或其值等于期望值时激活profile，如：
2. <profiles>
3. <profile>
4. <activation>
5. <property>
6. <name>actProp</name>
7. <value>x</value>
8. </property>
9. </activation>
10. </profile>

</profiles>

不要忘了，可以在命令行声明系统属性。如：

* 1. mvn clean install -DactProp=x

这其实也是一种从命令行激活profile的方法，而且多个profile完全可以使用同一个系统属性来激活。别忘了，系统属性可以通过mvn help:system来查看操作系统环境激活，

<profiles>

<profile>

<activation>

<os>

<name>Windows XP</name>

<family>Windows</family>

<arch>x86</arch>

<version>5.1.2600</version>

</os>

</activation>

</profile>

</profiles>

这里的family值包括Window、UNIX和Mac等，而其他几项对应系统属性的os.name、os.arch、os.version

5 . 文件存在与否激活

Maven能根5. 据项目中某个文件存在与否来决定是否激活profile

<profiles>

<profile>

<activation>

<file>

<missing>x.properties</missing>

<exists>y.properties</exists>

</file>

</activation>

</profile>

</profiles>

**Notice：插件maven-help-plugin提供了一个目标帮助用户了解当前激活的profile：**

1. **mvn help:active-profiles**

**另外还有一个目标来列出当前所有的profile：**

1. **mvn help:all-profiles**