# settings.xml文件结构详解

1. <https://blog.csdn.net/taiyangdao/article/details/52274263>
2. <https://github.com/dunwu/java-tutorial/blob/master/docs/javatool/build/maven/maven-settings.md>

## 一、settings.xml文件中顶级元素概览：

settings.xml文件可以包含的顶级节点如下：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <settings xsi:schemaLocation="http://maven.apache.org/  SETTINGS/1.1.0 http://maven.apache.org/xsd/settings-1.1.0.xsd" xmlns="http://maven.apache.org/SETTINGS/1.1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  <localRepository/>  <interactiveMode/>  <usePluginRegistry/>  <offline/>  <servers/>  <mirrors/>  <proxies/>  <profiles/>  <activeProfiles/>  <pluginGroups/>  </settings> |

## 二、localRepositories：

localRepositories表示本地仓库的路径，默认值为 ${user.home}/.m2/repository。

## 三、interactiveMode：

表示Maven执行过程中是否需要和用户交互以获取输入。

如果Maven需要和用户交互来获得输入，则设置成true，反之为false。默认为true。

## 四、usePluginRegistry：

是否使用plugin-registry.xml文件管理Maven插件的版本，默认为false。

该文件为用户提供了选择，可以使用指定版本的Maven插件，而非最新版本的Maven插件。

该文件是从Maven 2开始出现的，但是事实上更常用的是在POM中配置Maven插件的版本等参数，所以usePluginRegistry参数往往为false。

另外，与settings.xml文件类似，plugin-registry.xml文件也有全局和用户之分。

## 五、offline：

是否支持离线构建系统，默认是false。如果build服务器由于网络或者安全等原因不能连接远程库，则可将该参数设置为true。

## 六、pluginGroup：

给出Maven插件所在的groupId，一个可能的groupId使用一个<pluginGroup>给出。该参数只是为了简化执行Maven时的参数，如为了执行如下命令：

|  |
| --- |
| mvn org.mortbay.jetty:jetty-maven-plugin:run |

如果爱settings.xml文件中配置了如下<pluginGroup>：

|  |
| --- |
| <pluginGroups>  <pluginGroup>org.mortbay.netty</pluginGroup>  </pluginGroups> |

则可以直接执行如下命令：

|  |
| --- |
| mvn jetty:run |

【注】默认情况下，该列表包含了org.apache.maven.plugins和org.codehaus.mojo

## 七、servers：

一般，远程仓库的下载路径和部署路径是在pom.xml文件中的repositories 和 distributionManagement 元素中定义的。

然而，一般类似于仓库用户名、密码（有些仓库访问是需要安全认证机制的）等信息不应该在pom.xml文件中配置，这些信息可以配置在 settings.xml 中。

|  |
| --- |
| <settings xmlns=“<http://maven.apache.org/SETTINGs/1.0.0>”  xmlns:xsi=“<http://www.w3.org/2001/XMLSchema-instance>”  xsi:schemaLocation=“http://maven.apache.org/SETTINGS/1.0.0  <https://maven.apache.org/xsd/settings-1.0.0.xsd>”  <!-- 配置服务器端的一些设置,一些设置如安全证书不应该和pom.xml一起分发,这种类型的信息应该存在于构建服务器上的settings.xml文件中 -->  <servers>  <!-- 服务器元素包含配置服务器时需要的信息 -->  <server>  <!-- 这是server的id(注意不是用户登录的id),该id与distributionManagement中的repository元素中的id相匹配 -->  <id>server001</id>  <!-- 鉴权用户名。鉴权用户名和鉴权密码表示服务器认证所需要的登录名和密码 -->  <username>my\_login</username>  <!-- 鉴权密码。鉴权用户名和鉴权密码表示服务器认证所需要的登录名和密码。密码加密功能已被添加到2.1.0+。详情请访问密码加密页面。 -->  <password>my\_password</password>  <!-- 鉴权使用的私钥位置。和前两个元素类似,私钥位置和私钥密码指定了一个私钥的路径(默认是${user.home}/.ssh/id\_dsa)以及需要的话,一个密语。将来passphrase和password元素可能被提取到外部,但目前它们必须在settings.xml文件以纯文本的形式声明。 -->  <privateKey>${user.home}/.ssh/id\_dsa</privateKey>  <!—鉴权时使用的私钥密码 -->  <passphrase>some\_passphrase</passphrase>  <!-- 文件被创建时的权限。如果在部署的时候会创建一个仓库文件或者目录,这时候就可以使用权限(permission)。这两个元素合法的值是一个三位数字,其对应了unix文件系统的权限,如664或者775 -->  <filePermission>664</filePermission>  <!—目录被创建时的权限 -->  <directoryPermission></directoryPermission>  </server>  </servers>  </settings> |

## 八、mirrors：

为仓库列表配置的下载镜像列表。

|  |
| --- |
| <mirrors>  <!-- 给定仓库的下载镜像 -->  <mirror>  <!-- 该镜像的唯一标识。id用来区分不同的mirror元素 -->  <id>planetmirror.com</id>  <!-- 镜像名称 -->  <name>PlanetMirror Australia</name>  <!-- 该镜像的URL。构建系统会优先考虑该URL,而非默认的服务器URL -->  <url>[http://downloads.planetmirror.com/pub/maven2</url](http://downloads.planetmirror.com/pub/maven2%3c/url)>  <!-- 要镜像的远程库。 -->  <mirrorOf>central</mirrorOf>  </mirror>  </mirrors> |

## 九、proxies：

用来配置不同的代理

|  |
| --- |
| <proxies>  <!--代理元素包含配置代理时需要的信息 -->  <proxy>  <!--代理的唯一定义符，用来区分不同的代理元素。 -->  <id>myproxy</id>  <!--该代理是否是激活的那个。true则激活代理。当我们声明了一组代理，而某个时候只需要激活一个代理的时候，该元素就可以派上用处。 -->  <active>true</active>  <!--代理的协议。 协议://主机名:端口，分隔成离散的元素以方便配置。 -->  <protocol>http</protocol>  <!--代理的主机名。协议://主机名:端口，分隔成离散的元素以方便配置。 -->  <host>proxy.somewhere.com</host>  <!--代理的端口。协议://主机名:端口，分隔成离散的元素以方便配置。 -->  <port>8080</port>  <!--代理的用户名，用户名和密码表示代理服务器认证的登录名和密码。 -->  <username>proxyuser</username>  <!--代理的密码，用户名和密码表示代理服务器认证的登录名和密码。 -->  <password>somepassword</password>  <!--不该被代理的主机名列表。该列表的分隔符由代理服务器指定；例子中使用了竖线分隔符，使用逗号分隔也很常见。 -->  <nonProxyHosts>\*.google.com|ibiblio.org</nonProxyHosts>  </proxy>  </proxies> |

## 十、profiles：

根据环境参数来调整构件配置的列表

settings.xml中的profile元素是pom.xml中profile元素的裁剪版本。

它包含了id、activation、repositories、pluginRepositories和properties元素。这里的profile只包含这五个元素是因为这里只关心构建系统这个整体，而非单独的项目对象模型设置。如果一个settings.xml中的profile被激活，它的值会覆盖任何其他定义在pom.xml中带有相同id的profile。

|  |
| --- |
| <profiles>  <profile>  <!-- profile的唯一标识 -->  <id>test</id>  <!-- 自动触发profile的条件逻辑 -->  <activation />  <!-- 扩展属性列表 -->  <properties />  <!-- 远程仓库列表 -->  <repositories />  <!-- 插件仓库列表 -->  <pluginRepositories />  </profile>  </profiles> |

### 1、activation：

作用：自动触发profile的条件逻辑。

如pom.xml中的profile一样，profile的作用在于它能够在某些特定的环境中自动使用某些特定的值；这些环境通过activation元素指定。

activation元素并不是激活profile的唯一方式。settings.xml文件中的activeProfile元素包含profile的id。profile也可以通过在命令行，使用-P标记和逗号分隔的列表来显示激活。

|  |
| --- |
| <activation>  <!--profile默认是否激活的标识 -->  <activeByDefault>false</activeByDefault>  <!--当匹配的jdk被检测到，profile被激活。例如，1.4激活JDK1.4，1.4.0\_2，而!1.4激活所有版本不是以1.4开头的JDK。 -->  <jdk>1.5</jdk>  <!--当匹配的操作系统属性被检测到，profile被激活。os元素可以定义一些操作系统相关的属性。 -->  <os>  <!--激活profile的操作系统的名字 -->  <name>Windows XP</name>  <!--激活profile的操作系统所属家族(如 'windows') -->  <family>Windows</family>  <!--激活profile的操作系统体系结构 -->  <arch>x86</arch>  <!--激活profile的操作系统版本 -->  <version>5.1.2600</version>  </os>  <!--如果Maven检测到某一个属性（其值可以在POM中通过${name}引用），其拥有对应的name = 值，Profile就会被激活。如果值字段是空的，那么存在属性名称字段就会激活profile，否则按区分大小写方式匹配属性值字段 -->  <property>  <!--激活profile的属性的名称 -->  <name>mavenVersion</name>  <!--激活profile的属性的值 -->  <value>2.0.3</value>  </property>  <!--提供一个文件名，通过检测该文件的存在或不存在来激活profile。missing检查文件是否存在，如果不存在则激活profile。另一方面，exists则会检查文件是否存在，如果存在则激活profile。 -->  <file>  <!--如果指定的文件存在，则激活profile。 -->  <exists>${basedir}/file2.properties</exists>  <!--如果指定的文件不存在，则激活profile。 -->  <missing>${basedir}/file1.properties</missing>  </file>  </activation> |

### 2、properties：

作用：对应profile的拓展属性表

maven属性和ant中的属性一样，可以用来存放一些值。这些值可以在pom.xml中的

任何地方会用标记${x}来使用，这里x是指属性的名称。

属性有5种不同的形式，并且都能够在settings.xml文件中访问

|  |
| --- |
| <!--  1. env.X: 在一个变量前加上"env."的前缀，会返回一个shell环境变量。例如,"env.PATH"指代了$path环境变量（在Windows上是%PATH%）。  2. project.x：指代了POM中对应的元素值。例如: <project><version>1.0</version></project>通过${project.version}获得version的值。  3. settings.x: 指代了settings.xml中对应元素的值。例如：<settings><offline>false</offline></settings>通过 ${settings.offline}获得offline的值。  4. Java System Properties: 所有可通过java.lang.System.getProperties()访问的属性都能在POM中使用该形式访问，例如 ${java.home}。  5. x: 在<properties/>元素中，或者外部文件中设置，以${someVar}的形式使用。  -->  <properties>  <user.install>${user.home}/our-project</user.install>  </properties> |

### 3、repositories：

作用：远程仓库列表

|  |
| --- |
| <repositories>  <!--包含需要连接到远程仓库的信息 -->  <repository>  <!--远程仓库唯一标识 -->  <id>codehausSnapshots</id>  <!--远程仓库名称 -->  <name>Codehaus Snapshots</name>  <!--如何处理远程仓库里发布版本的下载 -->  <releases>  <!--true或者false表示该仓库是否为下载某种类型构件（发布版，快照版）开启。 -->  <enabled>false</enabled>  <!--该元素指定更新发生的频率。Maven会比较本地POM和远程POM的时间戳。这里的选项是：always（一直），daily（默认，每日），interval：X（这里X是以分钟为单位的时间间隔），或者never（从不）。 -->  <updatePolicy>always</updatePolicy>  <!--当Maven验证构件校验文件失败时该怎么做-ignore（忽略），fail（失败），或者warn（警告）。 -->  <checksumPolicy>warn</checksumPolicy>  </releases>  <!--如何处理远程仓库里快照版本的下载。有了releases和snapshots这两组配置，POM就可以在每个单独的仓库中，为每种类型的构件采取不同的策略。例如，可能有人会决定只为开发目的开启对快照版本下载的支持。参见repositories/repository/releases元素 -->  <snapshots>  <enabled />  <updatePolicy />  <checksumPolicy />  </snapshots>  <!--远程仓库URL，按protocol://hostname/path形式 -->  <url>http://snapshots.maven.codehaus.org/maven2</url>  <!--用于定位和排序构件的仓库布局类型-可以是default（默认）或者legacy（遗留）。Maven 2为其仓库提供了一个默认的布局；然而，Maven 1.x有一种不同的布局。我们可以使用该元素指定布局是default（默认）还是legacy（遗留）。 -->  <layout>default</layout>  </repository>  </repositories> |

### 4、pluginRepositories：

插件的远程仓库

## 十一、activeProfiles：

作用：手动激活profiles的列表，按照profile被应用的顺序定义activeProfile。

该元素包含一组activeProfile元素，每个activeProfile都含有一个profile id。任何在activeProfile中定义的profile id，不论环境设置如何，其对应的profile都会被激活。如果没有匹配的profile，则什么都不会发生。

例如，env\_test是一个activeProfile，则在pom.xml（或者profile.xml或者settings.xml）中对应的profile会被激活。如果运行过程中找不到这样一个profile，Maven则会像往常一样运行。

|  |
| --- |
| <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0  https://maven.apache.org/xsd/settings-1.0.0.xsd">  ...  <activeProfiles>  <!-- 要激活的profile id -->  <activeProfile>env-test</activeProfile>  </activeProfiles>  ...  </settings> |