< 2017年3月 >

日	一	二	三	四	五	六
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

昵称：Yakov
园龄：5年5个月
粉丝：24
关注：2
+加关注

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

最新随笔

1. NoSQL的一些碎碎念
2. Virtual Box的host-only网络，文件共享
3. Java脚本-BeanShell
4. JAVA编程风格
5. 检测单链表中是否存在环
6. 几个开源Project管理网站
7. HttpClient源代码分析之Http Client模块
8. 设计模式一句话
9. Maven2的配置文件settings.xml
10. Maven的配置文件pom.xml

随笔分类(22)

Maven2的配置文件settings.xml

简介：

概览

当Maven运行过程中的各种配置，例如pom.xml，不想绑定到一个固定的project或者要分配给用户时，我们使用settings.xml中的settings元素来确定这些配置。这包含了本地仓库位置，远程仓库服务器以及认证信息等。

settings.xml存在于两个地方：

1.安装的地方：$M2_HOME/conf/settings.xml

2.用户的目录：${user.home}/.m2/settings.xml

前者又被叫做全局配置，后者被称为用户配置。如果两者都存在，它们的内容将被合并，并且用户范围的settings.xml优先。

如果你偶尔需要创建用户范围的settings，你可以简单的copy Maven安装路径下的settings到目录${user.home}/.m2。Maven默认的settings.xml是一个包含了注释和例子的模板，你可以快速的修改它来达到你的要求。

下面是settings下的顶层元素的一个概览：

```
<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
             http://maven.apache.org/xsd/settings-1.0.0.xsd">
    <localRepository/>
    <interactiveMode/>
    <usePluginRegistry/>
    <offline/>
    <pluginGroups/>
    <servers/>
    <mirrors/>
    <proxies/>
    <profiles/>
    <activeProfiles/>
</settings>
```

settings的内容可以在下面这些地方篡改：

1.${user.home}和所有其他的系统属性

2.${env.HOME}等环境变量

Android开发(1)
Java开发(15)
Maven(5)
Scala编程
初级算法题集(1)
目录整理

随笔档案(23)

2012年9月 (1)
2012年3月 (1)
2011年12月 (5)
2011年11月 (7)
2011年10月 (9)

最新评论

1. Re:Maven2的配置文件settings.xml
内容很详细，记不清的时候就可以来查一下
--我是个热爱学习的人

2. Re:检测单链表中是否存在环
@规格严格-功夫到家这个是链表关于环的定义。有环的定义是，链表的尾节点指向了链接中间的某个节点。如果链表环在任何一个地方，那问题就相对来说更复杂了一些。我们可以把这个问题描述成：链表打结...
--ChanShuYi

3. Re:Maven2的配置文件settings.xml
抽象不容易理解。
--VimColD

4. Re:Maven的配置文件pom.xml
总结得非常好，够用。
--罗霄（南京）

5. Re:HttpClient源代码分析之HttpClient模块
就贴个类图有什么用~ 而且类图的重点都不突出。
--陆云帆

6. Re:HttpClient源代码分析之HttpClient模块
@gschen确实好是好，看不到更多的呀，新手学不到很多...
--辰月之征

7. Re:检测单链表中是否存在环
附件条件：这个环可以出现在任何地方呢？

不可能，只会在最后！

为什么只在最后呢？
--规格严格-功夫到家

注意：settins.xml中profiles下定义的属性不能被篡改。

配置细节：

简单的值

一半以上的顶级settings元素师简单的值，代表了一直处于活跃的构建系统的元素的取值范围。

```
<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
            http://maven.apache.org/xsd/settings-1.0.0.xsd">
    <localRepository>${user.home}/.m2/repository</localRepository>
    <interactiveMode>true</interactiveMode>
    <usePluginRegistry>false</usePluginRegistry>
    <offline>false</offline>
    ...
</settings>
```

localRepository：这个值是构建系统的本地仓库的路径。默认的值是${user.home}/.m2/repository.如果一个系统想让所有登陆的用户都用同一个本地仓库的话，这个值是极其有用的。

interactiveMode：如果Maven要试图与用户交互来得到输入就设置为true，否则就设置为false，默认为true。

usePluginRegistry：如果Maven使用${user.home}/.m2/plugin-registry.xml来管理plugin的版本，就设置为true，默认为false。

offline：如果构建系统要在离线模式下工作，设置为true，默认为false。如果构建服务器因为网络故障或者安全问题不能与远程仓库相连，那么这个设置是非常有用的。

插件组

这个元素包含了一系列pluginGroup元素，每个又包含了一个groupId。当一个plugin被使用，而它的groupId没有被提供的时候，这个列表将被搜索。这个列表自动的包含了org.apache.maven.plugins和org.codehaus.mojo。

```
<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
            http://maven.apache.org/xsd/settings-1.0.0.xsd">
    ...
    <pluginGroups>
        <pluginGroup>org.mortbay.jetty</pluginGroup>
    </pluginGroups>
    ...
</settings>
```

8. Re:HttpClient源代码分析之HttpClient模块
类图过多，看不到分析的重点。
--gschen
9. Re:HttpClient入门实例之简单的pdf文件爬虫
@Lording你可以到...
--Yakov
10. Re:HttpClient入门实例之简单的pdf文件爬虫
请问，那个org.apache.http包在哪里？
--Lording
11. Re:JAVA编程风格
支持一下！！好帖
--www.xinyidt.com

阅读排行榜

1. Maven2的配置文件settings.xml(144810)
2. Maven的配置文件pom.xml(77577)
3. HttpClient源代码分析之HttpClient模块(11883)
4. Java脚本-BeanShell(9872)
5. HttpClient入门实例之简单的pdf文件爬虫(5683)
6. Java程序低手之for-in语句(4924)
7. JAVA编程风格(2536)
8. 几个开源Project管理网站(2451)
9. Maven2实践2-plugin开发(1852)
10. Maven2实践1-环境安装与准备(1548)
11. Maven2的生命周期(1408)
12. Virtual Box的host-only网络，文件共享(1270)
13. Java程序低手之Autoboxing和unboxing(1130)
14. 检测单链表中是否存在环(1019)
15. NoSQL的一些碎碎念(976)
16. Java程序低手之静态的Import(937)
17. Java程序低手之vararg(625)
18. 设计模式一句话(606)
19. 深入理解Android学习笔记(236)
20. Java程序低手之Override返回类型以及Unicode和StringBuilder(222)

评论排行榜

例如，有了上面的配置，Maven命令行可以使用简单的命令执行org.morbay.jetty:jetty-maven-plugin:run，如下

```
mvn jetty run
```

服务器

用来下载和部署的仓库是用POM中的repositories和distributionManagement元素来定义的。但是某些配置例如username和password就不应该随着pom.xml来分配了。这种类型的信息应该保存在构建服务器中的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
            http://maven.apache.org/xsd/settings-1.0.0.xsd">
    ...
    <servers>
    <server>
    <id>server001</id>
    <username>my_login</username>
    <password>my_password</password>
    <privateKey>${user.home}/.ssh/id_dsa</privateKey>
    <passphrase>some_passphrase</passphrase>
    <filePermissions>664</filePermissions>
    <directoryPermissions>775</directoryPermissions>
    <configuration></configuration>
    </server>
    </servers>
    ...
</settings>
```

id：这是Server的ID(不是登录进来的user)，与Maven想要连接上的repository/mirror中的id元素相匹配。

username，password：这两个元素成对出现，表示连接这个server需要验证username和password。

privateKey，passphrase：与前两个元素一样，这两个成对出现，分别指向了一个私钥(默认的是${user.home}/.ssh/id_dsa)和一个passphrase。passphrase和password元素可能在将来被客观化，但是现在必须以文本形式在settings.xml中设置。

filePermissions，directoryPermissions：当一个仓库文件或者目录在部署阶段被创建的时候，就必须用到权限许可。他们合法的值是三个数字，就像*nix中的文件权限，例如：664，775.

注意：如果你使用了一个私钥来登录server，那么password元素必须被省略，否则私钥将被忽视。

密码加密

1. HttpClient源代码分析之HttpClient模块(3)
2. Maven2的配置文件settings.xml(2)
3. 检测单链表中是否存在环(2)
4. HttpClient入门实例之简单的pdf文件爬虫(2)
5. JAVA编程风格(1)
6. Maven的配置文件pom.xml(1)

推荐排行榜

1. Maven的配置文件pom.xml(4)
2. HttpClient源代码分析之HttpClient模块(3)
3. Maven2的配置文件settings.xml(3)
4. Virtual Box的host-only网络，文件共享(2)
5. Java脚本-BeanShell(2)
6. JAVA编程风格(2)
7. 深入理解Android学习笔记(1)
8. Java程序低手之静态的Import(1)
9. Java程序低手之关于泛型(Generic)(1)
10. HttpClient入门实例之简单的pdf文件爬虫(1)
11. Maven2实践2-plugin开发(1)
12. Maven2实践1-环境安装与准备(1)
13. Maven2的生命周期(1)

一个新特征：服务器password和passphrase加密已经被升到2.1.0+

镜像

```
<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
            http://maven.apache.org/xsd/settings-1.0.0.xsd">
    ...
    <mirrors>
        <mirror>
            <id>planetmirror.com</id>
            <name>PlanetMirror Australia</name>
<url>http://downloads.planetmirror.com/pub/maven2</url>
            <mirrorOf>central</mirrorOf>
        </mirror>
    </mirrors>
    ...
</settings>
```

id，name：唯一的镜像标识和用户友好的镜像名称。id被用来区分mirror元素，并且当连接时候被用来获得相应的证书。

url：镜像基本的URL，构建系统敬将使用这个URL来连接仓库，而不是原来的仓库URL。

mirrorOf：镜像所包含的仓库的Id。例如，指向Maven central仓库的镜像(http://repo1.maven.org/maven2/)，设置这个元素为central。更多的高级映射例如repo1,repo2 或者*,!inhouse都是可以的。没必要一定和mirror的id相匹配。

代理

```
<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
            http://maven.apache.org/xsd/settings-1.0.0.xsd">
    ...
    <proxies>
        <proxy>
            <id>myproxy</id>
            <active>true</active>
            <protocol>http</protocol>
            <host>proxy.somewhere.com</host>
            <port>8080</port>
            <username>proxyuser</username>
            <password>somepassword</password>

```

```
<nonProxyHosts>*.google.com|ibiblio.org</nonProxyHosts>
        </proxy>
    </proxies>
    ...
</settings>
```

id：proxy的唯一标识，用来区别proxy元素。

active：当proxy被激活的时候为true。当申明的代理很多的时候，这个很有用，但是同一时间仅有一个被激活。

protocol，host，port：代理地址protocol://host:port的分散形式。

username，password：两个元素成对出现，提供连接proxy服务器时的认证。

nonProxyHosts：这里列出了不需要使用代理的hosts。列表的分隔符是proxy服务器想要的类型。上面例子使用了pipe分隔符，逗号分隔符也比较通用。

配置文件

settings.xml中的profile是pom.xml中的profile的简洁形式。它包含了激活(activation)，仓库(repositories)，插件仓库(pluginRepositories)和属性(properties)元素。profile元素仅包含这四个元素是因为他们涉及到整个的构建系统，而不是个别的POM配置。

如果settings中的profile被激活，那么它的值将重载POM或者profiles.xml中的任何相等ID的profiles。

激活(activation)

activations是profile的关键，就像POM中的profiles，profile的能力在于它在特定情况下可以修改一些值。而这些情况是通过activation来指定的。

```
<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
            http://maven.apache.org/xsd/settings-1.0.0.xsd">
    ...
    <profiles>
        <profile>
            <id>test</id>
            <activation>
                <activeByDefault>false</activeByDefault>
                <jdk>1.5</jdk>
                <os>
                    <name>Windows XP</name>
                    <family>Windows</family>
                    <arch>x86</arch>
                    <version>5.1.2600</version>
                </os>
                <property>
                    <name>mavenVersion</name>
                    <value>2.0.3</value>
                </property>
                <file>
```

```
                    <exists>${basedir}/file2.properties</exists>
                    <missing>${basedir}/file1.properties</missing>
                </file>
            </activation>
            ...
        </profile>
    </profiles>
    ...
</settings>
```

如果所有指定的条件都达到了，那么，activation就被触发，而且不需要一次性全部达到。

jdk：在jdk元素中，activation有一个内建的，java版本检测。如果检测到jdk版本与期待的一样，那么就激活。在上面的例子中，1.5.0_06是满足的。

os：os元素可以定义一些上面所示的操作系统特定的属性。

property：如果Maven检测到相应的名值对的属性，那么，这个profile将被激活。

file：如果给定的文件存在，或者不存在那么将激活这个profile。

activation并不是唯一激活profile的途径。settings.xml中的activeProfile包含了profile的id。他们也可以通过命令行来显式的激活，例如-P test。

如果你想查看在一个构建过程中有哪些profile会被激活。就使用maven-help-plugin

mvn help:active-profiles

属性(properites)

Maven的属性是值占位符，就像Ant中的属性。如果X是一个属性的话，那么它的值在POM中可以使用${X}来进行任意地方的访问。他们来自于五种不同的风格，所有都可以从settings.xml文件中访问到。

1.env.X：使用"env."前缀将会返回当前的环境变量。例如${env.PATH}就是使用了$path环境变量。

2.project.X：一个点"."分割的路径，在POM中就是相关的元素的值。例如：<project><version>1.0</version></project>就可以通过${project.version}来访问。

3.settings.X：一个点"."分割的路径，在settings.xml中就是相对应的元素的值，例如：<settings><offline>false</offline></settings>就可以通过${settings.offline}来访问。

4.Java系统属性：所有通过java.lang.System.getProperties()来访问的属性都可以像POM中的属性一样访问，例如：${java.home}

5.X：被<properties/>或者外部文件定义的属性，值可以这样访问${someVar}

```
<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
            http://maven.apache.org/xsd/settings-1.0.0.xsd">
    ...
    <profiles>
        <profile>
            ...
            <properties>
                <user.install>${user.home}/our-
project</user.install>
            </properties>
            ...
        </profile>
    </profiles>
    ...
</settings>
```

如果这个profile被激活，那么属性${user.install}就可以被访问了。

仓库(repositories)

仓库是Maven用来构筑构建系统的本地仓库的远程项目集合。它来自于被Maven叫做插件和依赖的本地仓库。不同的远程仓库包含不同的项目，当profile被激活，他们就会需找匹配的release或者snapshot构件。

```
<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
            http://maven.apache.org/xsd/settings-1.0.0.xsd">
    ...
    <profiles>
        <profile>
        ...
        <repositories>
            <repository>
                <id>codehausSnapshots</id>
                <name>Codehaus Snapshots</name>
                <releases>
                    <enabled>false</enabled>
                    <updatePolicy>always</updatePolicy>
                    <checksumPolicy>warn</checksumPolicy>
                </releases>
                <snapshots>
                    <enabled>true</enabled>
                    <updatePolicy>never</updatePolicy>
                    <checksumPolicy>fail</checksumPolicy>
                </snapshots>
<url>http://snapshots.maven.codehaus.org/maven2</url>
                <layout>default</layout>
            </repository>
```

```
        </repositories>
        <pluginRepositories>
        ...
        </pluginRepositories>
        ...
        </profile>
    </profiles>
    ...
</settings>
```

releases，snapshots：这是各种构件的策略，release或者snapshot。因了这两个集合，POM可以在单个的仓库中不依赖于另外一个的策略而改变当前策略。例如：一个人可能只下载snapshot用来开发。

enable：true或者false，来标记仓库是否为各自的类型激活(release 或者snapshot)。

updatePolicy：这个元素指明了更新的频率。Maven会比较本地POM与远程的时间戳。可选的项目为：always，daily，interval:X，nerver。

checksumPolicy：当Maven向仓库部署文件的时候，它也部署了相应的校验和文件。可选的为：ignore，fail，warn，或者不正确的校验和。

layout：在上面描述仓库的时候，我们提到他们有统一的布局。这完全正确。使用这个来表明它是default还是legacy。

插件仓库(plugin repositories)

仓库包含了两种重要类型的构件。第一种是用来做其他构件依赖的构件，这是在中央仓库中的大多数插件。另外一种类型的构件就是插件。Maven的插件本身就是一种特殊的构件。因此，插件仓库被从其他仓库中分离出来。无论怎么说，pluginRepositories元素模块的结构与repositories模块很相似。pluginRepository元素指向一个可以找到新插件的远程地址。

激活配置(Active Profiles)

```
<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
            http://maven.apache.org/xsd/settings-1.0.0.xsd">
    ...
    <activeProfiles>
        <activeProfile>env-test</activeProfile>
    </activeProfiles>
</settings>
```

settings.xml最后一个谜题是activeProfiles元素。它包含一系列的activeProfile元素，每个都有一个profile id的值，任何profile id被定义到activeProfile的profile将被激活，不管其他的环境设置怎么样。如果没有匹配的profile被找到，那么就什么事情也不做。例如：如果env-test是一个activeProfile，一个在pom.xml或者profile.xml中的具有相应id的

profile将被激活。如果没有这样的profile被找到，就什么事也不做，一切照常。

原文地址：http://maven.apache.org/settings.html

分类: Maven

好文要顶 关注我 收藏该文

Yakov
关注 - 2
粉丝 - 24
+加关注

3 0

« 上一篇：Maven的配置文件pom.xml
» 下一篇：设计模式一句话

posted on 2011-11-26 19:40 Yakov 阅读(144813) 评论(2) 编辑 收藏

FeedBack:

#1楼 2016-03-04 18:15 VimColD

抽象不容易理解。

支持(0) 反对(0)

#2楼 2016-07-08 11:58 我是个热爱学习的人

内容很详细，记不清的时候就可以来查一下

支持(1) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

【推荐】50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
【推荐】Google+GitHub联手打造前端工程师课程

最新**IT**新闻:

- 直播新玩法 陌陌携手一亩田玩转直播+农业
- 京东开通港澳地区直送服务：运费最高429元
- 299元！小米众筹90分Ultra Smart智能跑鞋发布：Intel芯片
- 别做口头上的学习者
- Alphabet执行董事长施密特：大数据是各国争抢的强大武器

» 更多新闻...

最新知识库文章:

- 垃圾回收原来是这么回事
- 「代码家」的学习过程和学习经验分享
- 写给未来的程序媛
- 高质量的工程代码为什么难写
- 循序渐进地代码重构

» 更多知识库文章...