

Maven: settings.xml、pom.xml完整配置

来自: [彭世瑜的博客](#) 2021-11-26 2364

[举报](#)

简介: Maven: settings.xml、pom.xml完整配置

完整配置

settings.xml

```
1 <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
2     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
4         http://maven.apache.org/xsd/settings-1.0.0.xsd">
5     <!-- 本地仓库配置: 默认~/.m2/repository[店家推荐修改配置] -->
6     <localRepository>${user.home}/.m2/repository</localRepository>
7
8     <!-- 交互方式配置, 读取用户输入信息[使用默认即可, 很少修改] -->
9     <interactiveMode>true</interactiveMode>
10
11     <!-- 是否启用独立的插件配置文件, 一般很少启用[默认即可, 很少修改] -->
12     <usePluginRegistry>false</usePluginRegistry>
13
14     <!-- 是否启用离线构建模式, 一般很少修改[如果长时间不能联网的情况下可以修改] -->
15     <offline>false</offline>
16
17     <!-- 是否启用插件groupId自动扫描[很少使用, 配置插件时建议全信息配置] -->
18     <pluginGroups>
19         <pluginGroup>org.apache.maven.plugins</pluginGroup>
20     </pluginGroups>
21
22     <!--配置服务端的一些设置如身份认证信息(eg: 账号、密码) -->
23     <servers>
24         <!--服务器元素包含配置服务器时需要的信息 -->
25         <server>
26             <!--这是server的id(注意不是用户登陆的id)
27             该id与distributionManagement中repository元素的id相匹配。
28             -->
29             <id>server_001</id>
30             <!--身份鉴权令牌。鉴权/认证用户名和鉴权密码表示服务器认证所需要的登录名和密码。 -->
31             <username>my_login</username>
32             <!--身份鉴权密码。鉴权/认证用户名和鉴权密码表示服务器认证所需要的登录名和密码-->
33             <password>my_password</password>
34             <!--鉴权/认证时使用的私钥文件位置。和前两个元素类似
35             私钥位置和私钥密码指定了一个私钥的路径(默认是${user.home}/.ssh/id_dsa) ==>
```

```

36     <privateKey>${usr.home}/.ssh/id_dsa</privateKey>
37     <!--鉴权/认证时使用的私钥密码。 -->
38     <passphrase>some_passphrase</passphrase>
39     <!--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。 -->
40     <filePermissions>664</filePermissions>
41     <!--目录被创建时的权限。 -->
42     <directoryPermissions>775</directoryPermissions>
43 </server>
44 </servers>
45
46 <mirrors>
47     <!-- 默认仓库配置给定的下载镜像位置 -->
48     <mirror>
49         <!-- 该镜像的唯一标识符。id用来区分不同的mirror元素。 -->
50         <id>nexus aliyun</id>
51         <!-- 镜像名称 -->
52         <name>Nexus Aliyun</name>
53         <!-- 该镜像的URL。构建系统会优先考虑使用该URL，而非使用默认的服务器URL。 -->
54         <url>http://downloads.planetmirror.com/pub/maven2</url>
55         <!-- 被镜像的服务器的id。
56         如果我们要设置了一个Maven中央仓库（http://repo.maven.apache.org/maven2/）的镜像
57         就需要将mirrorOf设置成central。
58         保持和中央仓库的id central一致。 这样就能替代中央仓库的功能了-->
59         <mirrorOf>central</mirrorOf>
60     </mirror>
61 </mirrors>
62
63 <proxies>
64     <!--代理元素包含配置代理时需要的信息 -->
65     <proxy>
66         <!--代理的唯一定义符，用来区分不同的代理元素。 -->
67         <id>myproxy</id>
68         <!--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。 -->
69         <active>true</active>
70         <!--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。 -->
71         <protocol>http</protocol>
72         <!--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。 -->
73         <host>proxy.somewhere.com</host>
74         <!--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。 -->
75         <port>8080</port>
76         <!--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。 -->
77         <username>proxyuser</username>
78         <!--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。 -->
79         <password>somepassword</password>
80         <!--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。 -->
81         <nonProxyHosts>*.google.com|ibiblio.org</nonProxyHosts>
82     </proxy>

```

```

83     </proxies>
84
85     <profiles>
86         <profile>
87             <!-- profile的唯一标识 -->
88             <id>test</id>
89             <!-- 自动触发profile的条件逻辑 -->
90             <activation />
91             <!-- 扩展属性列表 -->
92             <properties />
93             <!-- 远程仓库列表 -->
94             <repositories />
95             <!-- 插件仓库列表 -->
96             <pluginRepositories />
97         </profile>
98     </profiles>
99
100
101     <activeProfiles>
102         <!-- 要激活的profile id -->
103         <activeProfile>env-test</activeProfile>
104     </activeProfiles>
105
106     <activation>
107         <!--profile默认是否激活的标识 -->
108         <activeByDefault>false</activeByDefault>
109         <!--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。 -->
110         <jdk>1.5</jdk>
111         <!--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。
            -->
112         <os>
113             <!--激活profile的操作系统的名字 -->
114             <name>Windows XP</name>
115             <!--激活profile的操作系统所属家族(如 'windows') -->
116             <family>Windows</family>
117             <!--激活profile的操作系统体系结构 -->
118             <arch>x86</arch>
119             <!--激活profile的操作系统版本 -->
120             <version>5.1.2600</version>
121         </os>
122         <!--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 -->
123         <property>
124             <!--激活profile的属性的名称 -->
125             <name>mavenVersion</name>
126             <!--激活profile的属性的值 -->
127             <value>2.0.3</value>

```

```

128     </property>
129     <!--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。 -->
130     <file>
131         <!--如果指定的文件存在，则激活profile。 -->
132         <exists>${basedir}/file2.properties</exists>
133         <!--如果指定的文件不存在，则激活profile。 -->
134         <missing>${basedir}/file1.properties</missing>
135     </file>
136 </activation>
137
138 <properties>
139     <spring.Version>5.2.8</spring.Version>
140 </properties>
141
142 <repositories>
143     <!--包含需要连接到远程仓库的信息 -->
144     <repository>
145         <!--远程仓库唯一标识 -->
146         <id>codehausSnapshots</id>
147         <!--远程仓库名称 -->
148         <name>Codehaus Snapshots</name>
149         <!--如何处理远程仓库里发布版本的下载 -->
150         <releases>
151             <!--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。 -->
152             <enabled>false</enabled>
153             <!--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）。 -->
154             <updatePolicy>always</updatePolicy>
155             <!--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）。 -->
156             <checksumPolicy>warn</checksumPolicy>
157         </releases>
158         <!--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 -->
159         <snapshots>
160             <enabled />
161             <updatePolicy />
162             <checksumPolicy />
163         </snapshots>
164         <!--远程仓库URL，按protocol://hostname/path形式 -->
165         <url>http://snapshots.maven.codehaus.org/maven2</url>
166         <!--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）。 -->
167         <layout>default</layout>
168     </repository>
169 </repositories>
170

```

pom.xml

```

1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4     http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <!--
6       1、项目基本信息配置
7
8     -->
9     <!--父项目的坐标。如果项目中没有规定某个元素的值，那么父项目中的对应值即为项目的默认值。
        坐标包括group ID, artifact ID和 version。 -->
10    <parent>
11        <!--被继承的父项目的构件标识符 -->
12        <artifactId />
13        <!--被继承的父项目的全球唯一标识符 -->
14        <groupId />
15        <!--被继承的父项目的版本 -->
16        <version />
17        <!--父项目的pom.xml文件的相对路径。相对路径允许你选择一个不同的路径。默认值
        是../pom.xml。Maven首先在构建当前项目的地方寻找父项目的pom，其次在文件系统的这个位置
        （relativePath位置），然后在本地仓库，最后在远程仓库寻找父项目的pom。 -->
18        <relativePath />
19    </parent>
20    <!--声明项目描述符遵循哪一个POM模型版本。模型本身的版本很少改变，虽然如此，但它仍然是必不可
        少的，这是为了当Maven引入了新的特性或者其他模型变更的时候，确保稳定性。 -->
21    <modelVersion>4.0.0</modelVersion>
22    <!--项目的全球唯一标识符，通常使用全限定的包名区分该项目和其他项目。并且构建时生成的路径也
        是由此生成，如com.mycompany.app生成的相对路径为：/com/mycompany/app -->
23    <groupId>asia.banseon</groupId>
24    <!--构件的标识符，它和group ID一起唯一标识一个构件。换句话说，你不能有两个不同的项目拥有
        同样的artifact ID和groupId；在某个特定的group
25    ID下，artifact ID也必须是唯一的。构件是项目产生的或使用的一个东西，Maven为项目产生
        的构件包括：JARs，源码，二进制发布和WARs等。 -->
26    <artifactId>banseon-maven2</artifactId>
27    <!--项目产生的构件类型，例如jar、war、ear、pom。插件可以创建他们自己的构件类型，所以前面
        列的不是全部构件类型 -->
28    <packaging>jar</packaging>
29    <!--项目当前版本，格式为：主版本.次版本.增量版本-限定版本号 -->
30    <version>1.0-SNAPSHOT</version>
31    <!--项目的名称，Maven产生的文档用 -->
32    <name>banseon-maven</name>
33    <!--项目主页的URL，Maven产生的文档用 -->
34    <url>http://www.baidu.com/banseon</url>
35    <!--项目的详细描述，Maven 产生的文档用。 当这个元素能够用HTML格式描述时（例如，CDATA中
        的文本会被解析器忽略，就可以包含HTML标签），

```

```

36      不鼓励使用纯文本描述。如果你需要修改产生的web站点的索引页面，你应该修改你自己的索引页
      文件，而不是调整这里的文档。 -->
37      <description>A maven project to study maven.</description>
38      <!--项目创建年份，4位数字。当产生版权信息时需要使用这个值。 -->
39      <inceptionYear />
40      <!--项目相关邮件列表信息 -->
41      <mailingLists>
42          <!--该元素描述了项目相关的所有邮件列表。自动产生的网站引用这些信息。 -->
43          <mailingList>
44              <!--邮件的名称 -->
45              <name>Demo</name>
46              <!--发送邮件的地址或链接，如果是邮件地址，创建文档时，mailto: 链接会被自动创建
-->
47              <post>Demo@126.com</post>
48              <!--订阅邮件的地址或链接，如果是邮件地址，创建文档时，mailto: 链接会被自动创建
-->
49              <subscribe>Demo@126.com</subscribe>
50              <!--取消订阅邮件的地址或链接，如果是邮件地址，创建文档时，mailto: 链接会被自动
创建 -->
51              <unsubscribe>Demo@126.com</unsubscribe>
52              <!--你可以浏览邮件信息的URL -->
53              <archive>http://localhost:8080/demo/dev/</archive>
54          </mailingList>
55      </mailingLists>
56      <!--项目开发者列表 -->
57      <developers>
58          <!--某个项目开发者的信息 -->
59          <developer>
60              <!--SCM里项目开发者的唯一标识符 -->
61              <id>HELLO WORLD</id>
62              <!--项目开发者的全名 -->
63              <name>youname</name>
64              <!--项目开发者的email -->
65              <email>youname@qq.com</email>
66              <!--项目开发者的主页的URL -->
67              <url />
68              <!--项目开发者在项目中扮演的角色，角色元素描述了各种角色 -->
69              <roles>
70                  <role>Project Manager</role>
71                  <role>Architect</role>
72              </roles>
73              <!--项目开发者所属组织 -->
74              <organization>demo</organization>
75              <!--项目开发者所属组织的URL -->
76              <organizationUrl>http://www.xxx.com/</organizationUrl>
77              <!--项目开发者属性，如即时消息如何处理等 -->
78              <properties>
79                  <dept>No</dept>
80              </properties>
81              <!--项目开发者所在时区， -11到12范围内的整数。 -->
82              <timezone>+8</timezone>

```

```

83         </developer>
84     </developers>
85     <!--项目的其他贡献者列表 -->
86     <contributors>
87         <!--项目的其他贡献者。参见developers/developer元素 -->
88         <contributor>
89             <name />
90             <email />
91             <url />
92             <organization />
93             <organizationUrl />
94             <roles />
95             <timezone />
96             <properties />
97         </contributor>
98     </contributors>
99     <!--该元素描述了项目所有License列表。 应该只列出该项目的license列表，不要列出依赖项目的
100     license列表。如果列出多个license，用户可以选择它们中的一个而不是接受所有license。 -->
101     <licenses>
102         <!--描述了项目的license，用于生成项目的web站点的license页面，其他一些报表和
103         validation也会用到该元素。 -->
104         <license>
105             <!--license用于法律上的名称 -->
106             <name>Apache 2</name>
107             <!--官方的license正文页面的URL -->
108             <url>http://www.xxxx.com/LICENSE-2.0.txt</url>
109             <!--项目分发的主要方式： repo，可以从Maven库下载 manual， 用户必须手动下载和安
110             装依赖 -->
111             <distribution>repo</distribution>
112             <!--关于license的补充信息 -->
113             <comments>A business-friendly OSS license</comments>
114         </license>
115     </licenses>
116     <!--SCM(Source Control Management)标签允许你配置你的代码库，供Maven web站点和其它插
117     件使用。 -->
118     <scm>
119         <!--SCM的URL,该URL描述了版本库和如何连接到版本库。欲知详情，请看SCMs提供的URL格式
120         和列表。该连接只读。 -->
121         <connection>
122             scm:svn:http://svn.xxxx.com/maven/xxxxx-maven2-trunk(dao-trunk)
123         </connection>
124         <!--给开发者使用的，类似connection元素。即该连接不仅仅只读 -->
125         <developerConnection>
126             scm:svn:http://svn.xxxx.com/maven/dao-trunk
127         </developerConnection>
128         <!--当前代码的标签，在开发阶段默认为HEAD -->
129         <tag />
130         <!--指向项目的可浏览SCM库（例如ViewVC或者Fisheye）的URL。 -->
131         <url>http://svn.xxxxx.com/</url>
132     </scm>
133     <!--描述项目所属组织的各种属性。Maven产生的文档用 -->

```

```

129     <organization>
130         <!--组织的全名 -->
131         <name>demo</name>
132         <!--组织主页的URL -->
133         <url>http://www.xxxxxx.com/</url>
134     </organization>
135
136
137
138     <!--
139     2、项目构建环境配置
140
141     -->
142     <!--描述了这个项目构建环境中的前提条件。 -->
143     <prerequisites>
144         <!--构建该项目或使用该插件所需要的Maven的最低版本 -->
145         <maven />
146     </prerequisites>
147     <!--项目的问题管理系统(Bugzilla, Jira, Scarab,或任何你喜欢的问题管理系统)的名称和
    URL, 本例为 jira -->
148     <issueManagement>
149         <!--问题管理系统（例如jira）的名字, -->
150         <system>jira</system>
151         <!--该项目使用的问题管理系统的URL -->
152         <url>http://jira.xxxx.com/xxxx</url>
153     </issueManagement>
154     <!--项目持续集成信息 -->
155     <ciManagement>
156         <!--持续集成系统的名字, 例如continuum -->
157         <system />
158         <!--该项目使用的持续集成系统的URL（如果持续集成系统有web接口的话）。 -->
159         <url />
160         <!--构建完成时, 需要通知的开发者/用户的配置项。包括被通知者信息和通知条件（错误, 失
    败, 成功, 警告） -->
161         <notifiers>
162             <!--配置一种方式, 当构建中断时, 以该方式通知用户/开发者 -->
163             <notifier>
164                 <!--传送通知的途径 -->
165                 <type />
166                 <!--发生错误时是否通知 -->
167                 <sendOnError />
168                 <!--构建失败时是否通知 -->
169                 <sendOnFailure />
170                 <!--构建成功时是否通知 -->
171                 <sendOnSuccess />
172                 <!--发生警告时是否通知 -->
173                 <sendOnWarning />
174                 <!--不赞成使用。通知发送到哪里 -->
175                 <address />
176                 <!--扩展配置项 -->

```



```

177         <configuration />
178     </notifier>
179 </notifiers>
180 </ciManagement>
181 <!--模块（有时称作子项目） 被构建成项目的一部分。列出的每个模块元素是指向该模块的目录的相
    对路径 -->
182 <modules />
183 <!--构建项目需要的信息 -->
184 <build>
185     <!--该元素设置了项目源码目录，当构建项目的时候，构建系统会编译目录里的源码。该路径是
    相对于pom.xml的相对路径。 -->
186     <sourceDirectory />
187     <!--该元素设置了项目脚本源码目录，该目录和源码目录不同：绝大多数情况下，该目录下的内
    容 会被拷贝到输出目录(因为脚本是被解释的，而不是被编译的)。 -->
188     <scriptSourceDirectory />
189     <!--该元素设置了项目单元测试使用的源码目录，当测试项目的时候，构建系统会编译目录里的
    源码。该路径是相对于pom.xml的相对路径。 -->
190     <testSourceDirectory />
191     <!--被编译过的应用程序class文件存放的目录。 -->
192     <outputDirectory />
193     <!--被编译过的测试class文件存放的目录。 -->
194     <testOutputDirectory />
195     <!--使用来自该项目的一系列构建扩展 -->
196     <extensions>
197         <!--描述使用到的构建扩展。 -->
198         <extension>
199             <!--构建扩展的groupId -->
200             <groupId />
201             <!--构建扩展的artifactId -->
202             <artifactId />
203             <!--构建扩展的版本 -->
204             <version />
205         </extension>
206     </extensions>
207     <!--当项目没有规定目标（Maven2 叫做阶段）时的默认值 -->
208     <defaultGoal />
209     <!--这个元素描述了项目相关的所有资源路径列表，例如和项目相关的属性文件，这些资源被包
    含在最终的打包文件里。 -->
210     <resources>
211         <!--这个元素描述了项目相关或测试相关的所有资源路径 -->
212         <resource>
213             <!--描述了资源的目标路径。该路径相对target/classes目录（例
    如${project.build.outputDirectory}）。举个例子，如果你想资源在特定的包里
    (org.apache.maven.messages)，你就必须该元素设置为org/apache/maven/messages。然而，如果
    你只是想把资源放到源码目录结构里，就不需要该配置。 -->
214             <targetPath />
215             <!--是否使用参数值代替参数名。参数值取自properties元素或者文件里配置的属
    性，文件在filters元素里列出。 -->
216             <filtering />
217             <!--描述存放资源的目录，该路径相对POM路径 -->

```

```

218         <directory />
219         <!--包含的模式列表，例如**/*.xml. -->
220         <includes />
221         <!--排除的模式列表，例如**/*.xml -->
222         <excludes />
223     </resource>
224 </resources>
225 <!--这个元素描述了单元测试相关的所有资源路径，例如和单元测试相关的属性文件。 -->
226 <testResources>
227     <!--这个元素描述了测试相关的所有资源路径，参见build/resources/resource元素的
说明 -->
228     <testResource>
229         <targetPath />
230         <filtering />
231         <directory />
232         <includes />
233         <excludes />
234     </testResource>
235 </testResources>
236 <!--构建产生的所有文件存放的目录 -->
237 <directory />
238 <!--产生的构件的文件名，默认值是${artifactId}-${version}。 -->
239 <finalName />
240 <!--当filtering开关打开时，使用到的过滤器属性文件列表 -->
241 <filters />
242 <!--子项目可以引用的默认插件信息。该插件配置项直到被引用时才会被解析或绑定到生命周
期。给定插件的任何本地配置都会覆盖这里的配置 -->
243 <pluginManagement>
244     <!--使用的插件列表 。 -->
245     <plugins>
246         <!--plugin元素包含描述插件所需要的信息。 -->
247         <plugin>
248             <!--插件在仓库里的group ID -->
249             <groupId />
250             <!--插件在仓库里的artifact ID -->

```

```

251         <artifactId />
252         <!--被使用的插件的版本（或版本范围） -->
253         <version />
254         <!--是否从该插件下载Maven扩展（例如打包和类型处理器），由于性能原因，
           只有在真需要下载时，该元素才被设置成enabled。 -->
255         <extensions />
256         <!--在构建生命周期中执行一组目标的配置。每个目标可能有不同的配置。 -->
257         <executions>
258             <!--execution元素包含了插件执行需要的信息 -->
259             <execution>
260                 <!--执行目标的标识符，用于标识构建过程中的目标，或者匹配继承过
           程中需要合并的执行目标 -->
261                 <id />
262                 <!--绑定了目标的构建生命周期阶段，如果省略，目标会被绑定到源数
           据里配置的默认阶段 -->
263                 <phase />
264                 <!--配置的执行目标 -->
265                 <goals />
266                 <!--配置是否被传播到子POM -->
267                 <inherited />
268                 <!--作为DOM对象的配置 -->
269                 <configuration />
270             </execution>
271         </executions>
272         <!--项目引入插件所需要的额外依赖 -->
273         <dependencies>
274             <!--参见dependencies/dependency元素 -->
275             <dependency>.....</dependency>
276         </dependencies>
277         <!--任何配置是否被传播到子项目 -->
278         <inherited />
279         <!--作为DOM对象的配置 -->
280         <configuration />
281     </plugin>
282 </plugins>
283 </pluginManagement>
284 <!--使用的插件列表 -->
285 <plugins>
286     <!--参见build/pluginManagement/plugins/plugin元素 -->
287     <plugin>
288         <groupId />
289         <artifactId />
290         <version />
291         <extensions />
292         <executions>
293             <execution>
294                 <id />
295                 <phase />
296                 <goals />
297                 <inherited />

```

```

298         <configuration />
299     </execution>
300 </executions>
301 <dependencies>
302     <!--参见dependencies/dependency元素 -->
303     <dependency>.....</dependency>
304 </dependencies>
305 <goals />
306 <inherited />
307 <configuration />
308 </plugin>
309 </plugins>
310 </build>
311 <!--在列的项目构建profile，如果被激活，会修改构建处理 -->
312 <profiles>
313     <!--根据环境参数或命令行参数激活某个构建处理 -->
314     <profile>
315         <!--构建配置的唯一标识符。即用于命令行激活，也用于在继承时合并具有相同标识符的
profile。 -->
316         <id />
317         <!--自动触发profile的条件逻辑。Activation是profile的开启钥匙。profile的力量
来自于它 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。 -->
318         <activation>
319             <!--profile默认是否激活的标志 -->
320             <activeByDefault />
321             <!--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。 -->
322             <jdk />
323             <!--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。 -->
324             <os>
325                 <!--激活profile的操作系统的名字 -->
326                 <name>Windows XP</name>
327                 <!--激活profile的操作系统所属家族(如 'windows') -->
328                 <family>Windows</family>
329                 <!--激活profile的操作系统体系结构 -->
330                 <arch>x64</arch>
331                 <!--激活profile的操作系统版本 -->
332                 <version>6.1.7100</version>
333             </os>
334             <!--如果Maven检测到某一个属性（其值可以在POM中通过${名称}引用），其拥有对
应的名称和值，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 -->
335             <property>
336                 <!--激活profile的属性的名称 -->
337                 <name>mavenVersion</name>
338                 <!--激活profile的属性的值 -->
339                 <value>2.0.3</value>

```

```

340         </property>
341         <!--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。 -->
342         <file>
343             <!--如果指定的文件存在，则激活profile。 -->
344             <exists>/usr/local/xxxx/xxxx-home/tomcat/maven-guide-zh-to-
production/workspace/
345             </exists>
346             <!--如果指定的文件不存在，则激活profile。 -->
347             <missing>/usr/local/xxxx/xxxx-home/tomcat/maven-guide-zh-to-
production/workspace/
348             </missing>
349         </file>
350     </activation>
351     <!--构建项目所需要的信息。参见build元素 -->
352     <build>
353         <defaultGoal />
354         <resources>
355             <resource>
356                 <targetPath />
357                 <filtering />
358                 <directory />
359                 <includes />
360                 <excludes />
361             </resource>
362         </resources>
363         <testResources>
364             <testResource>
365                 <targetPath />
366                 <filtering />
367                 <directory />
368                 <includes />
369                 <excludes />
370             </testResource>
371         </testResources>
372         <directory />
373         <finalName />
374         <filters />
375         <pluginManagement>
376             <plugins>
377                 <!--参见build/pluginManagement/plugins/plugin元素 -->
378                 <plugin>
379                     <groupId />
380                     <artifactId />
381                     <version />
382                     <extensions />
383                     <executions>
384                         <execution>

```

```

385         <id />
386         <phase />
387         <goals />
388         <inherited />
389         <configuration />
390     </execution>
391 </executions>
392 <dependencies>
393     <!--参见dependencies/dependency元素 -->
394     <dependency>.....</dependency>
395 </dependencies>
396 <goals />
397 <inherited />
398 <configuration />
399 </plugin>
400 </plugins>
401 </pluginManagement>
402 <plugins>
403     <!--参见build/pluginManagement/plugins/plugin元素 -->
404     <plugin>
405         <groupId />
406         <artifactId />
407         <version />
408         <extensions />
409         <executions>
410             <execution>
411                 <id />
412                 <phase />
413                 <goals />
414                 <inherited />
415                 <configuration />
416             </execution>
417         </executions>
418         <dependencies>
419             <!--参见dependencies/dependency元素 -->
420             <dependency>.....</dependency>
421         </dependencies>
422         <goals />
423         <inherited />
424         <configuration />
425     </plugin>
426 </plugins>
427 </build>
428 <!--模块（有时称作子项目） 被构建成项目的一部分。列出的每个模块元素是指向该模块
    的目录的相对路径 -->
429 <modules />
430 <!--发现依赖和扩展的远程仓库列表。 -->
431 <repositories>
432     <!--参见repositories/repository元素 -->
433     <repository>
434         <releases>

```

```

435         <enabled />
436         <updatePolicy />
437         <checksumPolicy />
438     </releases>
439     <snapshots>
440         <enabled />
441         <updatePolicy />
442         <checksumPolicy />
443     </snapshots>
444     <id />
445     <name />
446     <url />
447     <layout />
448 </repository>
449 </repositories>
450 <!--发现插件的远程仓库列表，这些插件用于构建和报表 -->
451 <pluginRepositories>
452     <!--包含需要连接到远程插件仓库的信息.参见repositories/repository元素
-->
453     <pluginRepository>
454         <releases>
455             <enabled />
456             <updatePolicy />
457             <checksumPolicy />
458         </releases>
459         <snapshots>
460             <enabled />
461             <updatePolicy />
462             <checksumPolicy />
463         </snapshots>
464         <id />
465         <name />
466         <url />
467         <layout />
468     </pluginRepository>
469 </pluginRepositories>
470 <!--该元素描述了项目相关的所有依赖。 这些依赖组成了项目构建过程中的一个个环节。
它们自动从项目定义的仓库中下载。要获取更多信息，请看项目依赖机制。 -->
471 <dependencies>
472     <!--参见dependencies/dependency元素 -->
473     <dependency>.....</dependency>
474 </dependencies>
475 <!--不赞成使用。现在Maven忽略该元素。 -->
476 <reports />
477 <!--该元素包括使用报表插件产生报表的规范。当用户执行“mvn site”，这些报表就会运
行。 在页面导航栏能看到所有报表的链接。参见reporting元素 -->
478 <reporting>.....</reporting>
479 <!--参见dependencyManagement元素 -->
480 <dependencyManagement>
481     <dependencies>
482         <!--参见dependencies/dependency元素 -->

```

```

483         <dependency>.....</dependency>
484     </dependencies>
485 </dependencyManagement>
486 <!--参见distributionManagement元素 -->
487 <distributionManagement>.....</distributionManagement>
488 <!--参见properties元素 -->
489 <properties />
490 </profile>
491 </profiles>
492
493
494 <!--
495 3、项目仓库管理配置
496
497 -->
498 <!--发现依赖和扩展的远程仓库列表。 -->
499 <repositories>
500     <!--包含需要连接到远程仓库的信息 -->
501     <repository>
502         <!--如何处理远程仓库里发布版本的下载 -->
503         <releases>
504             <!--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。
-->
505             <enabled />
506             <!--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）。 -->
507             <updatePolicy />
508             <!--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）。 -->
509             <checksumPolicy />
510         </releases>
511         <!--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 -->
512         <snapshots>
513             <enabled />
514             <updatePolicy />
515             <checksumPolicy />
516         </snapshots>
517         <!--远程仓库唯一标识符。可以用来匹配在settings.xml文件里配置的远程仓库 -->
518         <id>banseon-repository-proxy</id>
519         <!--远程仓库名称 -->
520         <name>banseon-repository-proxy</name>
521         <!--远程仓库URL，按protocol://hostname/path形式 -->
522         <url>http://10.10.10.123:8080/repository/</url>
523         <!--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
524         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）。 -->
525         <layout>default</layout>

```



```
526         </repository>
527     </repositories>
528
529     <!-- 发现插件的远程仓库列表，这些插件用于构建和报表 -->
530     <pluginRepositories>
531         <!-- 包含需要连接到远程插件仓库的信息. 参见repositories/repository元素 -->
532         <pluginRepository>.....</pluginRepository>
533     </pluginRepositories>
534
535     <!--
536     4、项目依赖管理配置
537
538     -->
539     <!-- 继承自该项目的所有子项目的默认依赖信息。这部分的依赖信息不会被立即解析, 而是当子项目声
540     明一个依赖（必须描述group ID和artifact
541     ID信息），如果group ID和artifact ID以外的一些信息没有描述，则通过group ID和
542     artifact ID匹配到这里的依赖，并使用这里的依赖信息。 -->
541     <dependencyManagement>
542     <dependencies>
```