

KentySky

站在巨人的肩上,希望自己也可以长高一点!!!

blog上的很多都是摘抄别人的知识
点.如果是您的原创,而且不能盗版,那
么请给我留言!谢谢!



[1.24] While there is life, there is
hope. 有生...

[我的主页](#) [个人资料](#)
[我的闪存](#) [发短消息](#)

<	2008年1月					>
日	一	二	三	四	五	六
30	31	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	9

搜索

常用链接

[我的随笔](#)
[我的空间](#)
[我的短信](#)
[我的评论](#)
[更多链接](#)

我的标签

[JSF\(1\)](#)

[博客园](#) [首页](#) [社区](#) [新随笔](#) [联系](#) [订阅](#) [XML](#) [管理](#)

随笔-245 评论-77 文章-1 trackbacks-3

maven

maven2 起步

相信maven1 大家都已经很熟悉了,具体maven能做什么,就不详细说了.个人觉得maven在开源项目中用的还是比较多的,公司内部,就不太清楚了.我以前的公司用过一段时间,不过后来就没有下文了。

与maven1 相比, maven2可算是几乎重写了,不过从速度来说应该更快。

主要的几个新特性包括: (详细参考<http://www.ibm.com/developerworks/cn/opensource/os-maven2/index.html>)

1. 更快、更简单

速度方面可以比上ant了

2. 更少的配置文件

现在的配置文件只剩下了settings.xml和pom.xml了。

3. Plugin语言更换

语言开始支持java, BeanShell和ant

4. 提供了预定义的模版

这点是最有帮助的, 用户可以自己定义自己的项目模版了, 就像用appfuse一样生成项目结构

5. 生命周期的引入

在Maven2中有了明确的生命周期概念, 而且都提供与之对应的命令, 使得项目构建更加清晰明了。

6. 新增Dependency Scope

这点也比较重要, 有些用于test范围的包, 可以不用加入依赖了

7. 传递依赖, 简化依赖管理

这是最为方便的, 可以省了很多配置. 如a 依赖 b, b 依赖c 默认 a也会依赖 c。但是也会带来隐患, 如版本冲突。不过maven 也已经考虑到了, 可以使用exclusions来排除相应的重复依赖

介绍了那么多, 现在切入正题, 开始maven2 之旅:

首些下载需要的工具:

[Proxy server 缓存 jsp html\(1\)](#)

随笔分类

[Asp.net\(8\)](#) 

[C#\(2\)](#) 

[DataBase\(18\)](#) 

[EasyJWeb](#) 

[JAVA\(37\)](#) 

[MAVEN\(2\)](#) 

[MYOB Develop\(3\)](#) 

[Other\(52\)](#) 

[Portal-Portlet\(15\)](#) 

[SubVersion\(4\)](#) 

随笔档案

[2009年10月 \(1\)](#)

[2009年8月 \(2\)](#)

[2009年7月 \(3\)](#)

[2009年2月 \(2\)](#)

[2008年12月 \(3\)](#)

[2008年11月 \(3\)](#)

[2008年10月 \(3\)](#)

[2008年9月 \(2\)](#)

[2008年7月 \(5\)](#)

[2008年6月 \(8\)](#)

[2008年5月 \(3\)](#)

[2008年4月 \(2\)](#)

[2008年3月 \(4\)](#)

[2008年2月 \(6\)](#)

[2008年1月 \(13\)](#)

[2007年12月 \(13\)](#)

[2007年8月 \(23\)](#)

[2007年7月 \(30\)](#)

maven2: <http://maven.apache.org/download.html> 最主要的

maven-proxy: 用来代理repository, 使用本地库代替maven2的远程库

<http://maven-proxy.codehaus.org/>

continuum: 一个不错的持续整合工具, 用于自动build。支持ant,maven

<http://maven.apache.org/continuum/>

svn: 版本控制工具相信都已经配置了。

maven 用于eclipse的插件, 在maven主站有下载, 不错的插件。当然idea也有相应的插件

最后, http, 服务器是必不可少的。用于内部开发使用。

可以使用apache, 或者jetty <http://www.mortbay.org/>

安装:

安装maven2很简单, 把下载来的maven包解开就行了。(目前我的配置都在win2003上, 还没有应用于linux, 所有的配置都针对 windows). 增加相应环境变量m2_home=maven2的安装目录, 不要忘了设置java_home的目录。

另外在path中增加% m2_home%\bin; 可以直接在命令行下面使用mvn。

其他工具的安装在后续的文章会介绍。

开始第一个maven2项目:

```
mvn archetype:create -DgroupId=com.mycompany.app \
-DartifactId=my-app
```

简单介绍一下 groupId相当于你的组织, 如同org.springframework, 会转化为相应本地路径 artifactId, 你主要的jar包名称, 也就是你要打成的jar 名称。

编译应用资源

```
mvn compile
```

编译相应的java 文件

编译测试类以及运行测试类

```
mvn test
```

运行测试类

如果只想编译test, 执行

```
mvn test-compile
```

打包和安装你的本地库

打包:

```
mvn package
```

安装:

```
mvn install
```

创建web site

```
mvn site
```

清除所有输出

```
mvn clean
```

创建相关的ide文件

```
mvn idea:idea 或者 mvn eclipse:eclipse
```

顺便说一下, maven2 是有生命周期这一概念的, 也就是说如果你执行package, 相应的以前步骤, 如compile,test等

[2007年6月 \(17\)](#)

[2007年5月 \(8\)](#)

[2007年4月 \(3\)](#)

[2007年3月 \(1\)](#)

[2007年1月 \(21\)](#)

[2006年12月 \(6\)](#)

[2006年11月 \(8\)](#)

[2006年10月 \(3\)](#)

[2006年9月 \(15\)](#)


[2006年8月 \(1\)](#)

[2006年7月 \(7\)](#)

[2006年6月 \(19\)](#)

[2006年5月 \(9\)](#)

文章分类

[JAVA study](#) 

文章档案

[2006年9月 \(1\)](#)

相册

[NANSHA](#)

[蒲州花园](#)

[三亚](#)

收藏夹

[myblog\(14\)](#) 

SUN PORTAL

[SUN PORTAL](#)

最新随笔

[1. 关于忘记MySQL的root用户密码](#)

都会自动执行。

刚开始执行会比较慢，需要从maven2远程库中下载所有的文件到本地。如果你的本地没有相应的依赖包，则每次maven都会去远程下载，所以配置一个镜像库就比较重要了。

另外介绍一下主要的参考资料：

mavn2 主站：主要的pom和settings.xml参考资料

Better Builds with Maven http://www.mergere.com/m2book_download.jsp

不错的书，主要通过例子介绍。可惜都是E文的，花点时间还是值得的。

下一个主题，会说一下maven2的主要配置。

maven 配置篇 之 settings.xml

maven2 比起maven1 来说，需要配置的文件少多了，主要集中在pom.xml和settings.xml中。

先来说说settings.xml，settings.xml对于maven来说相当于全局性的配置，用于所有的项目。在maven2中存在两个settings.xml，一个位于maven2的安装目录conf下面，作为全局性配置。对于团队设置，保持一致的定义是关键，所以maven2/conf下面的settings.xml就作为团队共同的配置文件。保证所有的团队成员都拥有相同的配置。当然对于每个成员，都需要特殊的 自定义设置，如用户信息，所以另外一个settings.xml就作为本地配置。默认的位置为：\${user.dir} /.m2/settings.xml目录中（\${user.dir} 指windows 中的用户目录）。

settings.xml基本结构如下：

```
<settings xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <localRepository/>
  <interactiveMode/>
  <usePluginRegistry/>
  <offline/>
  <pluginGroups/>
  <servers/>
  <mirrors/>
  <proxies/>
  <profiles/>
  <activeProfiles/>
</settings>
```

简单介绍一下几个主要的配置因素：

localRepository：表示本地库的保存位置，也就是maven2主要的jar保存位置，

默认在\${user.dir}/.m2/repository，如果需要另外设置，就换成其他的路径。

offline：如果不想每次编译，都去查找远程中心库，那就设置为true。当然前提是你已经下载了必须的依赖包。

Servers

在POM中的 distributionManagement元素定义了开发库。然而，特定的username和pwd不能用于pom.xml，所以通过此配置来保存server信息

```
<servers>
```

的问题

2. 利用JDK1.5的工具对远程的Java应用程序进行监测 (摘录)

3. Visualvm 远程测试 问题

4. MYOB 的一些开发资料

5. 连接MYOB ODBC,在MyEclipse下Commit成功,在Tomcat下单独运行,Commit显示Connection 已经关闭

6. MYOB Developer Jave 创建连接

7. tomcat dbcp jndi 配置

8. JAVA 事务处理

9. jsp中文件下载的实现

10. META Header

最新评论 XML

1. Re:Eclipse RCP开发桌面程序

太厉害了 照着你的提示我设计出了第一个菜单 第二个那个我没看过 有点难啊 呵呵 我想问一个问题: 我看的Java书编的桌面程序用的方法好像和 Eclipse用的方法不一样啊 例如用F...

---奔跑~

2. Re:Eclipse RCP开发桌面程序

高手 我是初学者 还请多指教 能留个QQ吗? 我的是 1136005852 谢啦

---奔跑~

3. Re:通过Dom4j读写XML文档

一堆笑脸,请楼主以代码的方式显示代码!

--CoderDream

4. Re:XML特殊符号

tkS

--天怒蒿

5. Re:Eclipse RCP开发桌面程序

谢谢

--lijian1111

阅读排行榜

```
<server>
  <id>server001</id>
  <username>my_login</username>
  <password>my_password</password>
  <privateKey>${usr.home}/.ssh/id_dsa</privateKey>
  <passphrase>some_passphrase</passphrase>
  <filePermissions>664</filePermissions>
  <directoryPermissions>775</directoryPermissions>
  <configuration></configuration>
</server>
</servers>
```

id:server 的id,用于匹配distributionManagement库id, 比较重要。

username, password: 用于登陆此服务器的用户名和密码

privateKey, passphrase: 设置private key, 以及passphrase

filePermissions, directoryPermissions: 当库文件或者目录创建后, 需要使用权限进行访问。参照unix文件许可, 如664和775

Mirrors

表示镜像库, 指定库的镜像, 用于增加其他库

```
<mirrors>
  <mirror>
    <id>planetmirror.com</id>
    <name>PlanetMirror Australia</name>
    <url>http://downloads.planetmirror.com/pub/maven2</url>
    <mirrorOf>central</mirrorOf>
  </mirror>
</mirrors>
```

id,name: 唯一的标志, 用于区别镜像

url: 镜像的url

mirrorOf: 此镜像指向的服务id

Proxies

此设置, 主要用于无法直接访问中心的库用户配置。

```
<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>
```

1. [shell 字符串操作\(11085\)](#)
2. [通过Dom4j读写XML文档\(6466\)](#)
3. [Eclipse RCP开发桌面程序\(4903\)](#)
4. [腰部肌肉锻炼\(4750\)](#)
5. [Eclipse中CVS版本管理\(4466\)](#)

评论排行榜

1. [vb 托盘图标\(8\)](#)
2. [ASP.NET 2.0 中Login控件的使用\(6\)](#)
3. [Eclipse RCP开发桌面程序\(6\)](#)
4. [用Java语句判断数据库表是否存在\(5\)](#)
5. [Linux下安装OpenOffice\(5\)](#)

```
</proxies>
id: 代理的标志
active: 是否激活代理
protocol, host, port: protocol://host:port 代理
username, password: 用户名和密码
nonProxyHosts: 不需要代理的host
```

Profiles

类似于pom.xml中的profile元素，主要包括activation, repositories, pluginRepositories 和properties元素。刚开始接触的时候，可能会比较迷惑，其实这是maven2中比较强大的功能。从字面上来说，就是个性配置。单独定义profile后，并不会生效，需要通过满足条件来激活。

repositories 和 pluginRepositories

定义其他开发库和插件开发库。对于团队来说，肯定有自己的开发库。可以通过此配置来定义。如下的配置，定义了本地开发库，用于release 发布。

```
<repositories>
  <repository>
    <id>repo-local</id>
    <name>Internal 开发库</name>
    <url>http://192.168.0.2:8082/repo-local</url>
    <releases>
      <enabled>true</enabled>
      <updatePolicy>never</updatePolicy>
      <checksumPolicy>warn</checksumPolicy>
    </releases>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
    <layout>default</layout>
  </repository>
</repositories>
<pluginRepositories>
  <pluginRepository>
    <id>repo-local</id>
    <name>Internal 开发库</name>
    <url>http://192.168.0.2:8082/repo-local</url>
    <releases>
      <enabled>true</enabled>
      <updatePolicy>never</updatePolicy>
      <checksumPolicy>warn</checksumPolicy>
    </releases>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
```

```
<layout>default</layout>
</pluginRepository>
</pluginRepositories>
```

releases, snapshots: 每个产品的版本的Release或者snapshot(注: release和snapshot的区别, release一般是比较稳定的版本, 而snapshot基本上不稳定, 只是作为快照)

properties

maven 的properties作为placeholder值, 如ant的properties。

包括以下的5种类型值:

1. env.X, 返回当前的环境变量
2. project.x: 返回pom中定义的元素值, 如project.version
3. settings.x: 返回settings.xml中定义的元素
4. java 系统属性: 所有经过java.lang.System.getProperties() 返回的值
5. x: 用户自己设定的值

Activation

用于激活此profile

```
<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>
```

jdk: 如果匹配指定的jdk版本, 将会激活

os: 操作系统

property: 如果maven能检测到相应的属性

file: 用于判断文件是否存在或者不存在

除了使用activation来激活profile, 同样可以通过activeProfiles来激活

Active Profiles

表示激活的profile, 通过profile id来指定。

```
<activeProfiles>
  <activeProfile>env-test</activeProfile> 指定的profile id
</activeProfiles>
```

maven 配置篇 之pom.xml

说完了settings.xml配置，下来说一下maven2的主要配置pom.xml

什么是pom?

pom作为项目对象模型。通过xml表示maven项目，使用pom.xml来实现。主要描述了项目：包括配置文件；开发者需要遵循的规则，缺陷管理系统，组织和licenses，项目的url，项目的依赖性，以及其他所有的项目相关因素。

快速察看：

```
<project>
  <modelVersion>4.0.0</modelVersion>

  <!-- The Basics -->
  <groupId>...</groupId>
  <artifactId>...</artifactId>
  <version>...</version>
  <packaging>...</packaging>
  <dependencies>...</dependencies>
  <parent>...</parent>
  <dependencyManagement>...</dependencyManagement>
  <modules>...</modules>
  <properties>...</properties>

  <!-- Build Settings -->
  <build>...</build>
  <reporting>...</reporting>

  <!-- More Project Information -->
  <name>...</name>
  <description>...</description>
  <url>...</url>
  <inceptionYear>...</inceptionYear>
  <licenses>...</licenses>
  <organization>...</organization>
  <developers>...</developers>
  <contributors>...</contributors>

  <!-- Environment Settings -->
```

```
<issueManagement>...</issueManagement>
<ciManagement>...</ciManagement>
<mailingLists>...</mailingLists>
<scm>...</scm>
<prerequisites>...</prerequisites>
<repositories>...</repositories>
<pluginRepositories>...</pluginRepositories>
<distributionManagement>...</distributionManagement>
<profiles>...</profiles>
</project>
```

基本内容:

POM包括了所有的项目信息。

maven 相关:

pom定义了最小的maven2元素, 允许groupId,artifactId,version。所有需要的元素

groupId: 项目或者组织的唯一标志, 并且配置时生成的路径也是由此生成, 如org.codehaus.mojo生成的相对路径为: /org/codehaus/mojo

artifactId: 项目的通用名称

version: 项目的版本

packaging: 打包的机制, 如pom, jar, maven-plugin, ejb, war, ear, rar, par

classifier: 分类

POM关系:

主要为依赖, 继承, 合成

依赖关系:

```
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.0</version>
<type>jar</type>
<scope>test</scope>
<optional>>true</optional>
</dependency>
...
</dependencies>
```

groupId, artifactId, version: 描述了依赖的项目唯一标志

可以通过 ([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]) ([32]) ([33]) ([34]) ([35]) ([36]) ([37]) ([38]) ([39]) ([40]) ([41]) ([42]) ([43]) ([44]) ([45]) ([46]) ([47]) ([48]) ([49]) ([50]) ([51]) ([52]) ([53]) ([54]) ([55]) ([56]) ([57]) ([58]) ([59]) ([60]) ([61]) ([62]) ([63]) ([64]) ([65]) ([66]) ([67]) ([68]) ([69]) ([70]) ([71]) ([72]) ([73]) ([74]) ([75]) ([76]) ([77]) ([78]) ([79]) ([80]) ([81]) ([82]) ([83]) ([84]) ([85]) ([86]) ([87]) ([88]) ([89]) ([90]) ([91]) ([92]) ([93]) ([94]) ([95]) ([96]) ([97]) ([98]) ([99]) ([100]) ([101]) ([102]) ([103]) ([104]) ([105]) ([106]) ([107]) ([108]) ([109]) ([110]) ([111]) ([112]) ([113]) ([114]) ([115]) ([116]) ([117]) ([118]) ([119]) ([120]) ([121]) ([122]) ([123]) ([124]) ([125]) ([126]) ([127]) ([128]) ([129]) ([130]) ([131]) ([132]) ([133]) ([134]) ([135]) ([136]) ([137]) ([138]) ([139]) ([140]) ([141]) ([142]) ([143]) ([144]) ([145]) ([146]) ([147]) ([148]) ([149]) ([150]) ([151]) ([152]) ([153]) ([154]) ([155]) ([156]) ([157]) ([158]) ([159]) ([160]) ([161]) ([162]) ([163]) ([164]) ([165]) ([166]) ([167]) ([168]) ([169]) ([170]) ([171]) ([172]) ([173]) ([174]) ([175]) ([176]) ([177]) ([178]) ([179]) ([180]) ([181]) ([182]) ([183]) ([184]) ([185]) ([186]) ([187]) ([188]) ([189]) ([190]) ([191]) ([192]) ([193]) ([194]) ([195]) ([196]) ([197]) ([198]) ([199]) ([200]) ([201]) ([202]) ([203]) ([204]) ([205]) ([206]) ([207]) ([208]) ([209]) ([210]) ([211]) ([212]) ([213]) ([214]) ([215]) ([216]) ([217]) ([218]) ([219]) ([220]) ([221]) ([222]) ([223]) ([224]) ([225]) ([226]) ([227]) ([228]) ([229]) ([230]) ([231]) ([232]) ([233]) ([234]) ([235]) ([236]) ([237]) ([238]) ([239]) ([240]) ([241]) ([242]) ([243]) ([244]) ([245]) ([246]) ([247]) ([248]) ([249]) ([250]) ([251]) ([252]) ([253]) ([254]) ([255]) ([256]) ([257]) ([258]) ([259]) ([260]) ([261]) ([262]) ([263]) ([264]) ([265]) ([266]) ([267]) ([268]) ([269]) ([270]) ([271]) ([272]) ([273]) ([274]) ([275]) ([276]) ([277]) ([278]) ([279]) ([280]) ([281]) ([282]) ([283]) ([284]) ([285]) ([286]) ([287]) ([288]) ([289]) ([290]) ([291]) ([292]) ([293]) ([294]) ([295]) ([296]) ([297]) ([298]) ([299]) ([300]) ([301]) ([302]) ([303]) ([304]) ([305]) ([306]) ([307]) ([308]) ([309]) ([310]) ([311]) ([312]) ([313]) ([314]) ([315]) ([316]) ([317]) ([318]) ([319]) ([320]) ([321]) ([322]) ([323]) ([324]) ([325]) ([326]) ([327]) ([328]) ([329]) ([330]) ([331]) ([332]) ([333]) ([334]) ([335]) ([336]) ([337]) ([338]) ([339]) ([340]) ([341]) ([342]) ([343]) ([344]) ([345]) ([346]) ([347]) ([348]) ([349]) ([350]) ([351]) ([352]) ([353]) ([354]) ([355]) ([356]) ([357]) ([358]) ([359]) ([360]) ([361]) ([362]) ([363]) ([364]) ([365]) ([366]) ([367]) ([368]) ([369]) ([370]) ([371]) ([372]) ([373]) ([374]) ([375]) ([376]) ([377]) ([378]) ([379]) ([380]) ([381]) ([382]) ([383]) ([384]) ([385]) ([386]) ([387]) ([388]) ([389]) ([390]) ([391]) ([392]) ([393]) ([394]) ([395]) ([396]) ([397]) ([398]) ([399]) ([400]) ([401]) ([402]) ([403]) ([404]) ([405]) ([406]) ([407]) ([408]) ([409]) ([410]) ([411]) ([412]) ([413]) ([414]) ([415]) ([416]) ([417]) ([418]) ([419]) ([420]) ([421]) ([422]) ([423]) ([424]) ([425]) ([426]) ([427]) ([428]) ([429]) ([430]) ([431]) ([432]) ([433]) ([434]) ([435]) ([436]) ([437]) ([438]) ([439]) ([440]) ([441]) ([442]) ([443]) ([444]) ([445]) ([446]) ([447]) ([448]) ([449]) ([450]) ([451]) ([452]) ([453]) ([454]) ([455]) ([456]) ([457]) ([458]) ([459]) ([460]) ([461]) ([462]) ([463]) ([464]) ([465]) ([466]) ([467]) ([468]) ([469]) ([470]) ([471]) ([472]) ([473]) ([474]) ([475]) ([476]) ([477]) ([478]) ([479]) ([480]) ([481]) ([482]) ([483]) ([484]) ([485]) ([486]) ([487]) ([488]) ([489]) ([490]) ([491]) ([492]) ([493]) ([494]) ([495]) ([496]) ([497]) ([498]) ([499]) ([500]) ([501]) ([502]) ([503]) ([504]) ([505]) ([506]) ([507]) ([508]) ([509]) ([510]) ([511]) ([512]) ([513]) ([514]) ([515]) ([516]) ([517]) ([518]) ([519]) ([520]) ([521]) ([522]) ([523]) ([524]) ([525]) ([526]) ([527]) ([528]) ([529]) ([530]) ([531]) ([532]) ([533]) ([534]) ([535]) ([536]) ([537]) ([538]) ([539]) ([540]) ([541]) ([542]) ([543]) ([544]) ([545]) ([546]) ([547]) ([548]) ([549]) ([550]) ([551]) ([552]) ([553]) ([554]) ([555]) ([556]) ([557]) ([558]) ([559]) ([560]) ([561]) ([562]) ([563]) ([564]) ([565]) ([566]) ([567]) ([568]) ([569]) ([570]) ([571]) ([572]) ([573]) ([574]) ([575]) ([576]) ([577]) ([578]) ([579]) ([580]) ([581]) ([582]) ([583]) ([584]) ([585]) ([586]) ([587]) ([588]) ([589]) ([590]) ([591]) ([592]) ([593]) ([594]) ([595]) ([596]) ([597]) ([598]) ([599]) ([600]) ([601]) ([602]) ([603]) ([604]) ([605]) ([606]) ([607]) ([608]) ([609]) ([610]) ([611]) ([612]) ([613]) ([614]) ([615]) ([616]) ([617]) ([618]) ([619]) ([620]) ([621]) ([622]) ([623]) ([624]) ([625]) ([626]) ([627]) ([628]) ([629]) ([630]) ([631]) ([632]) ([633]) ([634]) ([635]) ([636]) ([637]) ([638]) ([639]) ([640]) ([641]) ([642]) ([643]) ([644]) ([645]) ([646]) ([647]) ([648]) ([649]) ([650]) ([651]) ([652]) ([653]) ([654]) ([655]) ([656]) ([657]) ([658]) ([659]) ([660]) ([661]) ([662]) ([663]) ([664]) ([665]) ([666]) ([667]) ([668]) ([669]) ([670]) ([671]) ([672]) ([673]) ([674]) ([675]) ([676]) ([677]) ([678]) ([679]) ([680]) ([681]) ([682]) ([683]) ([684]) ([685]) ([686]) ([687]) ([688]) ([689]) ([690]) ([691]) ([692]) ([693]) ([694]) ([695]) ([696]) ([697]) ([698]) ([699]) ([700]) ([701]) ([702]) ([703]) ([704]) ([705]) ([706]) ([707]) ([708]) ([709]) ([710]) ([711]) ([712]) ([713]) ([714]) ([715]) ([716]) ([717]) ([718]) ([719]) ([720]) ([721]) ([722]) ([723]) ([724]) ([725]) ([726]) ([727]) ([728]) ([729]) ([730]) ([731]) ([732]) ([733]) ([734]) ([735]) ([736]) ([737]) ([738]) ([739]) ([740]) ([741]) ([742]) ([743]) ([744]) ([745]) ([746]) ([747]) ([748]) ([749]) ([750]) ([751]) ([752]) ([753]) ([754]) ([755]) ([756]) ([757]) ([758]) ([759]) ([760]) ([761]) ([762]) ([763]) ([764]) ([765]) ([766]) ([767]) ([768]) ([769]) ([770]) ([771]) ([772]) ([773]) ([774]) ([775]) ([776]) ([777]) ([778]) ([779]) ([780]) ([781]) ([782]) ([783]) ([784]) ([785]) ([786]) ([787]) ([788]) ([789]) ([790]) ([791]) ([792]) ([793]) ([794]) ([795]) ([796]) ([797]) ([798]) ([799]) ([800]) ([801]) ([802]) ([803]) ([804]) ([805]) ([806]) ([807]) ([808]) ([809]) ([810]) ([811]) ([812]) ([813]) ([814]) ([815]) ([816]) ([817]) ([818]) ([819]) ([820]) ([821]) ([822]) ([823]) ([824]) ([825]) ([826]) ([827]) ([828]) ([829]) ([830]) ([831]) ([832]) ([833]) ([834]) ([835]) ([836]) ([837]) ([838]) ([839]) ([840]) ([841]) ([842]) ([843]) ([844]) ([845]) ([846]) ([847]) ([848]) ([849]) ([850]) ([851]) ([852]) ([853]) ([854]) ([855]) ([856]) ([857]) ([858]) ([859]) ([860]) ([861]) ([862]) ([863]) ([864]) ([865]) ([866]) ([867]) ([868]) ([869]) ([870]) ([871]) ([872]) ([873]) ([874]) ([875]) ([876]) ([877]) ([878]) ([879]) ([880]) ([881]) ([882]) ([883]) ([884]) ([885]) ([886]) ([887]) ([888]) ([889]) ([890]) ([891]) ([892]) ([893]) ([894]) ([895]) ([896]) ([897]) ([898]) ([899]) ([900]) ([901]) ([902]) ([903]) ([904]) ([905]) ([906]) ([907]) ([908]) ([909]) ([910]) ([911]) ([912]) ([913]) ([914]) ([915]) ([916]) ([917]) ([918]) ([919]) ([920]) ([921]) ([922]) ([923]) ([924]) ([925]) ([926]) ([927]) ([928]) ([929]) ([930]) ([931]) ([932]) ([933]) ([934]) ([935]) ([936]) ([937]) ([938]) ([939]) ([940]) ([941]) ([942]) ([943]) ([944]) ([945]) ([946]) ([947]) ([948]) ([949]) ([950]) ([951]) ([952]) ([953]) ([954]) ([955]) ([956]) ([957]) ([958]) ([959]) ([960]) ([961]) ([962]) ([963]) ([964]) ([965]) ([966]) ([967]) ([968]) ([969]) ([970]) ([971]) ([972]) ([973]) ([974]) ([975]) ([976]) ([977]) ([978]) ([979]) ([980]) ([981]) ([982]) ([983]) ([984]) ([985]) ([986]) ([987]) ([988]) ([989]) ([990]) ([991]) ([992]) ([993]) ([994]) ([995]) ([996]) ([997]) ([998]) ([999])

- 使用以下的命令安装:

- mvn install:install-file -Dfile=non-maven-proj.jar -DgroupId=some.group -DartifactId=non-maven-proj -Dversion=1

- 创建自己的库,并配置,使用deploy:deploy-file

- 设置此依赖范围为`system`，定义一个系统路径。不提倡。

`type`: 相应的依赖产品包形式，如`jar`，`war`

`scope`: 用于限制相应的依赖范围，包括以下的几种变量：

- `compile`：默认范围，用于编译
- `provided`：类似于编译，但支持你期待`jdk`或者容器提供，类似于`classpath`
- `runtime`: 在执行时，需要使用
- `test`: 用于`test`任务时使用
- `system`: 需要外在提供相应得元素。通过`systemPath`来取得

`systemPath`: 仅用于范围为`system`。提供相应的路径

`optional`: 标注可选，当项目自身也是依赖时。用于连续依赖时使用

独占性

外在告诉`maven`你只包括指定的项目，不包括相关的依赖。此因素主要用于解决版本冲突问题

```
<dependencies>
<dependency>
  <groupId>org.apache.maven</groupId>
  <artifactId>maven-embedder</artifactId>
  <version>2.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.apache.maven</groupId>
      <artifactId>maven-core</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

表示项目`maven-embedder`需要项目`maven-core`，但我们不想引用`maven-core`

继承关系

另一个强大的变化,`maven`带来的是项目继承。主要的设置:

定义父项目

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>my-parent</artifactId>
  <version>2.0</version>
  <packaging>pom</packaging>
</project>
```

`packaging` 类型，需要`pom`用于`parent`和合成多个项目。我们需要增加相应的值给父`pom`，用于子项目继承。主要的元素如下：

依赖型

开发者和合作者

插件列表

报表列表

插件执行使用相应的匹配ids

插件配置

子项目配置

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>my-parent</artifactId>
    <version>2.0</version>
    <relativePath>../my-parent</relativePath>
  </parent>
  <artifactId>my-project</artifactId>
</project>
```

relativePath可以不需要，但是用于指明parent的目录，用于快速查询。

dependencyManagement :

用于父项目配置共同的依赖关系，主要配置依赖包相同因素，如版本，scope。

合成（或者多个模块）

一个项目有多个模块，也叫做多重模块，或者合成项目。

如下的定义：

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>my-parent</artifactId>
  <version>2.0</version>
  <modules>
    <module>my-project1</module>
    <module>my-project2</module>
  </modules>
</project>
```

build 设置

主要用于编译设置，包括两个主要的元素，build和report

build

主要分为两部分，基本元素和扩展元素集合

注意：包括项目build和profile build

```
<project>
  <!-- "Project Build" contains more elements than just the BaseBuild set -->
  <build>...</build>
  <profiles>
```

```

    <profile>
      <!-- "Profile Build" contains a subset of "Project Build"s elements -->
      <build>...</build>
    </profile>
  </profiles>
</project>

```

基本元素

```

<build>
  <defaultGoal>install</defaultGoal>
  <directory>${basedir}/target</directory>
  <finalName>${artifactId}-${version}</finalName>
  <filters>
    <filter>filters/filter1.properties</filter>
  </filters>
  ...
</build>

```

defaultGoal: 定义默认的目标或者阶段。如install

directory: 编译输出的目录

finalName: 生成最后的文件的样式

filter: 定义过滤，用于替换相应的属性文件，使用maven定义的属性。设置所有placeholder的值

资源(resources)

你项目中需要指定的资源。如spring配置文件,log4j.properties

```

<project>
  <build>
    ...
    <resources>
      <resource>
        <targetPath>META-INF/plexus</targetPath>
        <filtering>>false</filtering>
        <directory>${basedir}/src/main/plexus</directory>
        <includes>
          <include>configuration.xml</include>
        </includes>
        <excludes>
          <exclude>**/*.properties</exclude>
        </excludes>
      </resource>
    </resources>
    <testResources>
      ...
    </testResources>
  </build>
</project>

```

```
</testResources>
```

```
...
```

```
</build>
```

```
</project>
```

resources: resource的列表, 用于包括所有的资源

targetPath: 指定目标路径, 用于放置资源, 用于build

filtering: 是否替换资源中的属性placeholder

directory: 资源所在的位置

includes: 样式, 包括那些资源

excludes: 排除的资源

testResources: 测试资源列表

插件

在build时, 执行的插件, 比较有用的部分, 如使用jdk 5.0编译等等

```
<project>
```

```
<build>
```

```
...
```

```
<plugins>
```

```
<plugin>
```

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

```
<artifactId>maven-jar-plugin</artifactId>
```

```
<version>2.0</version>
```

```
<extensions>>false</extensions>
```

```
<inherited>>true</inherited>
```

```
<configuration>
```

```
<classifier>test</classifier>
```

```
</configuration>
```

```
<dependencies>...</dependencies>
```

```
<executions>...</executions>
```

```
</plugin>
```

```
</plugins>
```

```
</build>
```

```
</project>
```

extensions: true or false, 是否装载插件扩展。默认false

inherited: true or false, 是否此插件配置将会应用于poms, 那些继承于此的项目

configuration: 指定插件配置

dependencies: 插件需要依赖的包

executions: 用于配置execution目标, 一个插件可以有多个目标。

如下:

```
<plugin>
```

```
<artifactId>maven-antrun-plugin</artifactId>
```

```
<executions>
  <execution>
    <id>echodir</id>
    <goals>
      <goal>run</goal>
    </goals>
    <phase>verify</phase>
    <inherited>false</inherited>
    <configuration>
      <tasks>
        <echo>Build Dir: ${project.build.directory}</echo>
      </tasks>
    </configuration>
  </execution>
</executions>
</plugin>
```

说明:

id: 规定`execution` 的唯一标志

goals: 表示目标

phase: 表示阶段, 目标将会在什么阶段执行

inherited: 和上面的元素一样, 设置`false` `maven`将会拒绝执行继承给子插件

configuration: 表示此执行的配置属性

插件管理

pluginManagement: 插件管理以同样的方式包括插件元素, 用于在特定的项目中配置。所有继承于此项目的子项目都能使用。主要定义插件的共同元素

扩展元素集合

主要包括以下的元素:

Directories

用于设置各种目录结构, 如下:

```
<build>
  <sourceDirectory>${basedir}/src/main/java</sourceDirectory>
  <scriptSourceDirectory>${basedir}/src/main/scripts</scriptSourceDirectory>
  <testSourceDirectory>${basedir}/src/test/java</testSourceDirectory>
  <outputDirectory>${basedir}/target/classes</outputDirectory>
  <testOutputDirectory>${basedir}/target/test-classes</testOutputDirectory>
  ...
</build>
```

Extensions

表示需要扩展的插件，必须包括进相应的**build**路径。

```
<project>
  <build>
    ...
    <extensions>
      <extension>
        <groupId>org.apache.maven.wagon</groupId>
        <artifactId>wagon-ftp</artifactId>
        <version>1.0-alpha-3</version>
      </extension>
    </extensions>
    ...
  </build>
</project>
```

Reporting

用于在site阶段输出报表。特定的maven 插件能输出相应的定制和配置报表。

```
<reporting>
  <plugins>
    <plugin>
      <outputDirectory>${basedir}/target/site</outputDirectory>
      <artifactId>maven-project-info-reports-plugin</artifactId>
      <reportSets>
        <reportSet></reportSet>
      </reportSets>
    </plugin>
  </plugins>
</reporting>
```

Report Sets

用于配置不同的目标，应用于不同的报表

```
<reporting>
  <plugins>
    <plugin>
      ...
      <reportSets>
        <reportSet>
          <id>sunlink</id>
          <reports>
            <report>javadoc</report>
          </reports>
          <inherited>true</inherited>
        </reportSet>
      </reportSets>
    </plugin>
  </plugins>
</reporting>
```

```
<configuration>
  <links>
    <link>http://java.sun.com/j2se/1.5.0/docs/api/</link>
  </links>
</configuration>
</reportSet>
</reportSets>
</plugin>
</plugins>
</reporting>
```

更多的项目信息

name: 项目除了artifactId外，可以定义多个名称

description: 项目描述

url: 项目url

inceptionYear: 创始年份

Licenses

```
<licenses>
  <license>
    <name>Apache 2</name>
    <url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>
    <distribution>repo</distribution>
    <comments>A business-friendly OSS license</comments>
  </license>
</licenses>
```

Organization

配置组织信息

```
<organization>
  <name>Codehaus Mojo</name>
  <url>http://mojo.codehaus.org</url>
</organization>
```

Developers

配置开发者信息

```
<developers>
  <developer>
    <id>eric</id>
    <name>Eric</name>
    <email>eredmond@codehaus.org</email>
    <url>http://eric.propellors.net</url>
    <organization>Codehaus</organization>
```

```
<organizationUrl>http://mojo.codehaus.org</organizationUrl>
<roles>
  <role>architect</role>
  <role>developer</role>
</roles>
<timezone>-6</timezone>
<properties>
  <picUrl>http://tinyurl.com/prv4t</picUrl>
</properties>
</developer>
</developers>
```

Contributors

```
<contributors>
<contributor>
  <name>Noelle</name>
  <email>some.name@gmail.com</email>
  <url>http://noellemarie.com</url>
  <organization>Noelle Marie</organization>
  <organizationUrl>http://noellemarie.com</organizationUrl>
  <roles>
    <role>tester</role>
  </roles>
  <timezone>-5</timezone>
  <properties>
    <gtalk>some.name@gmail.com</gtalk>
  </properties>
</contributor>
</contributors>
```

环境设置

Issue Management

定义相关的bug跟踪系统，如bugzilla, testtrack, clearQuest 等

```
<issueManagement>
  <system>Bugzilla</system>
  <url>http://127.0.0.1/bugzilla</url>
</issueManagement>
```

Continuous Integration Management

连续整合管理，基于triggers或者timings

```
<ciManagement>
  <system>continuum</system>
  <url>http://127.0.0.1:8080/continuum</url>
```



```
<notifiers>
  <notifier>
    <type>mail</type>
    <sendOnError>true</sendOnError>
    <sendOnFailure>true</sendOnFailure>
    <sendOnSuccess>false</sendOnSuccess>
    <sendOnWarning>false</sendOnWarning>
    <configuration><address>continuum@127.0.0.1</address></configuration>
  </notifier>
</notifiers>
</ciManagement>
```

Mailing Lists

```
<mailingLists>
  <mailingList>
    <name>User List</name>
    <subscribe>user-subscribe@127.0.0.1</subscribe>
    <unsubscribe>user-unsubscribe@127.0.0.1</unsubscribe>
    <post>user@127.0.0.1</post>
    <archive>http://127.0.0.1/user/</archive>
    <otherArchives>
      <otherArchive>http://base.google.com/base/1/127.0.0.1</otherArchive>
    </otherArchives>
  </mailingList>
</mailingLists>
```

SCM

软件配置管理，如cvs 和svn

```
<scm>
  <connection>scm:svn:http://127.0.0.1/svn/my-project</connection>
  <developerConnection>scm:svn:https://127.0.0.1/svn/my-project</developerConnection>
  <tag>HEAD</tag>
  <url>http://127.0.0.1/websvn/my-project</url>
</scm>
```

Repositories

配置同setting.xml中的开发库

Plugin Repositories

配置同 repositories

Distribution Management

用于配置分发管理，配置相应的产品发布信息,主要用于发布，在执行mvn deploy后表示要发布的位置

1 配置到文件系统

```
<distributionManagement>
<repository>
<id>proficio-repository</id>
<name>Proficio Repository</name>
<url>file://${basedir}/target/deploy</url>
</repository>
</distributionManagement>
```

2 使用ssh2配置

```
<distributionManagement>
<repository>
<id>proficio-repository</id>
<name>Proficio Repository</name>
<url>scp://sshserver.yourcompany.com/deploy</url>
</repository>
</distributionManagement>
```

3 使用sftp配置

```
<distributionManagement>
<repository>
<id>proficio-repository</id>
<name>Proficio Repository</name>
<url>sftp://ftpserver.yourcompany.com/deploy</url>
</repository>
</distributionManagement>
```

4 使用外在的ssh配置

编译扩展用于指定使用wagon外在ssh提供，用于提供你的文件到相应的远程服务器。

```
<distributionManagement>
<repository>
<id>proficio-repository</id>
<name>Proficio Repository</name>
<url>scpexe://sshserver.yourcompany.com/deploy</url>
</repository>
</distributionManagement>
<build>
<extensions>
<extension>
<groupId>org.apache.maven.wagon</groupId>
<artifactId>wagon-ssh-external</artifactId>
<version>1.0-alpha-6</version>
</extension>
</extensions>
</build>
```

5 使用ftp配置

```
<distributionManagement>
<repository>
<id>proficio-repository</id>
<name>Proficio Repository</name>
<url>ftp://ftpserver.yourcompany.com/deploy</url>
</repository>
</distributionManagement>
<build>
<extensions>
<extension>
<groupId>org.apache.maven.wagon</groupId>
<artifactId>wagon-ftp</artifactId>
<version>1.0-alpha-6</version>
</extension>
</extensions>
</build>
```

repository 对应于你的开发库，用户信息通过settings.xml中的server取得

Profiles

类似于settings.xml中的profiles，增加了几个元素，如下的样式：

```
<profiles>
<profile>
<id>test</id>
<activation>...</activation>
<build>...</build>
<modules>...</modules>
<repositories>...</repositories>
<pluginRepositories>...</pluginRepositories>
<dependencies>...</dependencies>
<reporting>...</reporting>
<dependencyManagement>...</dependencyManagement>
<distributionManagement>...</distributionManagement>
</profile>
</profiles>
```

对于团队来说，建立统一的开发环境是必须的，而maven能很好帮助建立统一的环境。下面就介绍如何更有效的进行统一的配置。

准备工作：

下载必须的软件：

maven2: <http://maven.apache.org/download.html> 最主要的

maven-proxy: 用来代理repository，使用代理来访问多个远程库

<http://maven-proxy.codehaus.org/>

continuum: 一个不错的持续整合工具，用于自动build。支持ant,maven

<http://maven.apache.org/continuum/>

svn: 版本控制工具

创建一致的开发环境

在共享的开发环境中，更好的建议是保持maven的两个不同的配置文件分别管理，包括共享和用户自定义设置。共同的配置包括在安装目录中，而单独的开发设置保存在用户本地目录。

全局的配置文件**settings.xml**

```
<servers>
```

```
    //公司内部库，所有的release版本,serverid对应于repository id，用于在deploy时，访问使用，主要保存用户名和密码
```

```
<server>
```

```
<id>internal</id>
```

```
<username>${website.username}</username>
```

```
<password>${website.pwd}</password>
```

```
<filePermissions>664</filePermissions>
```

```
<directoryPermissions>775</directoryPermissions>
```

```
</server>
```

```
//目前的开发库，用于snapshot库
```

```
<server>
```

```
<id>snapshot</id>
```

```
<username>${website.username}</username>
```

```
<password>${website.pwd}</password>
```

```
<filePermissions>664</filePermissions>
```

```
<directoryPermissions>775</directoryPermissions>
```

```
</server>
```

```
</servers>
```

```
<profiles>
```

```
<!--定义核心库 maven 镜像,由maven-proxy实现-->
```

```
<profile>
```

```
<id>central-repo</id>
```

```
<repositories>
```

```
<repository>
```

```
<id>central</id>
```

```
<name>Internal Repository</name>
```

```
<url>http://192.168.0.2:9999/repository</url>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
<id>central</id>
<name>Internal Repository</name>
<url>http://192.168.0.2:9999/repository</url>
</pluginRepository>
</pluginRepositories>
</profile>
```

<!--定义内部库，包括公司的所有release版本-->

```
<profile>
<id>internal-repo</id>
<repositories>
<repository>
<id>internal</id>
<name>Internal Repository</name>
<url>http://192.168.0.2:8080/repo-local</url>
<releases>
<enabled>true</enabled>
<updatePolicy>never</updatePolicy>
<checksumPolicy>warn</checksumPolicy>
</releases>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
<id>internal</id>
<name>Internal Plugin Repository</name>
<url>http://192.168.0.2:8080/repo-local</url>
<releases>
<enabled>true</enabled>
<updatePolicy>never</updatePolicy>
<checksumPolicy>warn</checksumPolicy>
</releases>
</pluginRepository>
</pluginRepositories>
</profile>
```

<!--定义内部开发库，也可以合并snapshot和release-->

```
<profile>
<id>snapshot-repo</id>
<repositories>
<repository>
<id>snapshot</id>
<name>Internal Repository</name>
<url>http://192.168.0.2:8080/repo-snapshot</url>
<snapshots>
```

```

<enabled>true</enabled>
<updatePolicy>interval: 60</updatePolicy>
<checksumPolicy>warn</checksumPolicy>
</snapshots>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
<id>snapshot</id>
<name>Internal Plugin Repository</name>
<url>http://192.168.0.2:8080/repo-snapshot</url>
<snapshots>
<enabled>true</enabled>
<updatePolicy>interval: 60</updatePolicy>
<checksumPolicy>warn</checksumPolicy>
</snapshots>
</pluginRepository>
</pluginRepositories>
</profile>
</profiles>
<!-- 激活相应得配置-->
<activeProfiles>
<activeProfile>central-repo</activeProfile>
<activeProfile>internal-repo</activeProfile>
<activeProfile>snapshot-repo</activeProfile>
</activeProfiles>
<!-- 插件默认groupId -->
<pluginGroups>
<pluginGroup>com.mycompany.plugins</pluginGroup>
</pluginGroups>

```

包括了以下的共享因素：

服务器设置典型是共同的，只有用户名需要在用户环境中设置。使用一致的定义来配置共同的设置

profile定义了共同的因素，内部开发库，包括指定的组织或者部门发布的产品。这些库独立于核心开发库。

激活的**profiles**列表，用于激活相应的**profile**

plugin 组只有当你的组织中有自己定义的插件，用于命令行运行在**pom**中定义。

对于单独的用户来说，设置如下：

```

<settings>
<profiles>
<profile>
<id>property-overrides</id>
<properties>
<website.username>myuser</website.username>
<website.pwd>test</website.username>
</properties>

```

```
</profile>
</profiles>
</settings>
```

创建共享开发库

大多数组织将会创建自己的内部开发库，用于配置，而中心开发库用于连接maven

设置内部开发库是简单的，使用http协议，可以使用存在的http 服务器。或者创建新的服务，使用apache，或者jetty
假设服务器地址192.168.0.2 ,端口8080

http://192.168.0.2:8080/repo-local

设置另外一个开发库，用于设置项目的snapshot库http://192.168.0.2:8080/repo-snapshot

中心镜像库，使用maven-proxy创建，当然也可以创建自己的镜像。用于下载本地库中没有的artifact

maven-proxy 设置

从网上直接下载maven-proxy-standalone-0.2-app.jar和 proxy.properties

在命令行中，直接运行java -jar maven-proxy-standalone-0.2-app.jar proxy.properties

主要的配置：

设置repo.list 中增加相应的库就可以，如下定义：

repo.list=repo1.maven.org,...

#maven 的中心库

repo.repo1.maven.org.url=http://repo1.maven.org/maven2

repo.repo1.maven.org.description=maven.org

repo.repo1.maven.org.proxy=one

repo.repo1.maven.org.hardfail=false

repo.repo1.maven.org.cache.period=360000

repo.repo1.maven.org.cache.failures=true

以后所有的远程库，都通过此方式增加。顺便说一下，不要忘了注释原来的example，那是没有办法访问的。

其他配置如

端口号 port=9999

保存的位置 repo.local.store=target/repo

serverName=http://localhost:9999

创建标准的组织pom

定义共同的内容，包括公司的结构，如组织，部门以及团队。

察看一下maven 的自身，可以作为很好的参考。

如scm

```
<project>
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.apache.maven</groupId>
<artifactId>maven-parent</artifactId>
<version>1</version>
</parent>
<groupId>org.apache.maven.scm</groupId>
<artifactId>maven-scm</artifactId>
<url>http://maven.apache.org/maven-scm/</url>
...
<modules>
<module>maven-scm-api</module>
<module>maven-scm-providers</module>
...
</modules>
</project>
```

在maven父项目中可以看到如下定义：

```
<project>
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.apache</groupId>
<artifactId>apache</artifactId>
<version>1</version>
</parent>
<groupId>org.apache.maven</groupId>
<artifactId>maven-parent</artifactId>
<version>5</version>
<url>http://maven.apache.org/</url>
...
<mailingLists>
<mailingList>
<name>Maven Announcements List</name>
<post>announce@maven.apache.org</post>
...
</mailingList>
</mailingLists>
<developers>
<developer>
...
</developer>
</developers>
```



```
</project>
```

maven 父pom包括了共享的元素，如声明邮件列表，开发者。并且大多数项目继承apache组织：

```
<project>
<modelVersion>4.0.0</modelVersion>
<groupId>org.apache</groupId>
<artifactId>apache</artifactId>
<version>1</version>
<organization>
<name>Apache Software Foundation</name>
<url>http://www.apache.org/</url>
</organization>
<url>http://www.apache.org/</url>
...
<repositories>
<repository>
<id>apache.snapshots</id>
<name>Apache Snapshot Repository</name>
<url>http://svn.apache.org/maven-snapshot-repository</url>
<releases>
<enabled>false</enabled>
</releases>
</repository>
</repositories>
...
<distributionManagement>
<repository>
...
</repository>
<snapshotRepository>
...
</snapshotRepository>
</distributionManagement>
</project>
```

对于项目自身来说，父pom很少更新。所以，最后的方式保存父pom文件在单独的版本控制区域，它们能够check out，更改和配置。

使用**Continuum**持久整合

持续整合自动build你的项目，通过一定的时间，包括所有的冲突在早期察觉，而不是发布的时候。另外持续整合也是一种很好的开发方式，使团队成员能产生细微的，交互的变动，能更有效的支持平行开发进程。

maven continuum

可以使用 的 作为持久整合的服务。

安装continuum，比较简，使用以下的命令：

```
C:\mvnbook\continuum-1.0.3> bin\win32\run
```

可以通过<http://localhost:8082/continuum>来验证

为了支持continuum 发送e-mail提醒，你需要相应的smtp服务用于发送信息。默认使用localhost:25，如果你没有设置，编辑上面的文件改变smtp-host设置。

下一步，设置svn目录：

```
svn co file://localhost/C:/mvnbook/svn/proficio/trunk proficio
```

编辑pom.xml用于正确相应得e-mail地址。

```
...
<ciManagement>
<system>continuum</system>
<url>http://localhost:8080/continuum
<notifiers>
<notifier>
<type>mail</type>
<configuration>
<address>youremail@yourdomain.com</address>
</configuration>
</notifier>
</notifiers>
</ciManagement>
...
<scm>
<connection>
scm:svn:file://localhost/c:/mvnbook/svn/proficio/trunk
</connection>
<developerConnection>
scm:svn:file://localhost/c:/mvnbook/svn/proficio/trunk
</developerConnection>
</scm>
...
<distributionManagement>
<site>
<id>website</id>
<url>
file://localhost/c:/mvnbook/repository/sites/proficio
/reference/${project.version}
</url>
</site>
</distributionManagement>
```

提交相应的pom,然后执行mvn install

如果你返回http://localhost:8082/continuum, 你会看到相应的项目列表。

一旦你登录后, 你可以选择mavan 2.0项目用于增加相应的项目。你可以增加你的url或者提交你的本地内容。

你可以使用本地pom url, 如下file://localhost/c:mvnbook/proficio/pom.xml

在提交了此url后, continuum将会返回相应的成功信息。

以下的原则用于更好的帮助持续整合:

早提交, 经常提交: 当用户经常提交时, 持续整合是最有效的。这并不意味着, 提交不正确的代码。

经常运行build: 用于最快检测失败

尽快修正失败: 当失败发生时, 应该马上修正失败

建议一个有效的版本

运行clean build

运行复杂的综合测试

build所有的项目结构分支

持续运行项目的拷贝

=====

下载e

下载exe可安装的那种。

设maven安装在目录D: "Maven,

下面设置环境变量:

Maven_Home=D: "Maven

在PATH中加入;%Maven_Home%"bin

Maven默认的remote repository是ibiblio.org, 这个地方有些人上不去, 因此要更改一下。Maven找寻配置参数的默认的顺序是从maven.jar中的defaults.properties开始。打开该文件, 修改:

```
maven.repo.remote = http://apache.linuxforum.net/dist/java-repository, http://dist.codehaus.org, http://mirrors.sunsite.dk/maven/, http://public.planetmirror.com/pub/maven, http://www.ibiblio.org/maven
```

Maven默认的本地文件保存在Document and Setting的user.home里面, 非常不便。在%MAVEN_HOME%下建立一个local目录, 将本地的repostiory保存在此, 以免重装系统丢失。

```
maven.home.local = ${maven.home}/local
```

Maven的提示语言在中文环境中可能出现乱码, 进入maven.jar, 发现org"apache"maven"messages目录下的messages_zh_CN.properties里面直接写了中文字符。删掉它! 就不会乱码了。(或者用native2ascii转换)

好了, 可以开工了。执行maven genapp, 采用web模版。id设定为sample。输入maven eclipse, 为该项目增

加eclipse支持。

进入eclipse，导入workspace下面的Sample Project。由于缺少MAVEN_REPO变量，Eclipse提示编译失败。

在Eclipse的Preferences – Java - Build Path - Classpath Variables中添加名为MAVEN_REPO的变量，指向D: "maven"local"repository。

如果想让eclipse支持直接运行maven的goal，可以安装mavenide插件。

xe可安装的那种。

设maven安装在目录D: "Maven，

下面设置环境变量：

Maven_Home=D: "Maven

在PATH中加入;%Maven_Home%"bin

Maven默认的remote repository是ibiblio.org，这个地方有些人上不去，因此要更改一下。Maven找寻配置参数的默认的顺序是从maven.jar中的defaults.properties开始。打开该文件，修改：

```
maven.repo.remote = http://apache.linuxforum.net/dist/java-  
repository, http://dist.codehaus.org,  
http://mirrors.sunsite.dk/maven/, http://public.planetmirror.com/pub/maven,  
http://www.ibiblio.org/maven
```

Maven默认的本地文件保存在Document and Setting的user.home里面，非常不便。在%MAVEN_HOME%下建立一个local目录，将本地的repostiory保存在此，以免重装系统丢失。

```
maven.home.local = ${maven.home}/local
```

Maven的提示语言在中文环境中可能出现乱码，进入maven.jar，发现org"apache"maven"messages目录下的messages_zh_CN.properties里面直接写了中文字符。删掉它！就不会乱码了。（或者用native2ascii转换）

好了，可以开工了。执行maven genapp，采用web模版。id设定为sample。输入maven eclipse，为该项目增加eclipse支持。

进入eclipse，导入workspace下面的Sample Project。由于缺少MAVEN_REPO变量，Eclipse提示编译失败。

在Eclipse的Preferences – Java - Build Path - Classpath Variables中添加名为MAVEN_REPO的变量，指向D: "maven"local"repository。

如果想让eclipse支持直接运行maven的goal，可以安装mavenide插件。

0

0

(请您对文章做出评价)

« 上一篇: [TortoiseSVN使用](#)

» 下一篇: [Maven 让事情变得简单](#)

posted on 2008-01-18 17:31 [kenty](#) 阅读(354) 评论(0) 编辑 收藏 网摘 所属分类: [MAVEN](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#) 。

IT新闻:

- 这一年会怎样? 评点技术商业的5大趋势
- 分布式开源数据库Cassandra 0.5发布
- "无聊经济"时代来了
- EA创始人盯上社交游戏 将为Facebook开发应用
- 1992年的Windows 3.1在Web上复活

每天10分钟，轻松学英语

亚太软件研发团队管理年会



[Custom Eclipse Environ.](#)

Customization and private branding Let our experts show you how

www.poweredbypulse.com

相关搜索: [虚拟主机](#)

网站导航:

[博客园首页](#) [IT新闻](#) [个人主页](#) [闪存](#) [程序员招聘](#) [社区](#) [博问](#) [网摘](#)



[China-pub](#) 计算机图书网上专卖店! 6.5万品种2-8折!

[China-Pub](#) 计算机绝版图书按需印刷服务

相关搜索:

MAVEN

在知识库中查看:

[maven](#)

Powered by: [博客园](#) 模板提供: [沪江博客](#) Copyright ©2010 kenty