**学习maven的使用，看到一篇很实用的入门教程(菜鸟级入门)**

2007-08-28 14:01:04

标签：[maven](http://blog.51cto.com/tag-maven.html) [职场](http://blog.51cto.com/tag-%E8%81%8C%E5%9C%BA.html) [休闲](http://blog.51cto.com/tag-%E4%BC%91%E9%97%B2.html)

一、前言  
        早就知道maven 在java 项目的管理方面名声显赫，于是就想着学习掌握之，于是查阅了大量文档。发现这些文档的作者都是java 的大腕，大多都是站在掌握了一定maven 基础的角度上进行介绍，让我这初学者看的云里雾里不知所云。于是又去查看maven 的官方网站，总算是有所了解，但一旦动手实际操作却又雾里看花。唉，没办法，就只有一遍一遍的动手尝试，经过种  
种磨难总算是有一点眉目了。现在就将我的经验写出来，一来避免将来遗忘，二来也给和我一样的菜鸟们提供一点点的方便。呵呵。本文的主题在于实用操作，个中原理还是麻烦大家去查一下java大腕们的文章吧。这里推荐两篇文章：

Maven 2.0:编译、测试、部署、运行

Maven2 的新特性  
[[url]http://www-128.ibm.com/developerworks/cn/opensource/os-maven2/index.html[/url]](http://www-128.ibm.com/developerworks/cn/opensource/os-maven2/index.html)

二、maven2安装  
1、首先去官方网站下载之：[[url]http://maven.apache.org/download.html[/url]](http://maven.apache.org/download.html)，我选择的是截至目前最新版本maven2.0.4版本  
2、设置环境变量：Maven2下载完后，我将其解压到我的计算机的d:\maven204下，目录结构如下：  
D:\  
|--Maven204  
|-- bin  
|-- conf  
|-- core  
|-- lib  
|-- local

环境变量：（操作系统windows2003）  
        我的电脑----属性----高级----环境变量，点击“系统变量”下的新建，输入：变量名 MAVEN\_HOME; 变量值d:\Maven204，在系统变量列表中找到系统变量path，打开之，在变量值中追加”;%MAVEN\_HOME%\bin”，至此环境变量设置完毕。  
        检查一下是否已经完成安装，打开dos窗口，输入mvn –v，如果出现以下信息则表示maven2已经安装成功：  
X：〉mvn –v  
Maven Version 2.0.4

我已经说得够详细了，还不行？那我只能说你比我还菜。

三、用maven2创建第一个java工程  
1、选择一个根目录，我的java工作目录是D:\eclipse\workspace  
2、打开dos窗口采用cd命令进入D:\eclipse\workspace目录下  
3、输入下列指令  
D:\eclipse\workspace>mvn archetype:create -DgroupId=com.efn -DartifactId=study  
指令执行完毕会出现如下的目录结构：  
study  
|-- pom.xml  
`-- src  
|-- main  
| `-- java  
| `-- com  
| `-- mycompany  
| `-- app  
| `-- App.java  
`-- test  
`-- java  
`-- com  
`-- mycompany  
`-- app  
`-- AppTest.java

四、生成eclipse工程文件  
我采用的开发工具是eclipse,所以用maven的eclipse参数生成eclipse工程文件。输入如下  
1、 采用dos的cd命令进入study目录  
2、 输入指令：mvn eclipse:eclipse 如下所示：  
D:\eclipse\workspace\study>mvn eclipse:eclipse  
指令执行完毕后，发现多了两个文件：.classpath和.project

五、下载maven的eclipse插件  
1、打开eclipse  
2、打开菜单 help -Software updates-Find and Install  
3、选择第二项，进入下一步  
4、点击按钮“New Remote Site…”  
5、在Name中输入：Maven2，在URL中输入[url]http://m2eclipse.codehaus.org/[/url]  
Ok,现在一切都是自动化的，你所需要做的就是等待并按提示按按钮。

六、导入工程  
刚才已经生成了eclipse的工程文件，现在导入之。菜单File-import ，按“下一步”，选择工程文件导入完毕后，工程管理透视图就会出现study节点了，在工程节点上，按鼠标右键，打开弹出菜单，选择maven2菜单项，弹出子菜单->Enable，点击之打开，在Group Id中输入：study即可。

打开study的pom.xml文件，会有如下类似内容：

xml 代码

1. <project  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 [url]http://maven.apache.org/maven-v4\_0\_0.xsd[/url]">
2. <modelVersion>4.0.0</modelVersion>
3. <groupId>com.efn</groupId>
4. <artifactId>study</artifactId>
5. <packaging>jar</packaging>
6. <version>1.0-SNAPSHOT</version>
7. <name>Maven Quick Start Archetype</name>
8. <url>[url]http://maven.apache.org[/url]</url>
9. </project>

注意：有关这个文件的作用和各节点的作用不是本文的描述重点，可以到maven的官方网站查阅[[url]http://maven.apache.org/[/url]](http://maven.apache.org/)目前为止，所有的准备工作都已经就绪，现在我们就以一个主main函数访问数据库为例介绍在实际开发中的使用。

七、搭建架包  
每一个java工程都不免使用第三方的架包，Maven的好处就是可以统一管理这些架包，并使多个java工程共享这些架包。我们所需要做的就是按规则编辑pom.xml文件。Maven已经非常智能，使用maven的eclipse插件的向导，只要输入参数就可以将配置文件生成出来。我们的例子是用jdbc方式访问oracle的数据库。所以需要jdbc和oracle的架包的支持。下面我们就来搭建这两个架包到study工程里来。

有两种方式可以采用：  
1、自动向导  
第一种是采用向导方式并让向导直接去maven的服务器上下载你需要的架包，这种方式存在一定的风险，一来可能maven的服务器上并没有你需要的东东，二来每次智能的maven总是去寻找那并不存在的东东。抛出一大堆红字提示不说，达不到预期目标才是让人郁闷。不过为了保证文档的完整性，还是给出操作步骤。以junit为例（这个东东倒是没有问题，呵呵）当工程的maven被Enable后，弹出菜单的maven2子菜，选择子菜单的“Add Dependency”菜单项，在Query中输入“junit”，向导会自动列出相关列表供选择。选择你最需要的架包，按“OK” 按钮。  
如果你的本地仓库已经存在该架包，则向导只在pom.xml加入依赖项信息，pom.xml文件的内容如下

xml 代码

1. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://[url]www.w3.org/2001/XMLSchema-instance[/url]"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 [[url]http://maven.[/url]](http://maven.apache.org/maven-v4_0_0.xsd)"apache.org/maven-v4\_0\_0.xsd>
2. <modelVersion>4.0.0</modelVersion>
3. <groupId>com.efn</groupId>
4. <artifactId>study</artifactId>
5. <packaging>jar</packaging>
6. <version>1.0-SNAPSHOT</version>
7. <name>Maven Quick Start Archetype</name>
8. <url>[url]http://maven.apache.org[/url]</url>
9. <dependencies>
10. <dependency>
11. <groupId>junit</groupId>
12. <artifactId>junit</artifactId>
13. <version>3.8.1</version>
14. </dependency>
15. </dependencies>
16. </project>

如果本地仓库没有，则向导会去maven的服务器下载架包，并在控制台输出下载日志如下：  
06-8-4 上午09时21分07秒: Local repository folder "" does not exist  
06-8-4 上午09时21分11秒: Reading /study/pom.xml  
06-8-4 上午09时21分11秒: Local repository folder "" does not exist  
06-8-4 上午09时21分18秒: Reading /study/pom.xml  
06-8-4 上午09时21分18秒: Local repository folder "" does not exist  
06-8-4 上午09时21分19秒: Reading /study/pom.xml  
06-8-4 上午09时21分19秒: Local repository folder "" does not exist  
06-8-4 上午09时36分33秒: Local repository folder "" does not exist  
06-8-4 上午09时37分11秒: Reading / study /pom.xml  
06-8-4 上午09时37分11秒: Local repository folder "" does not exist  
06-8-4 上午09时37分15秒: Local repository folder "" does not exist  
06-8-4 上午09时40分07秒: Local repository folder "" does not exist  
06-8-4 上午09时40分08秒: Reading / study /pom.xml  
06-8-4 上午09时40分08秒: Local repository folder "" does not exist  
06-8-4 上午09时46分24秒: Reading / study /pom.xml  
06-8-4 上午09时46分24秒: Local repository folder "" does not exist  
06-8-4 上午09时46分28秒: Local repository folder "" does not exist  
06-8-4 上午09时46分40秒: Local repository folder "" does not exist  
06-8-4 上午09时46分47秒: Local repository folder "" does not exist  
06-8-4 上午09时46分47秒: Reading / study /pom.xml  
06-8-4 上午09时46分47秒: Local repository folder "" does not exist  
06-8-4 上午09 时46 分49 秒: Downloading [central] -> [[url]http://repo1.maven.org/maven2/junit/junit/3.8.1/junit-3.8.1.pom[/url]](http://repo1.maven.org/maven2/junit/junit/3.8.1/junit-3.8.1.pom)  
06-8-4 上午09 时46 分49 秒: Downloaded [central] -> [[url]http://repo1.maven.org/maven2/junit/junit/3.8.1/junit-3.8.1.pom[/url]](http://repo1.maven.org/maven2/junit/junit/3.8.1/junit-3.8.1.pom)  
06-8-4 上午09 时46 分51 秒: Downloading [central] -> [[url]http://repo1.maven.org/maven2/junit/junit/3.8.1/junit-3.8.1.jar[/url]](http://repo1.maven.org/maven2/junit/junit/3.8.1/junit-3.8.1.jar)  
06-8-4 上午09时47分00秒: Downloaded [central] -> [[url]http://repo1.maven.org/maven2/junit/junit/3.8.1/junit-3.8.1.jar[/url]](http://repo1.maven.org/maven2/junit/junit/3.8.1/junit-3.8.1.jar)  
----------------------------------------------------------------  
补充修正：如果出现错误提示同时你手头也有架包的话可以采用maven的指令进行本地化安装。比如我在安装hibernate的架包时告诉我jta无法下载。要求本地化安装，给出的提示如下：  
1) javax.transaction:jta:jar:1.0.1B  
Try downloading the file manually from:[url]http://java.sun.com/products/jta.Then[/url], install it using the command:  
mvn install:install-file -DgroupId=javax.transaction -DartifactId=jta \  -Dversion=1.0.1B -Dpackaging=jar -Dfile=/path/to/filePath to dependency:  
1) com.efn:mywebapp:war:1.0-SNAPSHOT  
2) org.hibernate:hibernate:jar:3.1rc2  
3) javax.transaction:jta:jar:1.0.1B  
----------  
1 required artifact is missing.for artifact: com.efn:mywebapp-1.0-SNAPSHOT.war这个提示是说可以先到sun的网站下载jta架包，然后采用命令行的方式按要求安装，因为我本来就有此架包，于是拷到一个方便的位置，比如c:\,然后在命令行下输入：  
mvn install:install-file -DgroupId=javax.transaction -DartifactId=jta -Dversion=1.0.1B -Dpackaging=jar -Dfile=C:/jta.jar  
执行完毕后，一切OK!  
--------------------------------------------------------------

2、手工配置  
        手工配置比起自动化来说是麻烦了些，不过任何东东掌握在自己手里总归是吃饱喝足谁也不服了不是J。而且配置一次就可以受益终身。更重要的是能解决自动化完成不了的任务。比如我现在要配置oracle的jdbc驱动，采用自动化就提示我下载不了让我改变下载地址等等，真累。  
        算了还是自己来吧。  
        手工配置前还是先介绍一下相关背景知识。首先了解一下jar仓库的概念，maven采用集中化的方式管理架包，凡用maven构建的java工程都可以重用统一管理下的架包，所以必须有一个统一存放jar文件的地方，这就是jar仓库，不论是开发还是部署都会建立一个本地仓库。这个仓库的位置默认是X:\Documents and Settings\Administrator.LIPENG\.m2\repository（X是你操作系统所在盘符）下，你可以修改配置文件改变默认的输出位置。该配置文件在maven目录下的conf子目录下。文件名叫settings.xml。在配置文件中加入如下节点信息即可D:/Repository。不过不建议改变，好像eclipse的maven插件只认默认的输出位置。我就是在这方面出了问题搞了半天才发现是输出路径的问题，也许是我还  
没有玩转插件。谁要是解决此问题别忘了跟大家共享。  
现在我们分析一下配置文件pom.xml的结构，以junit为例，格式如下：

xml 代码

1. <dependencies>
2. <dependency>
3. <groupId>junit</groupId>
4. <artifactId>junit</artifactId>
5. <version>3.8.1</version>
6. </dependency>
7. </dependencies>

在来看看本地仓库目录结构  
Repository  
`-- junit  
|-- junit  
| `-- 3.8.1  
| `-- junit-3.8.1.jar  
现在大家应该明白了吧，多余的话不说啦。照葫芦画瓢就是。不过注意先建目录后写配置文件，否则一旦保存，智能的插件就马上开始下载了…

现在开始手动建立oracle的jdbc目录并配置文件,首先建立目录结构如下：  
Repository  
`-- ojdbc  
|-- ojdbc  
| `-- 14  
| `-- ojdbc-14.jar  
如果你手头的jar文件名叫ojdbc14.jar，则改为ojdbc-14.jar,写配置文件:

xml 代码

1. <dependency>
2. <groupId>ojdbc</groupId>
3. <artifactId>ojdbc</artifactId>
4. <version>14</version>
5. </dependency>

那么现在一个完整的pom.xml文件如下：

xml 代码

1. <?xml version="1.0"?>
2. <project>
3. <modelVersion>4.0.0</modelVersion>
4. <groupId>com.mycompany.app</groupId>
5. <artifactId>myapp</artifactId>
6. <name>Maven Quick Start Archetype</name>
7. <version>1.0-SNAPSHOT</version>
8. <url>[url]http://maven.apache.org[/url]</url>
9. <dependencies>
10. <dependency>
11. <groupId>ojdbc</groupId>
12. <artifactId>ojdbc</artifactId>
13. <version>14</version>
14. </dependency>
15. <dependency>
16. <groupId>junit</groupId>
17. <artifactId>junit</artifactId>
18. <version>3.8.1</version>
19. </dependency>
20. </dependencies>
21. </project>

保存之，则发现工程管理透视图发生了一点变化,依此方法再加上jdbc的架包，现在可以开始写程序了，建一个类并添加main函数，编写程序如下：

java 代码

1. public static void main( String[] args )
2. {
3. Connection conn = null;
4. PreparedStatement ps = null;
5. ResultSet rs = null;
6. try {
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8. conn = DriverManager.getConnection("jdbc:oracle:thin:@(description=(address\_list=(address=(protocol=TCP)(port=1521)(host=192.168.0.240)))(connect\_data=(SERVER = DEDICATED)(SERVICE\_NAME = db.efriendnet.com)))", "efnx", "efnx");
9. ps = conn.prepareStatement("select \* From tb\_partyinfo");
10. rs = ps.executeQuery();
11. while(rs.next())
12. {
13. System.out.println(rs.getString("topic"));
14. }
15. } catch (Exception e) {
16. System.out.print(e.getMessage());
17. }
18. finally
19. {
20. if (rs != null) {try {rs.close();} catch (SQLException e) {}}
21. if (ps != null) {try {ps.close();} catch (SQLException e) {}}
22. if (conn != null) {try {conn.close();} catch (SQLException e) {}}
23. }
24. }

别忘了import相应的包

八、编译程序  
采用maven构建系统，则编译过程就独立了出来。这时你再用eclipse自带的编译工具就不起作用了。所以要想编译、调试、运行还要做一些工作。以前是在dos命令行方式下进行编译，现在的插件很好用，在eclipse配置一下就可以编译了。很方便。现在就做一个介绍。

Eclipse有一个扩展工具就是用来集成其他构建工具的在工程的节点上点击鼠标右键，选择属性，在“编译”的右边窗口选择”新建”按钮，在对话框的“name”中输入：study,点击“Browse Workspace…”列出工程列表供选择。  
选择完毕后，在goals中输入package。别忘了apply.好了，让我们Run吧。如果一切正常，  
控制台会打出maven的编译信息如下：  
[INFO] ---------------------------------------------------------------------  
[INFO] Building Maven Quick Start Archetype  
[INFO] task-segment: [package]  
[INFO] ---------------------------------------------------------------------  
[INFO] resources:resources  
[INFO] Using default encoding to copy filtered resources.  
[INFO] compiler:compile  
[INFO] Nothing to compile - all classes are up to date  
[INFO] resources:testResources  
[INFO] Using default encoding to copy filtered resources.  
[INFO] compiler:testCompile  
[INFO] Nothing to compile - all classes are up to date  
[INFO] surefire:test  
[INFO] Surefire report directory: D:\eclipse\workspace\study\target\s  
urefire-reports  
-------------------------------------------------------  
T E S T S  
-------------------------------------------------------  
Running com.efn.AppTest  
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.157 sec  
Results :  
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
[INFO] jar:jar  
[INFO] Building jar: D:\eclipse\workspace\study\target\study-1.0-SNAPSHOT.jar  
[INFO] ----------------------------------------------------------------------------  
[INFO] BUILD SUCCESSFUL  
[INFO] ---------------------------------------------------------------------  
[INFO] Total time: 4 second  
[INFO] Finished at: Fri Aug 04 10:55:42 CST 2006  
[INFO] Memory 2M/7M  
[INFO] --------------------------------------------------------------------  
注意，别忘了每一次程序改动完毕后都要经过这一步编译。因为这是maven的编译器！

九、调试程序  
经过以上步骤我们已经完成了mave管理下的软件生命周期，但是作为一个程序开发人员我们还要进行调试。这里的调试设置和普通的java程序的调试是一样的。  
首先，打开debug对话框：  
        因为是一个一般的java应用程序，所以我们选择 Java Application,点击“New”按钮，输入一些相应的参数，apply-Debug  
Ok,一切正常！希望你也顺利！

十、结束语  
本文只是简单的对maven的操作步骤做一个指南性的说明，实际应用还有很多东西需要实践。如果发现哪里有错误和纰漏也欢迎批评指正！