Maven学习笔记

Window下maven环境安装和配置

步骤一：

配置好java环境

JAVA\_HOME C:\Program Files\Java\jdk1.8.0\_101

Path %JAVA\_HOME%\bin

配置Maven环境



解压文件

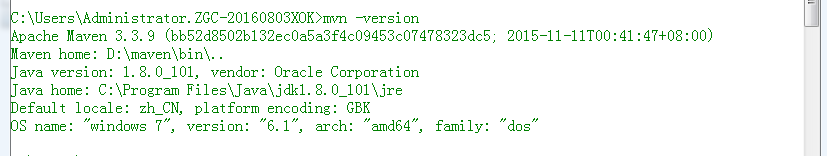


配置路径

M\_HOME D:\maven

Path %M\_HOME%\bin

打开cmd测试环境是否配置成功

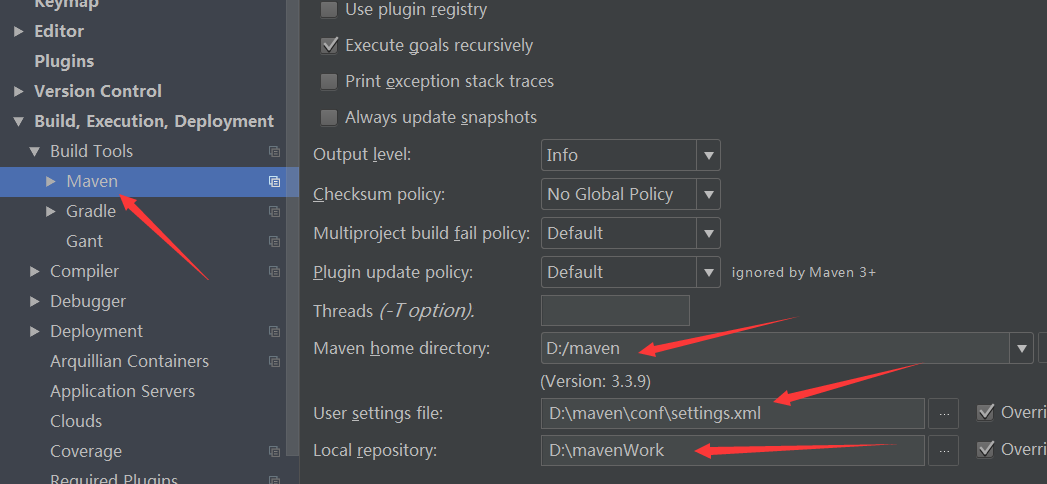


如果想要修改Maven的本地仓库位置，则可以直接在Maven的安装目录下找到conf文件下的setting配置文件中，设置localRepository为本地仓库位置

　　　　<localRepository>E:\java\repo</localRepository>

配置idea中Maven插件

当配置完Maven之后，我们需要给Idea配置Maven，那么首先必须先要安装Idea，Idea的安装在这里就不累赘了，请自行百度，非常简单，下一步下一步即可，安装完成之后打开Idea设置Maven

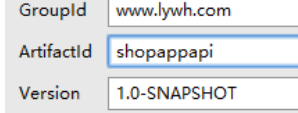


创建Maven管理的Java Web项目，

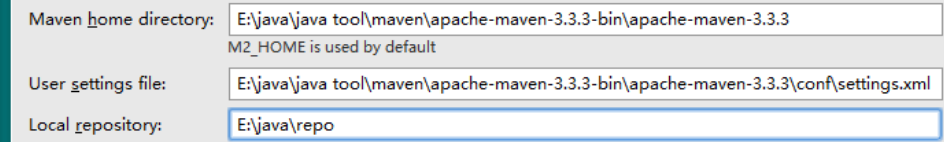
单击File->New Project->选择Maven

选中Createfrom archetype，选择maven-archetype-webapp

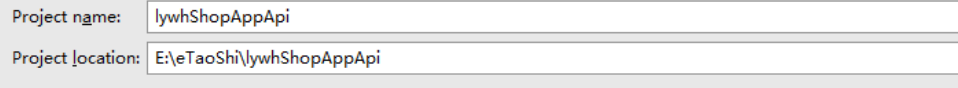
单击Next填写GroupId和ArtifactId和Version



单击Next，此页面获取的是maven的安装信息



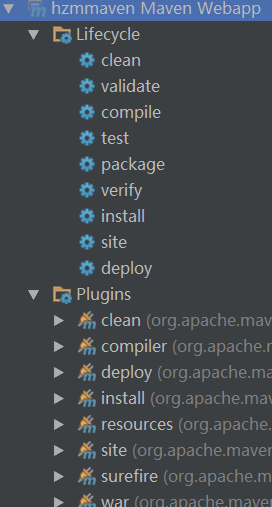
单击Next，填写项目名称和项目存放的路经



单击 Finish完成，即创建Maven项目成功

如果单击完成在下面的提示中报错，出现问题的可能性是Maven和Idea的兼容性问题，建议将Maven换成低版本的即可

通过以上步骤即安装了Maven和开发了一个Maven的JavaWeb项目，那么接下来就需要将JavaWeb打包(war文件)发布到Tomcat下，

.在Idea中的最右边的导航栏中可以看到一个Maven Projects，单击打开，如图所示：

接下来我们就能够利用这个简单的工具对Maven进行打包(war)

单击Run Maven Build

当单击Run Maven Build的时候，出现错误，如图所示



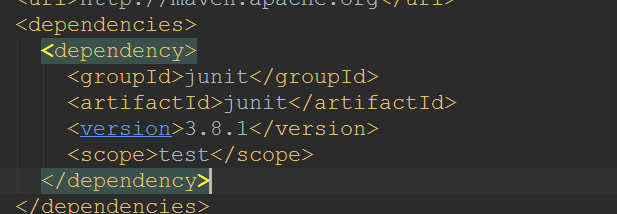
出现以上错误之后，按照下面的步骤解决，单击File->Setting->在搜索框中输入Maven回车直接定位到Maven节点->Runner,打开之后将这段内容（-Dmaven.multiModuleProjectDirectory=$M2\_HOME）复制到VM Options的文本框中，单击OK即可。

选择install右键选择Run运行，运行完成之后则会提示你打包的war包在哪里

然后找到war包,复制到Tomcat的WebApps文件夹下面，然后直接访问网站即可访问

重要的点：

Maven项目的核心是pom.xml ,pom (项目对象模型) 定义了项目的基本信息， 用于描述项目如何构建，声明项目依赖



GroupId定义了项目是哪个组 ，这个组往往和项目所在的组织或公司存在关联

Artifactid 定义了当前maven 项目在组中的唯一的id

Version 指定了hello World 项目当前的版本

通过定义一个Maven项目的pom他能让项目对象模型最大程度地与实际代码相独立

也可以被称之为解耦，避免了java代码和pom代码的相互影响

问题：无法添加oralce的支持包：

解决方式手动添加：

<dependency>  
 <groupId>com.oracle</groupId>  
 <artifactId>ojdbc6\_g</artifactId>  
 <version>11.2.0.4.0</version>  
</dependency>

mvn install:install-file -DgroupId=com.oracle -DartifactId=ojdbc6\_g -Dversion=11.2.0.4.0 -Dpackaging=jar -Dfile=ojdbc6\_g.jar

mvn install:install-file -DgroupId=com.oracle -DartifactId=ojdbc6 -Dversion=11.2.0.4.0 -Dpackaging=jar -Dfile=ojdbc6.jar

Maven常用命令：

1. 创建Maven的普通java项目：

mvn archetype:create

-DgroupId=packageName

-DartifactId=projectName

2. 创建Maven的Web项目：

mvn archetype:create

-DgroupId=packageName

-DartifactId=webappName

-DarchetypeArtifactId=maven-archetype-webapp

3. 编译源代码： mvn compile

4. 编译测试代码：mvn test-compile

5. 运行测试：mvn test

6. 产生site：mvn site

7. 打包：mvn package

8. 在本地Repository中安装jar：mvn install

9. 清除产生的项目：mvn clean

10. 生成eclipse项目：mvn eclipse:eclipse

11. 生成idea项目：mvn idea:idea

12. 组合使用goal命令，如只打包不测试：mvn -Dtest package

13. 编译测试的内容：mvn test-compile

14. 只打jar包: mvn jar:jar

15. 只测试而不编译，也不测试编译：mvn test -skipping compile -skipping test-compile

( -skipping 的灵活运用，当然也可以用于其他组合命令)

16. 清除eclipse的一些系统设置:mvn eclipse:clean

ps：

一般使用情况是这样，首先通过cvs或svn下载代码到本机，然后执行mvn eclipse:eclipse生成ecllipse项目文件，然后导入到eclipse就行了；修改代码后执行mvn compile或mvn test检验，也可以下载eclipse的maven插件。

mvn -version/-v 显示版本信息

mvn archetype:generate 创建mvn项目

mvn archetype:create -DgroupId=com.oreilly -DartifactId=my-app 创建mvn项目

mvn package 生成target目录，编译、测试代码，生成测试报告，生成jar/war文件

mvn jetty:run 运行项目于jetty上,

mvn compile 编译

mvn test 编译并测试

mvn clean 清空生成的文件

mvn site 生成项目相关信息的网站

mvn -Dwtpversion=1.0 eclipse:eclipse 生成Wtp插件的Web项目

mvn -Dwtpversion=1.0 eclipse:clean 清除Eclipse项目的配置信息(Web项目)

mvn eclipse:eclipse 将项目转化为Eclipse项目

在应用程序用使用多个存储库

<repositories>

<repository>

<id>Ibiblio</id>

<name>Ibiblio</name>

<url>http://www.ibiblio.org/maven/</url>

</repository>

<repository>

<id>PlanetMirror</id>

<name>Planet Mirror</name>

<url>http://public.planetmirror.com/pub/maven/</url>

</repository>

</repositories>

mvn deploy:deploy-file -DgroupId=com -DartifactId=client -Dversion=0.1.0 -Dpackaging=jar -Dfile=d:\client-0.1.0.jar -DrepositoryId=maven-repository-inner -Durl=ftp://xxxxxxx/opt/maven/repository/

发布第三方Jar到本地库中：

mvn install:install-file -DgroupId=com -DartifactId=client -Dversion=0.1.0 -Dpackaging=jar -Dfile=d:\client-0.1.0.jar

-DdownloadSources=true

-DdownloadJavadocs=true

mvn -e 显示详细错误 信息.

mvn validate 验证工程是否正确，所有需要的资源是否可用。

mvn test-compile 编译项目测试代码。 。

mvn integration-test 在集成测试可以运行的环境中处理和发布包。

mvn verify 运行任何检查，验证包是否有效且达到质量标准。

mvn generate-sources 产生应用需要的任何额外的源代码，如xdoclet。

常用命令：

mvn -v 显示版本

mvn help:describe -Dplugin=help 使用 help 插件的 describe 目标来输出 Maven Help 插件的信息。

mvn help:describe -Dplugin=help -Dfull 使用Help 插件输出完整的带有参数的目标列

mvn help:describe -Dplugin=compiler -Dmojo=compile -Dfull 获取单个目标的信息,设置 mojo 参数和 plugin 参数。此命令列出了Compiler 插件的compile 目标的所有信息

mvn help:describe -Dplugin=exec -Dfull 列出所有 Maven Exec 插件可用的目标

mvn help:effective-pom 看这个“有效的 (effective)”POM，它暴露了 Maven的默认设置

mvn archetype:create -DgroupId=org.sonatype.mavenbook.ch03 -DartifactId=simple -DpackageName=org.sonatype.mavenbook 创建Maven的普通java项目，在命令行使用Maven Archetype 插件

mvn exec:java -Dexec.mainClass=org.sonatype.mavenbook.weather.Main Exec 插件让我们能够在不往 classpath 载入适当的依赖的情况下，运行这个程序

mvn dependency:resolve 打印出已解决依赖的列表

mvn dependency:tree 打印整个依赖树

mvn install -X 想要查看完整的依赖踪迹，包含那些因为冲突或者其它原因而被拒绝引入的构件，打开 Maven 的调试标记运行

mvn install -Dmaven.test.skip=true 给任何目标添加maven.test.skip 属性就能跳过测试

mvn install assembly:assembly 构建装配Maven Assembly 插件是一个用来创建你应用程序特有分发包的插件

mvn jetty:run 调用 Jetty 插件的 Run 目标在 Jetty Servlet 容器中启动 web 应用

mvn compile 编译你的项目

mvn clean install 删除再编译

mvn hibernate3:hbm2ddl 使用 Hibernate3 插件构造数据库