运行tomcat

|  |
| --- |
| <plugin>  <groupId>org.apache.tomcat.maven</groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <version>2.2</version>  <configuration>  <port>8080</port>  <useTestClasspath>false</useTestClasspath>  <additionalClasspathDirs>  <additionalClasspathDir></additionalClasspathDir>  </additionalClasspathDirs>  </configuration>  </plugin> |

运行tomcat7：mvn -X tomcat7:run

编译源代码**maven-compile-plugin**

    这个插件就如同名字所显示的这样，用来编译源代码的。最开始碰到这个插件是在于有的时候我们下载了一些工程需要编译的时候，比如我们输入命令：mvn install ，但是系统编译的时候报错了，错误的信息如下：

Shell代码  [藏代码](javascript:void())

1. [ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:2.0.2:compile (default-compile) on project springJMS: Compilation failure: Compilation failure:
2. [ERROR] /home/frank/programcode/SpringJMSSample/src/main/java/huangbowen/net/jms/MessageSender.java:[6,1] error: annotations are not supported in -source 1.3
3. [ERROR]
4. [ERROR] (use -source 5 or higher to enable annotations)
5. [ERROR] /home/frank/programcode/SpringJMSSample/src/main/java/net/EmbedBrokerApp.java:[5,7] error: static import declarations are not supported in -source 1.3
6. [ERROR] -> [Help 1]
7. [ERROR]
8. [ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
9. [ERROR] Re-run Maven using the -X switch to enable full debug logging.
10. [ERROR]
11. [ERROR] For more information about the errors and possible solutions, please read the following articles:
12. [ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException

    从错误显示的信息我们就可以看出，这是因为编译的时候是默认用的javac 1.3版本的，太老了不支持代码里的特性。为了修改这个问题，我们需要设置编译器的版本。解决这个问题的办法也比较简单，就是直接在后面的插件部分增加如 下的插件，比如如下部分，将编译器的版本设定为1.6：

Xml代码  [藏代码](javascript:void())

1. <build>
2. <plugins>
3. <plugin>
4. <groupId>org.apache.maven.plugins</groupId>
5. <artifactId>maven-compiler-plugin</artifactId>
6. <version>2.3.2</version>
7. <configuration>
8. <source>1.7</source>
9. <target>1.7</target>
10. </configuration>
11. </plugin>
12. </plugins>
13. </build>

 发布war包

|  |
| --- |
| <profiles>  <profile>  <id>cdc-deploy</id>  <build>  <plugins>  <plugin>  <artifactId>exec-maven-plugin</artifactId>  <groupId>org.codehaus.mojo</groupId>  <version>1.2.1</version>  <executions>  <execution>  <id>deploy-i</id>  <phase>install</phase>  <goals>  <goal>exec</goal>  </goals>  <configuration>  <executable>${basedir}/shell/remote-deploy.sh</executable>  </configuration>  </execution>  </executions>  </plugin>  </plugins>  </build>  </profile>  </profiles> |

**运行jar exec-maven-plugin**

    我们写一些java console相关的程序时，比较头疼的一点就是真正要通过命令行将打包后的程序执行起来还是比较麻烦的。我们需要在命令行里敲如下的命令：java -cp \*\*\*.jar:\*\*.jar:/\*\*/ 。因为要将classpath目录以及引用的类库都加入进来，并指定运行的入口，这样子每次用手工的方式来处理实在是太繁琐也比较容易出错。所以一种办法 就是利用这个插件，通过一些基本的配置，我们可以执行起代码来的时候很方便。一个典型的配置如下：

Xml代码  [藏代码](javascript:void())

1. <plugin>
2. <groupId>org.codehaus.mojo</groupId>
3. <artifactId>exec-maven-plugin</artifactId>
4. <version>1.2.1</version>
5. <executions>
6. <execution>
7. <goals>
8. <goal>java</goal>
9. </goals>
10. </execution>
11. </executions>
12. <configuration>
13. <mainClass>com.yunzero.App</mainClass>
14. </configuration>
15. </plugin>

   如果我们运行的时候需要提供一些输入的参数，也可以通过configuration的元素里添加。这样后续要执行这个程序时，我们只需要在命令行执行如下命令：mvn exec:java ，然后程序就可以运行起来了。

**打包拷贝资源文件1 maven-dependency-plugin**

　　还有一个比较常用的插件就是这个。我们在IDE的环境里编译和执行代码的时候，那是直接引用一些类库。但是在我们实际部署的环境里，那边很可能 就一个java执行环境，不可能有源代码和IDE。这个时候，我们需要将源代码编译打包。这个时候的一个问题就是如果我们引用的库很多的话，我们希望能够 把他们统一打包到一个目录下，比如lib文件夹。这样部署执行的时候只需要将编译生成的程序jar包和依赖包文件夹拷到特定目录去执行。要实现这个效果也 比较容易：

Xml代码  [藏代码](javascript:void())

1. <plugin>
2. <artifactId>maven-dependency-plugin</artifactId>
3. <executions>
4. <execution>
5. <phase>install</phase>
6. <goals>
7. <goal>copy-dependencies</goal>
8. </goals>
9. <configuration>
10. <outputDirectory>${project.build.directory}/lib</outputDirectory>
11. </configuration>
12. </execution>
13. </executions>
14. </plugin>

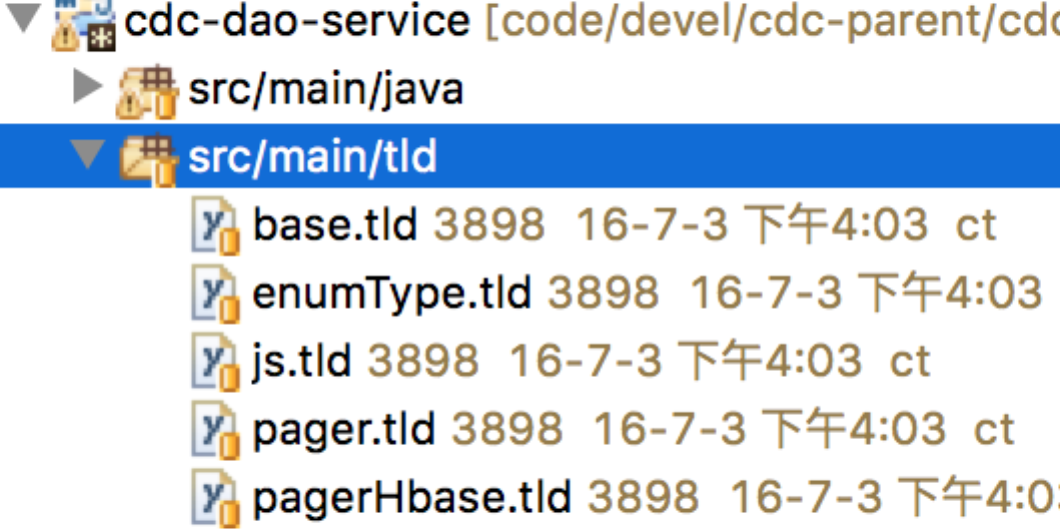
     从前面的配置里我们可以看到，插件的执行被配置到install这个阶段。这样，当我们执行命令：mvn clean install 的时候，会发现对应的target目录里生成了对应的jar包和依赖包。

**打包拷贝资源文件2 resources**

|  |
| --- |
| <build>  <resources>  <resource>  <directory>src/main/resources</directory>  </resource>  </resources>  </build> |

**打包拷贝自定义标签 resources**

|  |
| --- |
| <build>  <resources>  <resource>  <directory>src/main/tld</directory>  <targetPath>META-INF</targetPath>  </resource>  </resources>  </build> |
| <build>  <resources>  <resource>  <directory>src/main/java/com/xjgz/cdc/runtime/phoenix/select</directory>  <includes>  <include>\*.json</include>  </includes>  <targetPath>com/xjgz/cdc/runtime/phoenix/select</targetPath>  </resource>  <resource>  <directory>src/main/java/com/xjgz/cdc/runtime/hadoop/script</directory>  </resource>  <resource>  <directory>src/main/resources</directory>  </resource>  </resources>  <testResources>  <testResource>  <directory>src/test/resources</directory>  <filtering>true</filtering>  </testResource>  </testResources> |

****