https://www.iteye.com/blog/donald-draper-2382829

从github检出netty项目，编译遇到问题总结

**博客分类：**

* [Netty](https://www.iteye.com/category/372679)
* [Maven](https://www.iteye.com/category/361588)

[maven](http://www.iteye.com/blogs/tag/maven)[netty](http://www.iteye.com/blogs/tag/netty)

Maven学习记录3——创建、编译、打包、运行项目 ：<http://blog.csdn.net/yaya1943/article/details/48464371>   
使用maven编译Java项目：[http://www.tuicool.com/articles/YfIfIrq](http://www.tuicool.com/articles/YfIfIrq" \t "_blank)   
netty github 导入 Eclipse：[http://www.th7.cn/Program/java/201502/389732.shtml](http://www.th7.cn/Program/java/201502/389732.shtml" \t "_blank)   
netty源码编译环境搭建 ：<http://blog.csdn.net/wuyinxian/article/details/46382051>   
Plugin error: execution not covered by lifecycle configuration：   
<https://stackoverflow.com/questions/7391201/plugin-error-execution-not-covered-by-lifecycle-configuration>  
Execution Not Covered：[http://www.eclipse.org/m2e/documentation/m2e-execution-not-covered.html](http://www.eclipse.org/m2e/documentation/m2e-execution-not-covered.html" \t "_blank)   
netty项目github地址：[https://github.com/netty/netty.git](https://github.com/netty/netty.git" \t "_blank)   
       由于netty项目在github中，没有.project和.classpath文件，所以不能够直接转换为eclipse工程Maven项目，要手动去编译，具体可参考netty源码编译环境搭建和netty github导入Eclipse这两篇文章。   
我们总结一下从github检出netty项目，编译maven项目，主要遇到的问题：   
**1.Failure to transfer io.netty:netty-tcnative:jar:${tcnative.classifier}:2.0.3.Final**   
这个问题主要是tcnative jar在netty-parent：

**Java代码  [收藏代码](javascript:void())**

1. <groupId>io.netty</groupId>
2. <artifactId>netty-parent</artifactId>
3. <packaging>pom</packaging>
4. <version>4.1.13.Final-SNAPSHOT</version>

的POM中定义了如下片段：

**Java代码  [收藏代码](javascript:void())**

1. <tcnative.artifactId>netty-tcnative</tcnative.artifactId>
2. <tcnative.version>2.0.3.Final</tcnative.version>
3. <tcnative.classifier>${os.detected.classifier}</tcnative.classifier>

根据操作系统探测tcnative的classifier，我们只需要将上面这句话注释掉如下：

**Java代码  [收藏代码](javascript:void())**

1. <!-- <tcnative.classifier>${os.detected.classifier}</tcnative.classifier> -->

同时在相应子模块项目中，tcnative包依赖中将：

**Java代码  [收藏代码](javascript:void())**

1. <dependency>
2. <groupId>${project.groupId}</groupId>
3. <artifactId>${tcnative.artifactId}</artifactId>
4. <classifier>${tcnative.classifier}</classifier>
5. <optional>**true**</optional>
6. </dependency>

改为：

**Java代码  [收藏代码](javascript:void())**

1. <dependency>
2. <groupId>${project.groupId}</groupId>
3. <artifactId>${tcnative.artifactId}</artifactId>
4. <!-- <classifier>${tcnative.classifier}</classifier> -->
5. <version>${tcnative.version}</version>
6. <optional>**true**</optional>
7. </dependency>

这里我们是以netty-handler子模块为例：

**Java代码  [收藏代码](javascript:void())**

1. <parent>
2. <groupId>io.netty</groupId>
3. <artifactId>netty-parent</artifactId>
4. <version>4.1.13.Final-SNAPSHOT</version>
5. </parent>
6. <artifactId>netty-handler</artifactId>
7. <packaging>jar</packaging>

**2.Plugin execution not covered by lifecycle configuration:   
org.codehaus.mojo:build-helper-maven-plugin:1.10:add-source   
(execution: add-source, phase: generate-sources)**   
这个问题主要是执行不能被声明周期配置覆盖导致，我们以netty-common子模块为例:

**Java代码  [收藏代码](javascript:void())**

1. <plugin>
2. <groupId>org.codehaus.mojo</groupId>
3. <artifactId>build-helper-maven-plugin</artifactId>
4. <version>1.10</version>
5. <executions>
6. <execution>
7. <id>add-source</id>
8. <phase>generate-sources</phase>
9. <goals>
10. <goal>add-source</goal>
11. </goals>
12. <configuration>
13. <sources>
14. <source>${collection.src.dir}</source>
15. </sources>
16. </configuration>
17. </execution>
18. </plugin>

groupId为org.codehaus.mojo，artifactId为build-helper-maven-plugin的add-source执行不行被生命周期覆盖，解决方式，在pom文件的build标签下加入pluginManagement对应的片段：

**Java代码  [收藏代码](javascript:void())**

1. <build>
2. <pluginManagement>
3. <plugins>
4. <plugin>
5. <groupId>org.eclipse.m2e</groupId>
6. <artifactId>lifecycle-mapping</artifactId>
7. <version>1.0.0</version>
8. <configuration>
9. <lifecycleMappingMetadata>
10. <pluginExecutions>
11. <!-- plugin执行器过滤器 -->
12. <pluginExecution>
13. <!-- plugin执行器过滤器 -->
14. <pluginExecutionFilter>
15. <!-- 这里的groupId和artifactId和上面的相对应 -->
16. <groupId>org.codehaus.mojo</groupId>
17. <artifactId>build-helper-maven-plugin</artifactId>
18. <!--  版本大于等于1.0.0 -->
19. <versionRange>[1.0.0,)</versionRange>
20. <goals>
21. <!--  目标与上面对应者execution goals中goal -->
22. <goal>add-source</goal>
23. <goal>add-test-source</goal>
24. </goals>
25. </pluginExecutionFilter>
26. <!-- plugin执行器忽略 groupId，artifactId 对应的goals中goal -->
27. <action>
28. <!--忽略执行 -->
29. <ignore/>
30. </action>
31. </pluginExecution>
32. <!-- plugin执行器过滤器 -->
33. <pluginExecution>
34. <!-- plugin执行器过滤器 -->
35. <pluginExecutionFilter>
36. <groupId>org.codehaus.mojo</groupId>
37. <artifactId>xml-maven-plugin</artifactId>
38. <versionRange>[1.0.0,)</versionRange>
39. <goals>
40. <goal>parse-version</goal>
41. <goal>check-style</goal>
42. </goals>
43. </pluginExecutionFilter>
44. <!-- plugin执行器忽略 groupId，artifactId 对应的goals中goal -->
45. <action>
46. <ignore/>
47. </action>
48. </pluginExecution>
49. <!-- plugin执行器过滤器 -->
50. <pluginExecution>
51. <!-- plugin执行器过滤器 -->
52. <pluginExecutionFilter>
53. <groupId>org.codehaus.gmaven</groupId>
54. <artifactId>groovy-maven-plugin</artifactId>
55. <versionRange>[1.0.0,)</versionRange>
56. <goals>
57. <goal>execute</goal>
58. </goals>
59. </pluginExecutionFilter>
60. <!-- plugin执行器忽略 groupId，artifactId 对应的goals中goal -->
61. <action>
62. <ignore/>
63. </action>
64. </pluginExecution>
65. </pluginExecutions>
66. </lifecycleMappingMetadata>
67. </configuration>
68. </plugin>
69. </plugins>
70. </pluginManagement>
71. </build>

还有一种方式为配置Maven -> Lifecycle mapping->lifecycle-mapping-metadata.xml ，   
官方显示这种方法针对Eclipse 4.2，具体配置在Windows -> Preferences -> Maven -> Lifecycle mapping ，文件（eclipse/plugins/org.eclipse.m2e.lifecyclemapping.defaults\_1.2.0.20120903-1050.jar/lifecycle-mapping-metadata.xml）内容如下：

**Java代码  [收藏代码](javascript:void())**

1. <?xml version="1.0" encoding="UTF-8"?>
2. <lifecycleMappingMetadata>
3. <pluginExecutions>
4. <pluginExecution>
5. <pluginExecutionFilter>
6. <groupId>org.codehaus.mojo</groupId>
7. <artifactId>buildnumber-maven-plugin</artifactId>
8. <goals>
9. <goal>create-timestamp</goal>
10. </goals>
11. <versionRange>[0.0,)</versionRange>
12. </pluginExecutionFilter>
13. <action>
14. <ignore />
15. </action>
16. </pluginExecution>
18. <pluginExecution>
19. <pluginExecutionFilter>
20. <groupId>org.apache.maven.plugins</groupId>
21. <artifactId>maven-dependency-plugin</artifactId>
22. <goals>
23. <goal>list</goal>
24. </goals>
25. <versionRange>[0.0,)</versionRange>
26. </pluginExecutionFilter>
27. <action>
28. <ignore />
29. </action>
30. </pluginExecution>
32. <pluginExecution>
33. <pluginExecutionFilter>
34. <groupId>org.zeroturnaround</groupId>
35. <artifactId>jrebel-maven-plugin</artifactId>
36. <goals>
37. <goal>generate</goal>
38. </goals>
39. <versionRange>[0.0,)</versionRange>
40. </pluginExecutionFilter>
41. <action>
42. <ignore />
43. </action>
44. </pluginExecution>
46. <pluginExecution>
47. <pluginExecutionFilter>
48. <groupId>org.codehaus.mojo</groupId>
49. <artifactId>gwt-maven-plugin</artifactId>
50. <goals>
51. <goal>compile</goal>
52. </goals>
53. <versionRange>[0.0,)</versionRange>
54. </pluginExecutionFilter>
55. <action>
56. <ignore />
57. </action>
58. </pluginExecution>
60. <pluginExecution>
61. <pluginExecutionFilter>
62. <groupId>org.apache.maven.plugins</groupId>
63. <artifactId>maven-dependency-plugin</artifactId>
64. <goals>
65. <goal>copy-dependencies</goal>
66. <goal>unpack</goal>
67. </goals>
68. <versionRange>[0.0,)</versionRange>
69. </pluginExecutionFilter>
70. <action>
71. <ignore />
72. </action>
73. </pluginExecution>
74. </pluginExecutions>
75. </lifecycleMappingMetadata>

注意：pluginExecutionFilter中的groupId，artifactId，goal的对应关系。   
  
    在配置完后记得要重新加载。如果你有多个Eclipse工作空间或者为一个团队项目，强烈建立使用配置POM文件的方式，，即第一种方式。   
  
然后更新项目模块（maven-》update勾选force online update）。   
  
如果还有问题，看看是不是JRE的原因(JRE6-8)，一般为JRE7，   
  
如果项目没有maven依赖包（Maven Dependencies），查看项目的.class   
和.project文件   
**.class文件是否包含如下信息：**

**Java代码  [收藏代码](javascript:void())**

1. <classpathentry kind="con" path="org.eclipse.jdt.launching.JRE\_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.StandardVMType/jdk1.7.0\_17">
2. <attributes>
3. <attribute name="maven.pomderived" value="true"/>
4. </attributes>
5. </classpathentry>
6. <classpathentry kind="con" path="org.eclipse.m2e.MAVEN2\_CLASSPATH\_CONTAINER">
7. <attributes>
8. <attribute name="maven.pomderived" value="true"/>
9. </attributes>
10. </classpathentry>

不包括，则添加上，不同，则修改。   
  
**.project文件是否为如下：**

**Java代码  [收藏代码](javascript:void())**

1. <?xml version="1.0" encoding="UTF-8"?>
2. <projectDescription>
3. <name>netty\_trunk</name>
4. <comment></comment>
5. <projects>
6. </projects>
7. <buildSpec>
8. <buildCommand>
9. <name>org.eclipse.jdt.core.javabuilder</name>
10. <arguments>
11. </arguments>
12. </buildCommand>
13. <buildCommand>
14. <name>org.eclipse.m2e.core.maven2Builder</name>
15. <arguments>
16. </arguments>
17. </buildCommand>
18. </buildSpec>
19. <natures>
20. <nature>org.eclipse.m2e.core.maven2Nature</nature>
21. <nature>org.eclipse.jdt.core.javanature</nature>
22. </natures>
23. </projectDescription

>   
对于没有含有项，则添加上，不同，则修改。   
  
最后，记得刷新工作空间项目。