要写自己的maven plugin的话，lifecycle与phase与goal与mojo的概念是一定要理解的，下面是我自己的一些见解

**lifecycle：**生命周期，这是 maven最高级别的的控制单元，它是一系列的phase组成，也就是说，一个生命周期，就是一个大任务的总称，不管它里面分成多少个子任务，反正就是运 行一个lifecycle，就是交待了一个任务，运行完后，就得到了一个结果，中间的过程，是phase完成的，自己可以定义自己的lifecycle， 包含自己想要的phase

常见的lifecycle有 | clean | package ear | pageage jar | package war | site等等

**phase：**可以理解为任务单 元，lifecycle是总任务，phase就是总任务分出来的一个个子任务，但是这些子任务是被规格化的，它可以同时被多个lifecycle所包含， 一个lifecycle可以包含任意个phase，phase的执行是按顺序的，一个phase可以绑定很多个goal，至少为一个，没有goal的 phase是没有意义的

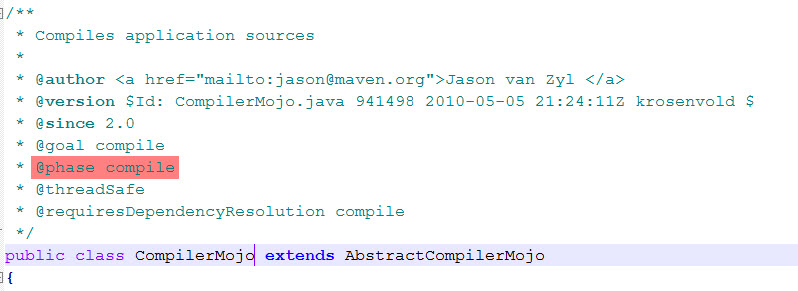
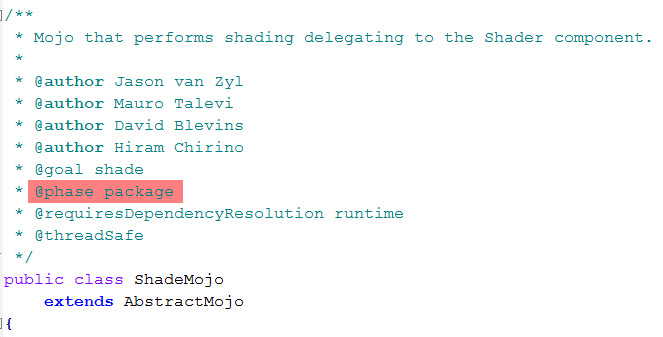
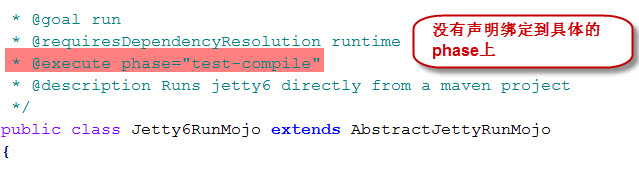
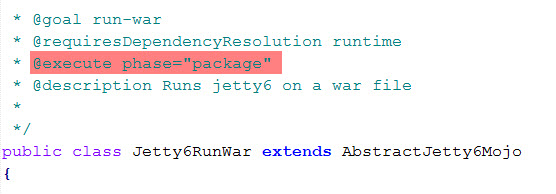
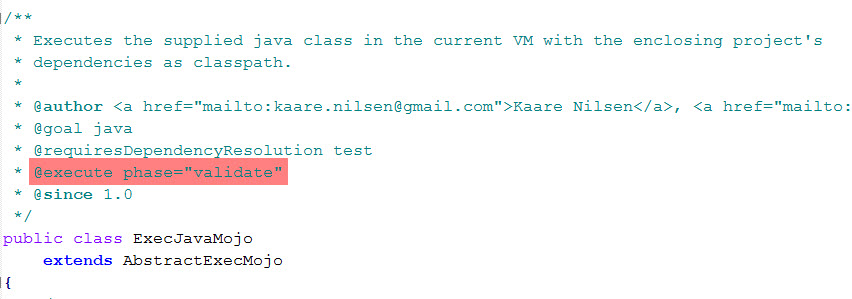
下面就是一些default lifecycle的phase

validate   
initialize   
generate-sources   
process-sources   
generate-resources   
process-resources  
compile compile   
process-classes   
generate-test-sources   
process-test-sources   
generate-test-resources   
process-test-resources   
test-compile   
process-test-classes   
test   
prepare-package  
package   
pre-integration-test   
integration-test   
post-integration-test   
verify   
install   
deploy

**goal:** 这是执行任务的最小单元，它可以 绑定到任意个phase中，一个phase有一个或多个goal，goal也是按顺序执行的，一个phase被执行时，绑定到phase里的goal会按 绑定的时间被顺序执行，不管phase己经绑定了多少个goal，你自己定义的goal都可以继续绑到phase中

**mojo:** lifecycle与phase 与goal都是概念上的东西，mojo才是做具体事情的，可以简单理解mojo为goal的实现类，它继承于AbstractMojo，有一个 execute方法，goal等的定义都是通过在mojo里定义一些注释的anotation来实现的，maven会在打包时，自动根据这些 anotation生成一些xml文件，放在plugin的jar包里

      抛开mojo不讲，lifecycle与phase与goal就是级别的大小问题，引用必须是从高级引用下级（goal绑定到phase，也可理间为 phase引用goal，只是在具体绑定时，不会phase定义引用哪些goal，但是执行是，却是phase调用绑定到它那的goal），也不能跨级引 用，如lifecycle可以引用任意的phase，不同lifecycle可以同时引用相同的phase，lifecycle不能跨级引用goal。 goal会绑定到任意的phase中，也就是说不同的phase可以同时引用相同的goal，所以goal可以在一个lifecycle里被重复执行 哦，goal自然也不能说绑定到lifecycle中，它们三者的关系可以用公司里的 总领导，组领导，与职员的关系来解释

这其实并不是一个复杂的问题，但是似乎没有什么资料对此做过清晰的说明，本文将对这个问题做一个详细的解释。  
 **背景知识**  
  
maven 对构建(build)的过程进行了抽象和定义，这个过程被称为构建的生命周期(lifecycle)。生命周期(lifecycle)由多个阶段 (phase)组成,每个阶段(phase)会挂接一到多个goal。goal是maven里定义任务的最小单元，相当于ant里的target。  
  
**以phase为目标构建**  
  
以phase为目标进行构建是最常见 的，如我们平时经常执行的mvn compile,mvn test,mvn package...等等,compile,test,package都是maven生命周期(lifecycle)里的phase,通过mvn命令，你 可以指定一次构建执行到那一个阶段，在执行过程中，所有经历的执行阶段(phase)上绑定的goal都将得到执行。例如，对于一个jar包应用，当执行 mvn package命令时，maven从validate阶段一个阶段一个阶段的执行，在执行到compile阶段时，compiler插件的compile goal会被执行，因为这个goal是绑定在compile阶段(phase)上的。这一点可从其对应的mojo类上得知：  
  
再比如经常使用的打包插件shade，它的goal是绑定到package阶段的，这样，使用mvn package进行打包时都会执行shade的。  
  
  
**以goal为目标构建**  
  
虽 然以phase为目标的构建最常见，但是有时候我们会发现，**一些插件的goal并不 适合绑定到任何阶段(phase)上，或者是，这些goal往往是单独执行，不需要同某个阶段(phase)绑定在一起**，比如 hibernate插件的导入\导出goal多数情况下是根据需要要手动执行的(当然，也可以绑定到某个阶段上，比如进行单元测试时，可考虑将其绑定到 test阶段上)。再比如jetty(6.1.26)插件，它的goal都是将打包或未打包的工程部署到jetty里然后启动jetty容器的，多数情况 下，人们都是独立运行这些goal的，比如：人们希望当键入mvn jetty:run后，工程就能完成编译后启动jetty,而jetty插件也确实是这样做的，它的run goal的mojo是这样声明的：  
  
其中@execute phase="test-compile"指明jetty:run这一goal会促使maven先build到test-compile阶段，再执行这个 goal.同样，对于jetty:run-war这个goal则要求先build到package阶段再执行该goal.  
  
而另外一个例子是exec插件的exec:java.  
  
这个goal也声明了execute的phase,但却是validate，这样，如果代码没有编译，执行这个goal就会出错， 所以多数情况下，人们总是使用下面的方式执行的：  
mvn clean compile exec:java