InfoQ Maven专栏（八）——常用Maven插件介绍（下）

May 11th, 2011

我们都知道Maven本质上是一个插件框架，它的核心并不执行任何具体的构建任务，所有这些任务都交给插件来完成，例如编译源代码是由maven- compiler-plugin完成的。进一步说，每个任务对应了一个插件目标（goal），每个插件会有一个或者多个目标，例如maven- compiler-plugin的compile目标用来编译位于src/main/java/目录下的主源码，testCompile目标用来编译位于src/test/java/目录下的测试源码。

用户可以通过两种方式调用Maven插件目标。第一种方式是将插件目标与生命周期阶段（lifecycle phase）绑定，这样用户在命令行只是输入生命周期阶段而已，例如Maven默认将maven-compiler-plugin的compile目标与compile生命周期阶段绑定，因此命令**mvn compile**实际上是先定位到compile这一生命周期阶段，然后再根据绑定关系调用maven-compiler-plugin的compile目标。第二种方式是直接在命令行指定要执行的插件目标，例如**mvn archetype:generate** 就表示调用maven-archetype-plugin的generate目标，这种带冒号的调用方式与生命周期无关。

认识上述Maven插件的基本概念能帮助你理解Maven的工作机制，不过要想更高效率地使用Maven，了解一些常用的插件还是很有必要的，这可以帮助你避免一不小心重新发明轮子。多年来Maven社区积累了大量的经验，并随之形成了一个成熟的插件生态圈。Maven官方有两个插件列表，第一个列表的GroupId为org.apache.maven.plugins，这里的插件最为成熟，具体地址为：http://maven.apache.org/plugins/index.html。第二个列表的GroupId为org.codehaus.mojo，这里的插件没有那么核心，但也有不少十分有用，其地址为：http://mojo.codehaus.org/plugins.html。

接下来笔者根据自己的经验介绍一些最常用的Maven插件，在不同的环境下它们各自都有其出色的表现，熟练地使用它们能让你的日常构建工作事半功倍。本文为下半部分。（上半部分内容参见InfoQ Maven专栏（七）——常用Maven插件介绍（上））

maven-resources-plugin

http://maven.apache.org/plugins/maven-resources-plugin/

为了使项目结构更为清晰，Maven区别对待Java代码文件和资源文件，maven-compiler-plugin用来编译Java代码，maven-resources-plugin则用来处理资源文件。默认的主资源文件目录是src/main/resources，很多用户会需要添加额外的资源文件目录，这个时候就可以通过配置maven-resources-plugin来实现。此外，资源文件过滤也是Maven的一大特性，你可以在资源文件中使用*${propertyName}*形式的Maven属性，然后配置maven-resources-plugin开启对资源文件的过滤，之后就可以针对不同环境通过命令行或者Profile传入属性的值，以实现更为灵活的构建。

maven-surefire-plugin

http://maven.apache.org/plugins/maven-surefire-plugin/

可能是由于历史的原因，Maven 2/3中用于执行测试的插件不是maven-test-plugin，而是maven-surefire-plugin。其实大部分时间内，只要你的测试类遵循通用的命令约定（以Test结尾、以TestCase结尾、或者以Test开头），就几乎不用知晓该插件的存在。然而在当你想要跳过测试、排除某些测试类、或者使用一些TestNG特性的时候，了解maven-surefire-plugin的一些配置选项就很有用了。例如 **mvn test -Dtest=FooTest** 这样一条命令的效果是仅运行FooTest测试类，这是通过控制maven-surefire-plugin的test参数实现的。

build-helper-maven-plugin

http://mojo.codehaus.org/build-helper-maven-plugin/

Maven默认只允许指定一个主Java代码目录和一个测试Java代码目录，虽然这其实是个应当尽量遵守的约定，但偶尔你还是会希望能够指定多个源码目录（例如为了应对遗留项目），build-helper-maven-plugin的add-source目标就是服务于这个目的，通常它被绑定到默认生命周期的generate-sources阶段以添加额外的源码目录。需要强调的是，这种做法还是不推荐的，因为它破坏了 Maven的约定，而且可能会遇到其他严格遵守约定的插件工具无法正确识别额外的源码目录。

build-helper-maven-plugin的另一个非常有用的目标是attach-artifact，使用该目标你可以以classifier的形式

选取部分项目文件生成附属构件，并同时install到本地仓库，也可以deploy到远程仓库。

exec-maven-plugin

http://mojo.codehaus.org/exec-maven-plugin/

exec-maven-plugin很好理解，顾名思义，它能让你运行任何本地的系统程序，在某些特定情况下，运行一个Maven外部的程序可能就是最简单的问题解决方案，这就是**exec:exec**的用途，当然，该插件还允许你配置相关的程序运行参数。除了exec目标之外，exec-maven-plugin还提供了一个java目标，该目标要求你提供一个mainClass参数，然后它能够利用当前项目的依赖作为classpath，在同一个JVM中运行该mainClass。有时候，为了简单的演示一个命令行Java程序，你可以在POM中配置好exec-maven-plugin的相关运行参数，然后直接在命令运行 **mvn exec:java** 以查看运行效果。

jetty-maven-plugin

http://wiki.eclipse.org/Jetty/Feature/Jetty_Maven_Plugin

在进行Web开发的时候，打开浏览器对应用进行手动的测试几乎是无法避免的，这种测试方法通常就是将项目打包成war文件，然后部署到Web容器中，再启动容器进行验证，这显然十分耗时。为了帮助开发者节省时间，jetty-maven-plugin应运而生，它完全兼容 Maven项目的目录结构，能够周期性地检查源文件，一旦发现变更后自动更新到内置的Jetty Web容器中。做一些基本配置后（例如Web应用的contextPath和自动扫描变更的时间间隔），你只要执行 **mvn jetty:run** ，然后在IDE中修改代码，代码经IDE自动编译后产生变更，再由jetty-maven-plugin侦测到后更新至Jetty容器，这时你就可以直接测试Web页面了。需要注意的是，jetty-maven-plugin并不是宿主于Apache或Codehaus的官方插件，因此使用的时候需要额外的配置settings.xml的pluginGroups元素，将org.mortbay.jetty这个pluginGroup加入。

versions-maven-plugin

http://mojo.codehaus.org/versions-maven-plugin/

很多Maven用户遇到过这样一个问题，当项目包含大量模块的时候，为他们集体更新版本就变成一件烦人的事情，到底有没有自动化工具能帮助完成这件事情呢？（当然你可以使用sed之类的文本操作工具，不过不在本文讨论范围）答案是肯定的，versions-maven- plugin提供了很多目标帮助你管理Maven项目的各种版本信息。例如最常用的，命令 **mvn versions:set -DnewVersion=1.1-SNAPSHOT** 就能帮助你把所有模块的版本更新到1.1-SNAPSHOT。该插件还提供了其他一些很有用的目标，display-dependency- updates能告诉你项目依赖有哪些可用的更新；类似的display-plugin-updates能告诉你可用的插件更新；然后use- latest-versions能自动帮你将所有依赖升级到最新版本。最后，如果你对所做的更改满意，则可以使用 **mvn versions:commit** 提交，不满意的话也可以使用 **mvn versions:revert** 进行撤销。

小结

本文介绍了一些最常用的Maven插件，这里指的“常用”是指经常需要进行配置的插件，事实上我们用Maven的时候很多其它插件也是必须的，例如默认的编译插件maven-compiler-plugin和默认的打包插件maven-jar-plugin，但因为很少需要对它们进行配置，因此不在本文讨论范围。了解常用的Maven插件能帮助你事半功倍地完成项目构建任务，反之你就可能会因为经常遇到一些难以解决的问题而感到沮丧。本文介绍的插件基本能覆盖大部分Maven用户的日常使用需要，如果你真有非常特殊的需求，自行编写一个Maven插件也不是难事，更何况还有这么多开放源代码的插件供你参考。

本文的这个插件列表并不是一个完整列表，读者有兴趣的话也可以去仔细浏览一下Apache和Codehaus Mojo的Maven插件列表，以得到一个更为全面的认识。最后，在线的Maven仓库搜索引擎如http://search.maven.org/也能帮助你快速找到自己感兴趣的Maven插件。
