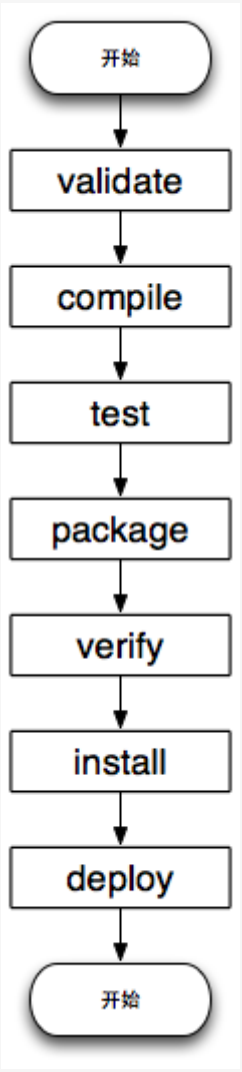
Maven POM

Maven 构建配置文件

Maven 构建生命周期

Maven 构建生命周期定义了一个项目构建跟发布的过程。

一个典型的 Maven 构建（build）生命周期是由以下几个阶段的序列组成的：



阶段	处理	描述
验证 validate	验证项目	验证项目是否正确且所有必须信息是可用的
编译 compile	执行编译	源代码编译在此阶段完成
测试 Test	测试	使用适当的单元测试框架（例如JUnit）运行测试。
包装 package	打包	创建JAR/WAR包如在 pom.xml 中定义提及的包
检查 verify	检查	对集成测试的结果进行检查，以保证质量达标
安装 install	安装	安装打包的项目到本地仓库，以供其他项目使用

阶段	处理	描述
部署 deploy	部署	拷贝最终的工程包到远程仓库中，以共享给其他开发人员和工程

为了完成 default 生命周期，这些阶段（包括其他未在上面罗列的生命周期阶段）将被按顺序地执行。

Maven 有以下三个标准的生命周期：

- **clean**：项目清理的处理
- **default(或 build)**：项目部署的处理
- **site**：项目站点文档创建的处理

构建阶段由插件目标构成

一个插件目标代表一个特定的任务（比构建阶段更为精细），这有助于项目的构建和管理。这些目标可能被绑定到多个阶段或者无绑定。不绑定到任何构建阶段的目标可以在构建生命周期之外通过直接调用执行。这些目标的执行顺序取决于调用目标和构建阶段的顺序。

例如，考虑下面的命令：

clean 和 pakage 是构建阶段，dependency:copy-dependencies 是目标

```
mvn clean dependency:copy-dependencies package
```

这里的 clean 阶段将会被首先执行，然后 dependency:copy-dependencies 目标会被执行，最终 package 阶段被执行。

Clean 生命周期

当我们执行 mvn post-clean 命令时，Maven 调用 clean 生命周期，它包含以下阶段：

- pre-clean：执行一些需要在clean之前完成的工作
- clean：移除所有上一次构建生成的文件
- post-clean：执行一些需要在clean之后立刻完成的工作

mvn clean 中的 clean 就是上面的 clean，在一个生命周期中，运行某个阶段的时候，它之前的所有阶段都会被运行，也就是说，如果执行 mvn clean 将运行以下两个生命周期阶段：

```
pre-clean, clean
```

如果我们运行 mvn post-clean ，则运行以下三个生命周期阶段：

```
pre-clean, clean, post-clean
```

我们可以通过在上面的 clean 生命周期的任何阶段定义目标来修改这部分的操作行为。

在下面的例子中，我们将 maven-antrun-plugin:run 目标添加到 pre-clean、clean 和 post-clean 阶段中。这样我们可以在 clean 生命周期的各个阶段显示文本信息。

我们已经在 C:\MVN\project 目录下创建了一个 pom.xml 文件。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.companyname.projectgroup</groupId>
<artifactId>project</artifactId>
<version>1.0</version>
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-antrun-plugin</artifactId>
<version>1.1</version>
<executions>
<execution>
<id>id.pre-clean</id>
<phase>pre-clean</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>pre-clean phase</echo>
</tasks>
</configuration>
</execution>
<execution>
<id>id.clean</id>
<phase>clean</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>clean phase</echo>
</tasks>
</configuration>
</execution>
<execution>
<id>id.post-clean</id>
<phase>post-clean</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>post-clean phase</echo>
</tasks>
</configuration>
</execution>
</executions>
```

```
</plugin>
</plugins>
</build>
</project>
```

现在打开命令控制台，跳转到 pom.xml 所在目录，并执行下面的 mvn 命令。

```
C:\MVN\project>mvn post-clean
```

Maven 将会开始处理并显示 clean 生命周期的所有阶段。

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building Unnamed - com.companyname.projectgroup:project:jar:1.0
[INFO]    task-segment: [post-clean]
[INFO] ------------------------------------------------------------------------
[INFO] [antrun:run {execution: id.pre-clean}]
[INFO] Executing tasks
     [echo] pre-clean phase
[INFO] Executed tasks
[INFO] [clean:clean {execution: default-clean}]
[INFO] [antrun:run {execution: id.clean}]
[INFO] Executing tasks
     [echo] clean phase
[INFO] Executed tasks
[INFO] [antrun:run {execution: id.post-clean}]
[INFO] Executing tasks
     [echo] post-clean phase
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: < 1 second
[INFO] Finished at: Sat Jul 07 13:38:59 IST 2012
[INFO] Final Memory: 4M/44M
[INFO] ------------------------------------------------------------------------
```

你可以尝试修改 mvn clean 命令，来显示 pre-clean 和 clean，而在 post-clean 阶段不执行任何操作。

Default (Build) 生命周期

这是 Maven 的主要生命周期，被用于构建应用，包括下面的 23 个阶段：

生命周期阶段	描述
validate	检查工程配置是否正确，完成构建过程的所有必要信息是否能够获取到。
initialize	初始化构建状态，例如设置属性。

生命周期阶段	描述
generate-sources	生成编译阶段需要包含的任何源码文件。
process-sources	处理源代码，例如，过滤任何值（filter any value）。
generate-resources	生成工程包中需要包含的资源文件。
process-resources	拷贝和处理资源文件到目的目录中，为打包阶段做准备。
compile	编译工程源码。
process-classes	处理编译生成的文件，例如 Java Class 字节码的加强和优化。
generate-test-sources	生成编译阶段需要包含的任何测试源代码。
process-test-sources	处理测试源代码，例如，过滤任何值（filter any values)。
test-compile	编译测试源代码到测试目的目录。
process-test-classes	处理测试代码文件编译后生成的文件。
test	使用适当的单元测试框架（例如JUnit）运行测试。
prepare-package	在真正打包之前，为准备打包执行任何必要的操作。
package	获取编译后的代码，并按照可发布的格式进行打包，例如 JAR、WAR 或者 EAR 文件。
pre-integration-test	在集成测试执行之前，执行所需的操作。例如，设置所需的环境变量。
integration-test	处理和部署必须的工程包到集成测试能够运行的环境中。
post-integration-test	在集成测试被执行后执行必要的操作。例如，清理环境。
verify	运行检查操作来验证工程包是有效的，并满足质量要求。
install	安装工程包到本地仓库中，该仓库可以作为本地其他工程的依赖。
deploy	拷贝最终的工程包到远程仓库中，以共享给其他开发人员和工程。

有一些与 Maven 生命周期相关的重要概念需要说明：

当一个阶段通过 Maven 命令调用时，例如 mvn compile，只有该阶段之前以及包括该阶段在内的所有阶段会被执行。

不同的 maven 目标将根据打包的类型（JAR / WAR / EAR），被绑定到不同的 Maven 生命周期阶段。

在下面的例子中，我们将 maven-antrun-plugin:run 目标添加到 Build 生命周期的一部分阶段中。这样我们可以显示生命周期的文本信息。

我们已经更新了 C:\MVN\project 目录下的 pom.xml 文件。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.companyname.projectgroup</groupId>
<artifactId>project</artifactId>
<version>1.0</version>
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-antrun-plugin</artifactId>
<version>1.1</version>
<executions>
<execution>
<id>id.validate</id>
<phase>validate</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>validate phase</echo>
</tasks>
</configuration>
</execution>
<execution>
<id>id.compile</id>
<phase>compile</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>compile phase</echo>
</tasks>
</configuration>
</execution>
<execution>
<id>id.test</id>
<phase>test</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>test phase</echo>
</tasks>
</configuration>
</execution>
<execution>
<id>id.package</id>
```

```
<phase>package</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>package phase</echo>
</tasks>
</configuration>
</execution>
<execution>
<id>id.deploy</id>
<phase>deploy</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>deploy phase</echo>
</tasks>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
</project>
```

现在打开命令控制台，跳转到 pom.xml 所在目录，并执行以下 mvn 命令。

```
C:\MVN\project>mvn compile
```

Maven 将会开始处理并显示直到编译阶段的构建生命周期的各个阶段。

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building Unnamed - com.companyname.projectgroup:project:jar:1.0
[INFO]    task-segment: [compile]
[INFO] ------------------------------------------------------------------------
[INFO] [antrun:run {execution: id.validate}]
[INFO] Executing tasks
     [echo] validate phase
[INFO] Executed tasks
[INFO] [resources:resources {execution: default-resources}]
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources,
i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\MVN\project\src\main\resources
[INFO] [compiler:compile {execution: default-compile}]
[INFO] Nothing to compile - all classes are up to date
[INFO] [antrun:run {execution: id.compile}]
```

```
[INFO] Executing tasks
     [echo] compile phase
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2 seconds
[INFO] Finished at: Sat Jul 07 20:18:25 IST 2012
[INFO] Final Memory: 7M/64M
[INFO] ------------------------------------------------------------------------
```

命令行调用

在开发环境中，使用下面的命令去构建、安装工程到本地仓库

```
mvn install
```

这个命令在执行 install 阶段前，按顺序执行了 default 生命周期的阶段（validate，compile，package，等等），我们只需要调用最后一个阶段，如这里是 install。

在构建环境中，使用下面的调用来纯净地构建和部署项目到共享仓库中

```
mvn clean deploy
```

这行命令也可以用于多模块的情况下，即包含多个子项目的项目，Maven 会在每一个子项目执行 clean 命令，然后再执行 deploy 命令。

Site 生命周期

Maven Site 插件一般用来创建新的报告文档、部署站点等。

- pre-site：执行一些需要在生成站点文档之前完成的工作
- site：生成项目的站点文档
- post-site： 执行一些需要在生成站点文档之后完成的工作，并且为部署做准备
- site-deploy：将生成的站点文档部署到特定的服务器上

这里经常用到的是site阶段和site-deploy阶段，用以生成和发布Maven站点，这可是Maven相当强大的功能，Manager比较喜欢，文档及统计数据自动生成，很好看。 在下面的例子中，我们将 maven-antrun-plugin:run 目标添加到 Site 生命周期的所有阶段中。这样我们可以显示生命周期的所有文本信息。

我们已经更新了 C:\MVN\project 目录下的 pom.xml 文件。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>
<groupId>com.companyname.projectgroup</groupId>
<artifactId>project</artifactId>
<version>1.0</version>
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-antrun-plugin</artifactId>
<version>1.1</version>
<executions>
<execution>
<id>id.pre-site</id>
<phase>pre-site</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>pre-site phase</echo>
</tasks>
</configuration>
</execution>
<execution>
<id>id.site</id>
<phase>site</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>site phase</echo>
</tasks>
</configuration>
</execution>
<execution>
<id>id.post-site</id>
<phase>post-site</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>post-site phase</echo>
</tasks>
</configuration>
</execution>
<execution>
<id>id.site-deploy</id>
<phase>site-deploy</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
```

```
<tasks>
<echo>site-deploy phase</echo>
</tasks>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
</project>
```

现在打开命令控制台，跳转到 pom.xml 所在目录，并执行以下 mvn 命令。

```
C:\MVN\project>mvn site
```

Maven 将会开始处理并显示直到 site 阶段的 site 生命周期的各个阶段。

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building Unnamed - com.companyname.projectgroup:project:jar:1.0
[INFO]    task-segment: [site]
[INFO] ------------------------------------------------------------------------
[INFO] [antrun:run {execution: id.pre-site}]
[INFO] Executing tasks
     [echo] pre-site phase
[INFO] Executed tasks
[INFO] [site:site {execution: default-site}]
[INFO] Generating "About" report.
[INFO] Generating "Issue Tracking" report.
[INFO] Generating "Project Team" report.
[INFO] Generating "Dependencies" report.
[INFO] Generating "Project Plugins" report.
[INFO] Generating "Continuous Integration" report.
[INFO] Generating "Source Repository" report.
[INFO] Generating "Project License" report.
[INFO] Generating "Mailing Lists" report.
[INFO] Generating "Plugin Management" report.
[INFO] Generating "Project Summary" report.
[INFO] [antrun:run {execution: id.site}]
[INFO] Executing tasks
     [echo] site phase
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3 seconds
[INFO] Finished at: Sat Jul 07 15:25:10 IST 2012
```

```
[INFO] Final Memory: 24M/149M
[INFO] ------------------------------------------------------------------------```
```

Maven POM

Maven 构建配置文件

点我分享笔记