Maven 仓库 Maven 构建 Java 项目

Maven 插件

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

每个生命周期中都包含着一系列的阶段(phase)。这些 phase 就相当于 Maven 提供的统一的接口，然后这些 phase 的实现由 Maven 的插件来完成。

我们在输入 mvn 命令的时候 比如 `mvn clean`，clean 对应的就是 Clean 生命周期中的 clean 阶段。但是 clean 的具体操作是由 **maven-clean-plugin** 来实现的。

所以说 Maven 生命周期的每一个阶段的具体实现都是由 Maven 插件实现的。

Maven 实际上是一个依赖插件执行的框架，每个任务实际上是由插件完成。Maven 插件通常被用来：

- 创建 jar 文件
- 创建 war 文件
- 编译代码文件
- 代码单元测试
- 创建工程文档
- 创建工程报告

插件通常提供了一个目标的集合，并且可以使用下面的语法执行：

```
<code>mvn [plugin-name]:[goal-name]</code>
```

例如，一个 Java 工程可以使用 maven-compiler-plugin 的 compile-goal 编译，使用以下命令：

```
<code>mvn compiler:compile</code>
```

插件类型

Maven 提供了下面两种类型的插件：

类型	描述
Build plugins	在构建时执行，并在 pom.xml 的 元素中配置。
Reporting plugins	在网站生成过程中执行，并在 pom.xml 的 元素中配置。

下面是一些常用插件的列表：

插件	描述
clean	构建之后清理目标文件。删除目标目录。
compiler	编译 Java 源文件。
surefile	运行 JUnit 单元测试。创建测试报告。
jar	从当前工程中构建 JAR 文件。
war	从当前工程中构建 WAR 文件。
javadoc	为工程生成 Javadoc。
antrun	从构建过程的任意一个阶段中运行一个 ant 任务的集合。

实例

我们已经在我们的例子中大量使用了 **maven-antrun-plugin** 来输出数据到控制台上。请查看 **Maven - 构建配置文件** 章节。让我们用一种更好的方式理解这部分内容，在 C:\MVN\project 目录下创建一个 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.clean</id>
<phase>clean</phase>
<goals>
<goal>run</goal>
</goals>
<configuration>
<tasks>
<echo>clean phase</echo>
</tasks>
</configuration>
</execution>
</executions>
</plugin>
```

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

接下来，打开命令终端跳转到 pom.xml 所在的目录，并执行下面的 mvn 命令。

```
mvn clean
```

Maven 将开始处理并显示 clean 生命周期的 clean 阶段。

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building Unnamed - com.companyname.projectgroup:project:jar:1.0
[INFO]    task-segment: [post-clean]
[INFO] ------------------------------------------------------------------------
[INFO] [clean:clean {execution: default-clean}]
[INFO] [antrun:run {execution: id.clean}]
[INFO] Executing tasks
     [echo] 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] ------------------------------------------------------------------------
```

上面的例子展示了以下关键概念：

- 插件是在 pom.xml 中使用 plugins 元素定义的。
- 每个插件可以有多个目标。
- 你可以定义阶段，插件会使用它的 phase 元素开始处理。我们已经使用了 **clean** 阶段。
- 你可以通过绑定到插件的目标的方式来配置要执行的任务。我们已经绑定了 **echo** 任务到 maven-antrun-plugin 的 **run** 目标。
- 就是这样，Maven 将处理剩下的事情。它将下载本地仓库中获取不到的插件，并开始处理。

← Maven 仓库 Maven 构建 Java 项目 →