# Maven插件

1. <https://www.yiibai.com/maven/maven_plugins.html>

2. <https://www.iteye.com/blog/juvenshun-232916>

## 一、什么是Maven的插件?

Maven是一个执行插件的框架，每一个任务实际上是有由插件完成的。

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

如果你理解设计模式中的模板方法（Template Method）模式，那么这会变得很好理解。

在模板方法模式中，父类定义了一些抽象方法，并且决定这些方法的调用顺序，子类只需要实现这些方法。

比如沏茶这个用例，父类定义的抽象方法有：烧开水，洗茶杯，倒水，放茶叶，并且有一个沏茶方法来顺序调用这些方法。而子类会去实现如何烧开水，如何洗茶杯，如何倒水，如何放茶叶。实例化一个子类之后，调用沏茶方法，这些步骤就会依次执行了。

同样的道理，在Maven中，生命周期定义了编译，测试，打包，部署等等阶段，而插件会去实现这些对应的阶段，如上面提到的Compiler插件实现了如何编译。

一个插件通常提供了一组目标，而目标通常是和生命周期中的某个阶段绑定在一起，一个插件目标可以绑定到多个生命周期的阶段，可使用如下语法来执行

mvn [plugin-name]:[goal-name]

## 二、插件类型：

Maven提供了以下两种类型插件：

|  |  |
| --- | --- |
| 类型 | 描述 |
| 构建插件 | 在生成过程中执行,并在pom.xml中的<build/>元素进行配置 |
| 报告插件 | 在网站生成期间执行,在pom.xml中的<reporting/>元素进行配置 |

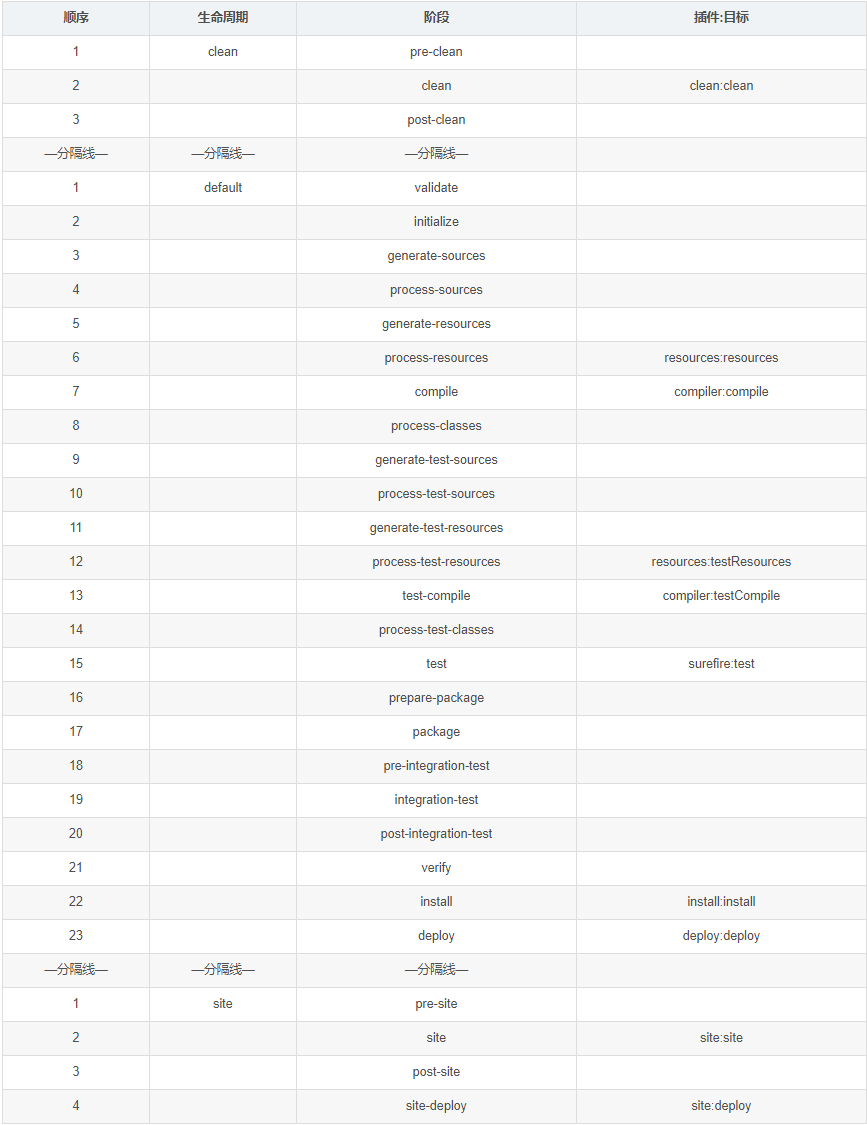
## 三、常见插件列表：

|  |  |
| --- | --- |
| 插件 | 描述 |
| clean | 编译后的清理目标,删除目标目录 |
| compile | 编译Java源文件 |
| surefire | 进行单元测试,创建测试报告 |
| jar | 从当前项目构建JAR文件 |
| war | 从当前项目构建WAR文件 |
| javadoc | 产生用于该项目的javadoc |
| antrun | 从构建的任何阶段运行一组Ant任务 |

## 四、默认生命周期阶段与插件目标的绑定关系：

Maven为实现快速构建，有一套内置的插件绑定。

三套生命周期的插件绑定如下（其实是各个生命周期阶段与插件目标的绑定）



## 五、示例：

### 1、antrun插件：

<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>

【解读】

这段代码的意思是：

当进行到了Clean生命周期的clean阶段(<phase>clean</phase>)的时候，执行antrun插件(maven-antrun-plugin)的run目标(<goal>run</goal>)，而该目标所要做的任务是输出“clean phase”(<tasks><echo>clean phase</echo></tasks>)。

### 2、Tomcat插件：

<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>

<!-- 配置Tomcat插件 -->

<plugin>

<!-- https://mvnrepository.com/artifact/org.apache.tomcat/tomcat-catalina -->

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.1</version>

<configuration>

<path>/</path>

<port>8080</port>

</configuration>

</plugin>

</plugins>

</build>

</project>

配置了Tomcat插件之后，运行web项目的时候，就不用我们本地下载Tomcat服务器

### 3、compile插件：

<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>

<!-- 配置编译插件 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<encoding>utf-8</encoding>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin></plugins>

</build>

</project>