03 生命周期和插件

C-2 创建: 张林伟, 最后修改: 张林伟 2018-08-03 19:08

目录

生命周期

Maven拥有 3 套相互独立的生命周期：

- clean：清理项目
- default：构建项目
- site：建立项目站点

clean生命周期

1. pre-clean：执行一些清理前需要完成的工作
2. clean：清理上一次构建生成的文件
3. post-clean：执行一些清理后需要完成的工作

default生命周期

它是所有生命周期中最核心的部分

1. validate
2. initialize
3. generate-sources
4. process-sources：处理项目主资源文件。对 src/main/resources 目录内容进行变量替换等工作后，复制到项目输出的主 classpath 目录中
5. generate-resources
6. process-resources
7. compile：编译项目的主源码。编译 src/main/java 目录下的 java 文件至项目输出的主 classpath 目录中
8. process-classes
9. generate-test-sources
10. process-test-sources：处理项目测试资源文件。对 src/test/resources 目录内容进行变量替换等工作后，复制到项目输出的测试 classpath 目录中
11. generate-test-resources
12. process-test-resources
13. test-compile：编译项目的测试代码。编译 src/test/java 目录下的 java 文件至项目输出的测试 classpath 目录中
14. process-test-classes
15. test：使用单元测试框架运行测试，测试代码不会被打包或部署
16. prepare-package
17. package：接受编译好的代码，打包成可发布的格式，如jar
18. pre-integration-test
19. integration-test
20. post-integration-test
21. verify
22. install：将包安装到 Maven 本地仓库
23. deploy：将最终的包复制到远程仓库

site生命周期

1. pre-site：执行一些生成项目站点之前需要完成的工作

2. site：生成项目站点文档
3. post-site：执行一些生成项目站点之前需要完成的工作
4. site-deploy：将生成的项目站点发布到服务器上

插件绑定

Maven 的生命周期与插件相互绑定，用以完成实际的构建任务。

例如，

maven-clean-plugin: clean 与 clean 生命周期的 clean 阶段绑定；

maven-site-plugin: site 与 site 生命周期的 site 阶段绑定，maven-site-plugin: deploy 与 site 生命周期的 site-deploy 阶段绑定。

生命周期阶段	插件目标	执行任务
process-resources	maven-resources-plugin:resources	复制主资源文件至主输出目录
compile	maven-compile-plugin:compile	编译主代码至主输出目录
process-test-resources	maven-resources-plugin:testRresources	复制测试资源文件至测试输出目录
test-compile	maven-compiler-plugin:testCompile	编译测试代码至测试输出目录
test	maven-surefire-plugin:test	执行测试用例
package	maven-jar-plugin:jar	创建项目jar包
install	maven-install-plugin:install	将项目输出构件安装到本地仓库
deploy	maven-deploy-plugin:deploy	将项目输出构件部署到远程仓库

生命周期某些阶段没有绑定任何插件，因此没有任何实际行为。

自定义绑定

例如，将 maven-source-plugin: jar-no-fork 目标绑定到 default 生命周期的 verify 阶段上

代码块 XML

```xml
<build>
  <plugins>
    ...
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-source-plugin</artifactId>
      <version>3.0.1</version>
      <executions>
        <execution>
          <id>attach-sources</id>
          <phase>verify</phase>
          <goals>
            <goal>jar-no-fork</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
    ...
  </plugins>
</build>
```

该例中配置了一个 id 为 attach-sources 的任务，通过 phrase 配置，将其绑定到 verify 生命周期阶段上，再通过 goals 配置指导要执行的插件目标。运行 **mvn verify** 即可看到效果。

如果多个目标被绑定到同一个阶段，插件声明的顺序决定了目标的执行顺序。