

## Buildtools 2:特定于语言的构建工具

约瑟夫·哈利特

2023 年 1 月 13 日



上次…

我们讨论过Make作为编译代码的工具

►我们讨论了如何使用Make在不同格式之间转换文档

但有一件事Make s做不到……

# 版本控制

现代开发广泛使用外部库.....

但Make在处理它们时很垃圾：

▶ 不知道如何获取依赖项 ▶ 不跟踪源版本比对象

新的版本

LanguageTool是一个很酷的小型 Java 语法检查器： ▶ 仅

该工具的核心就使用了多少个库？

mvn 依赖项 :tree -D 输出类型=点 |点-Tpdf



## 古时候…

传统上,您必须手动下载所有依赖项……

►然后编译并安装它们►非常繁琐且容易出错

所以我们自动化了它!

## 现代构建工具

(几乎)每种语言都有自己的库管理工具

► 让开发人员指定依赖项 ► 告诉编译器如何  
重建项目

...这意味着对于您使用的每种语言,您都需要学习其构建工具...

► 耶?

(老实说,我还在用Make ,但我老了,脾气暴躁)

所以我们有……

Commonlisp ASDF 和 Quicklisp

去构建

哈斯克尔阴谋集团

Java Ant、Maven、Gradle...

JavaScript NPM Perl CPAN

Python Distutils 和 requirements.txt

克兰

Ruby Gem Rust

Cargo LATEX

CTAN 和 TeXlive …… 等等。

而且他们都是不同的

它们之间几乎没有相似之处。 ▶你需要学习你所使用的那些。 ▶我们将在实验室中使用Maven for Java 进行一些操作



**Maven 快速入门** mkdir /tmp/src cd /

tmp/src mvn

archetype:generate \

-DgroupId=uk.ac.bristol.cs \

-DartifactId=hello \

-DarchetypeArtifactId=maven-archetype-quickstart \

-DinteractiveMode=false

扫描项目\ldots{}

-----< org.apache.maven:standalone-pom > -----

构建 Maven 存根项目 (无 POM)1

-----[ pom ]-----

-----

>>> maven-archetype-plugin:3.2.1:generate (default-cli) > 生成源@独立

<<< maven-archetype-plugin:3.2.1:generate (default-cli) < 生成源@独立

--- maven-

archetype-

plugin:3.2.1:generate (default-cli) @standalone-pom ---

批量生成项目

-----

-----

.....吐出所有这些之后.....

查找 /tmp/src -输入 f

▶ ( “/tmp/src/hello/pom.xml” )

▶ ( “/tmp/src/hello/src/main/java/uk/ac/bristol/cs/App.java” ) ▶ ( “/  
tmp/ src/hello/src/test/java/uk/ac/bristol/cs/AppTest.java” )

## pom.xml

```
<项目 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/maven-v4_0_0.xsd > <modelVersion>4.0.0</modelVersion>  
    <groupId>uk.ac.bristol.cs< /groupId> <artifactId>  
    你好</artifactId> <打包>jar</打包> <版本>1.0-  
    SNAPSHOT</版本> <名称>你好</名称>  
    <url>http://maven.apache.org< /url>  
    <依赖项> <依赖项> <groupId>junit</groupId>  
    <artifactId>junit</  
    artifactId> <版本>3.8.1</版本> <范围>测试</范围>  
    </依赖项> </依赖项>  
    </项目>
```

## 如果我们尝试构建... mvn package

扫描项目\ldots{}

-----&lt; uk.ac.bristol.cs:hello &gt;-----

构建你好 1.0 -快照

-----[罐

子]-----

--- maven-resources-plugin:2.6:resources

(default-

resources) @ hello ---

使用平台编码 (实际上是 US-ASCII)复制过滤后的资源,即构建 i

跳过不存在的资源目录 /tmp/src /hello/src/main/resources

--- maven-compiler-

plugin:3.1:compile (default-compile) @ hello ---

检测到更改- 重新编译模块!

文件编码尚未设置,使用平台编码 US-ASCII,即构建为 plat

编译 1 个

源文件到 /tmp/src/hello/target/classes

编译错误:

-----

-----

-----

来源不再支持选项 5.使用 7 或更高版

本。

不再支持目标选项 5.使用 7 或更高版本。

2 个错误

## 让我们解决这个问题吧？

(要么就是这样,要么安装一个古老的Java编译器.....) ed /tmp/src/hello/

pom.xml <<EOF 10i <properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target> </属性>

。

瓦克

EOF

639778

## 如果我们再次尝试构建... mvn package

扫

描项目\ldots{}

----->

```
uk.ac.bristol.cs:hello >-----
```

大楼你好1.0-

## SNAPSHOT

[jar]-----

```
--- maven-resources-plugin:2.6 :resources
```

(default-

resources) @ hello ---

使用平台编码 (实际上是 US-ASCII)复制过滤后的资源,即构建 i

跳过不存在的资源目录 /tmp/src/hello/src/main/resources

```
--- maven-compiler-
```

plugin:3.1:compile (默认编译) @ hello ---

检测到更改 - 重新编译模块!

文件编

码尚未设置,使用平台编码 US-ASCII,即构建为 plat

编译 1 个源文件到 /tmp/src/hello/

target/classes

```
--- maven-resources-
```

```
plugin:2.6:testResources (default-testResources) @ hello ---
```

使用平台编码（实际上是 US-ASCII）复制过滤

后的资源,即 build i

跳过不存在的资源目录 /tmp/src/hello/src/test/

## resources

```
--- maven-compiler-plugin :3.1:testCompile (默认testCompile)@你好---
```

# 有人真的知道为什么 Java 的东西如此冗长吗？

查找 /tmp/src -输入 f

```

▶ ( "/tmp/src/hello/pom.xml" )
▶ ( "/tmp/src/hello/src/main/java/uk/ac/bristol/cs/App.java" ) ▶ ( "/"
tmp/ src/hello/src/test/java/uk/ac/bristol/cs/AppTest.java" ) ▶ (" /tmp/
src/hello/target/maven-status/maven-compiler-plugin/compile/default-compile /
  createdFiles.lst" ) ▶ (" /
tmp/src/hello/target/maven-status/maven-compiler-plugin/compile/default-compile/
  inputFiles.lst" )
▶ (" /tmp/src/hello/target/maven-status/maven-compiler-plugin/testCompile/default-
  testCompile/createdFiles.lst" )
▶ ( "/tmp/src/hello/target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/
  inputFiles.lst" ) ▶ ( "/tmp/
src/hello/target/classes/uk/ac /bristol/cs/App.class" ) ▶ ( "/tmp/src/
hello/target/test-classes/uk/ac/bristol/cs/AppTest.class" ) ▶ (" /tmp/src/hello/
target /surefire-reports/uk.ac.bristol.cs.AppTest.txt" ) ▶ ( "/tmp/src/hello/target/
surefire-reports/TEST-uk.ac.bristol.cs.AppTest.xml" ) ▶ ( "/tmp/src/hello/target/maven-
archiver/pom.properties" ) ▶ ( "/tmp/src/hello/target/hello-1.0-
SNAPSHOT.jar" )

```

## 其他有用的命令

`mvn test` 运行测试套件

`mvn install` 将 JAR 安装到本地 JAR 包中

`mvn clean` 删除所有内容

如果我有点刻薄…… <https://>

[gradle.org](https://gradle.org) 更好的 Java 构建工具

(这并不是在所有地方都有效,当你尝试做更复杂的事情时,它比 Maven 更糟糕.....)



# 包起来

►存在特定于语言的构建工具►您可能应该使用它们  
►（但我仍然使用更好的 `ol` `make` 更多）

有时你会

发现你拉了一个项目,它使用了某种构建系统,你只知道你将不得不花一天的时间来对抗它。…请不要使用 CMake。