👷‍♀️

Built Tools 2: Language Specific Tools (1)

In modern software development, lots of the code you use will come from external libraries, e.g. in Python

Versioning

Make is **terrible** and dealing with **external libraries**

- e.g. there is X library available for solving my problem, lets use that instead of building a Hash table for example
- Make doesn't know how to fetch **dependencies**
    - **Dependancies -** are relationships between software components whereby one component relies on another to function correctly
- it cant track versions beyond **output is newer than input**

Before this modern era you had to download all the dependancies manually

- then compile and install them
- this was tedious and **very** error prone

Therefore it became **automated**

Modern Build Tooling

- (almost) every programming language these days comes with its own **library management tooling**
- this lets devs specify dependancies

    - i.e. which libraries they are using and when
    - also tells the compiler how to rebuild the project

  Java Build Tools

This means that for each programming language you use, you will **need to learn its build tools…**

- and they are all different and incompatible

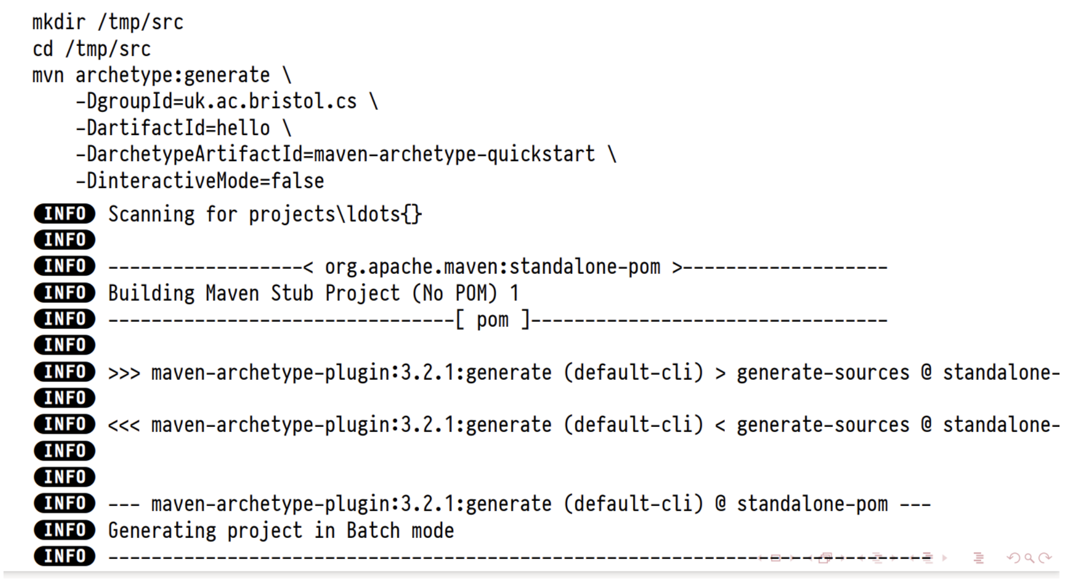
Java Build Tool: Maven

Maven is a powerful project management and comprehension tool used primarily for Java projects. It is designed to simplify the build process for Java projects by providing a standardized way to build projects, manage dependencies, deploy artifacts, and document the steps involved. Maven utilizes an XML file (pom.xml) to describe the project configuration, dependencies, plugins, and other aspects. Here's a breakdown of its key features and components:

useful maven commands:

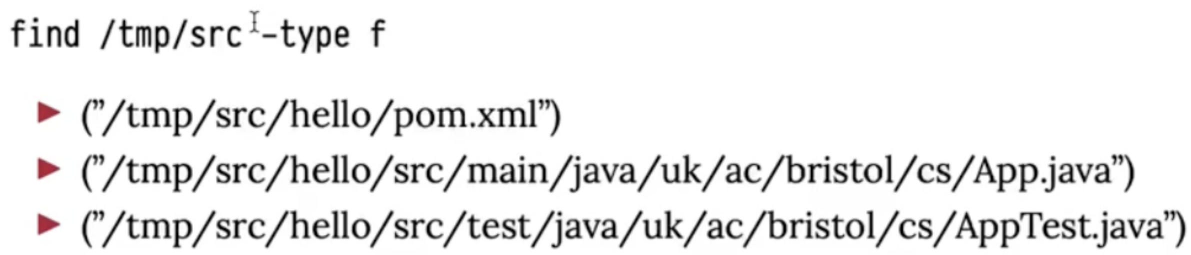
```
mvn test // runs the test suite

mvn install // install JAR into local JAR packages

mvn clean // delete everything

mvn package // builds the project
```

Maven Quickstart



```
mkdir /tmp/src
cd /tmp/src
mvn archetype:generate \
    -DgroupId=uk.ac.bristol.cs \
    -DartifactId=hello \
    -DarchetypeArtifactId=maven-archetype-quickstart \
    -DinteractiveMode=false
INFO Scanning for projects\ldots{}
INFO
INFO ------------------< org.apache.maven:standalone-pom >-------------------
INFO Building Maven Stub Project (No POM) 1
INFO --------------------------------[ pom ]---------------------------------
INFO
INFO >>> maven-archetype-plugin:3.2.1:generate (default-cli) > generate-sources @ standalone-
INFO
INFO <<< maven-archetype-plugin:3.2.1:generate (default-cli) < generate-sources @ standalone-
INFO
INFO
INFO --- maven-archetype-plugin:3.2.1:generate (default-cli) @ standalone-pom ---
INFO Generating project in Batch mode
INFO ----------------------------------------------------------------------------
```

- in Java your package should be named after your URL
  `DgroupOd=uk.ac.bristol.ac`



```
find /tmp/src -type f
```

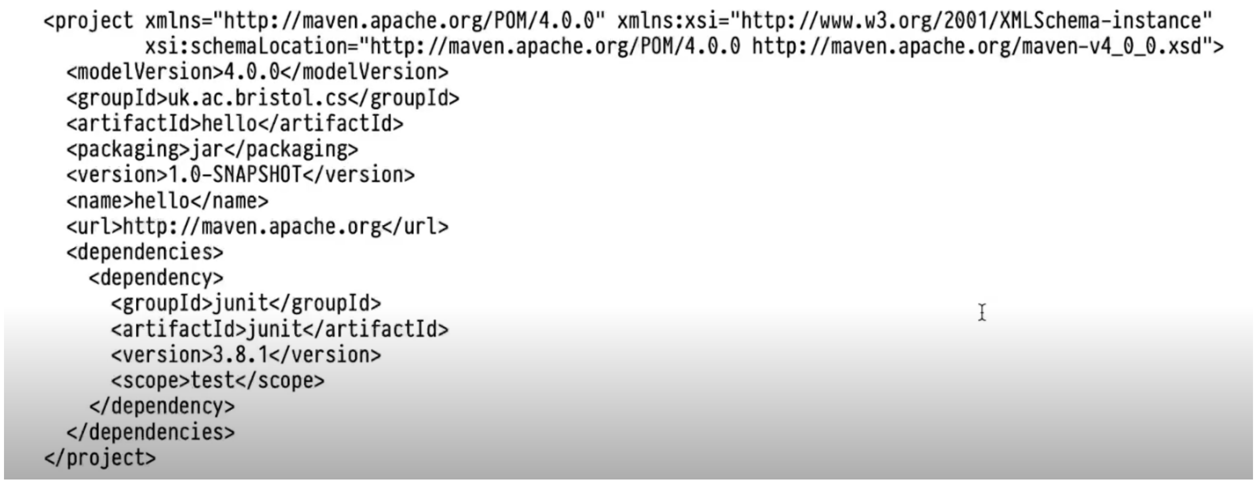
- ("/tmp/src/hello/pom.xml")
- ("/tmp/src/hello/src/main/java/uk/ac/bristol/cs/App.java")
- ("/tmp/src/hello/src/test/java/uk/ac/bristol/cs/AppTest.java")

- this has created a pom.xml file
- a java class file
- and a java test file

pom.xml

1. Project Object Model (POM)

The core of Maven's configuration is the Project Object Model (POM), defined in a `pom.xml` file. This file contains information about the project and configuration details used by Maven to build the project. Key elements include project dependencies, build directory, source directory, test source directory, plugin configurations, and goals.



```
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>uk.ac.bristol.cs</groupId>
  <artifactId>hello</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>hello</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

- the important part is the dependancies block in the above
- Maven will know to fetch junit at a *specific version*

Adding dependencies to pm.xml