



Build Tools and Maven

DevOps Practitioner

transforming performance
through learning

Outline

- **What are build tools**
 - Why are they useful
 - Different build tools available
- **Introducing Maven**
 - What is Maven
 - Why do we like it
 - Maven Principles
- **Installing Maven**
 - Pulling down an archetype
 - Running a project

What are build tools

- **Build managers aim to simplify the process of building, testing and running programs**
 - Deal with dependencies
 - Run automated tests
 - Package all the files into a single executable
 - Package and launch web servers as needed
 - Check code style against guidelines
- **Part of the Continuous Integration / Continuous Delivery tool chain**
 - Builds need to be automated
 - Manual builds are error prone and slow

3

Wikipedia's List: http://en.wikipedia.org/wiki/List_of_build_automation_software

Prevents us from a lot of bad things happening: software version changed, Plugin is missing, library can change etc. So process is very unstable. Tools will make the process stable.

Deal with dependencies – finds libraries and makes sure, that they have same version.

Run automated tests – people don't like boring things. Test can be done automatically.

Package all the files into a single executable – automatically can build jar or war

Package and launch web servers as needed – possible to build and launch web server from a command line

Check code style against guidelines – can check if submitted code is built following guidelines

Part of the Continuous Integration / Continuous delivery

Advantages to build tools

- **Automates and simplifies code compilation**
 - Code always compiled the same way
 - Same dependencies always used
 - Reliable!
 - Useful when you have several projects or dependencies between them
- **Plugins are available for IDEs**
 - Maven has built in support for eclipse and netbeans out of the box
 - Plugins can be written for any task
- **All the build information is configured in a single place**
 - Allows version control and standardisation!

4

Reliable – no “it worked on my machine”. No it works on any machine: windows, mac, linux etc.

Repeatable code compiled the same way from same dependencies.

Useful when you have several projects or dependencies

Available Build Managers

- **Make**
 - Based on *makefiles*.
 - Used to build projects on the command line
 - Usually C / C++
- **Ant**
 - Multi-language support – Not used as much these days
- **Maven**
 - Multi-language, but mostly Java
 - Apache's build tool for their projects
 - Based on XML files
- **Gradle**
 - Popular for Java / Scala
 - Built using Groovy
- **SBT**
 - Simple build tool for scala projects
 - Based on scala
 - .sbt files or full scala builds
- **Bamboo**
 - Atlassian's build tool
 - Multi-language support
 - Integration with other atlassian products such as JIRA and bitbucket

5

Wikipedia's List: http://en.wikipedia.org/wiki/List_of_build_automation_software

There are many build tools:

Ant – for multi-language projects

Maven and Gradle – popular for Java projects : do similar jobs but describe it different ways (maven – xml based, Gradle – groovy based)

SBT for Scala

Maven

Accumulator of knowledge – Yiddish language

- **Part of the apache software group**
- **Controls the build using the Project Object Model (POM)**
 - XML file in which you can specify dependencies, build goals and others
- **Create new projects using archetypes**
- **Objectives**
 - Make the build process easy
 - Uniform build system
 - Quality build information
 - Guidelines for best practice
 - Transparent migration of new features



6

Objectives stated on: <https://maven.apache.org/what-is-maven.html> (18/5/2015)

Started as part of apache software group

Apache build very large and complicated projects and it becomes out of hands.

Apache foundation find the way how to describe and to build it. And they started to do this with Java projects.

Now it can be anything

Everything is controlled from the single point – Project Object Model (POM file)

Here you can put in goals, dependencies and source controls management you want.

When you start a new project you can ask Maven to get a skeleton for this. Its all build into one.

Maven Principles

- **Make the build process easy**
 - Developer / Ops don't need to know about how something is being built, just the command to tell maven to do it
- **Uniform build system**
 - Write once, runs the same on all systems
- **Quality build information**
 - Can produce documentation about the code
 - Can create change logs / error logs
- **Guidelines for best practice**
 - Default directory structure keeps languages and source / test parts separate
 - Can link to source control directly to commit changes
- **Transparent migration of new features**
 - Installation of new plugins and repositories is simple and fast

7

- Make the build process easy – in the command line its MVN GOAL (it does not matter what, maven will go and reach a goal).
- Uniform build system – it does not matter where you build your system – it will run the same way on another as well
- Quality build information – designed to give outputs after building some your project - forms, dashboard etc. – says what was happen with your software and is something needs to be changed in terms of testing
- Guidelines for best practice – there are a lot of guidelines of Maven just by lucking basic setup for the project. Source control goes to the src folder and test – to test folder. You can change these folders, but traditionally is build for every one (he can look and knows how it works).
- Transparent migration of new features – you can install new plugins and repositories just by adding something to your pom file.

Why is maven popular

- **Old and stable project**
 - People have been using it for several years
 - Active community – more archetypes being released for new technologies
- **Dependencies managed**
 - Incredibly useful
 - No more downloading jar files to add to projects and realising you have the wrong version number
- **Integration with other tools**
 - Eclipse metadata generator
- **XML is easy enough to understand**
 - But a little verbose

8

A lot of people really don't like maven, but it is still going strong despite this. Gradle works in a similar way to maven but it is not XML based and considered to be more flexible.

Installing Maven

- **Installing on Linux:**

- Maven is included the apt repositories
- On Centos 7 you can use the yum repository

```
$ sudo apt-get install maven
```

- Centos 6 is a little more involved, we will look at it in the exercise!

- **Installing on Windows:**

- Download the zip file
- Unpack to any directory
- Add that directory to the windows path
- Make sure you also have JAVA_HOME on your path

- **Installing on Macs:**

- Install homebrew (package manager for macs)

```
$ brew install maven
```

9

On Centos 6 and Amazon:

Installing Maven

- **Installing on Centos 6 or Amazon web services:**

```
# get the zip file from the website
$ wget http://apache.mirrors.ionfish.org/maven/maven-3\
/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz

# unzip this to /usr/local
$ sudo tar xzf apache-maven-3.3.9-bin.tar.gz -C /usr/local

# create a symbolic link between the program and an
easier to remember directory name
$ sudo ln -s /usr/local/apache-maven-3.3.9 /usr/local/maven

# Finally, link between the mvn binary and the /bin folder
$ sudo ln -s /usr/local/maven/bin/mvn /bin/mvn
```

10

On Centos 6 and Amazon:

Hello Maven!

- **Now we can see if maven is running on our system**

```
$ mvn --version
Apache Maven 3.2.5
Maven home: /usr/local/Cellar/maven/3.2.5/libexec
Java version: 1.8.0_25, vendor: Oracle Corporation
Java home: /Library/Java/JavaVirtualMachines/
                                         jdk1.8.0_25.jdk/Contents/Home/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "10.10.3", arch: "x86_64",
                                         family: "mac"
```

The output will change based on which operating system you have and which version number of the software you have!

Hello Maven!

- **Maven projects are created by using an archetype**
 - There is a hello world style archetype we can use to see this running

```
mvn archetype:generate  
    -DarchetypeArtifactId=maven-archetype-quickstart
```

- This will create a new project in interactive mode
- -D gives a set of options. We can not include any additional information and maven will prompt us to provide it
- Other options:
 - artifactGroupId – the group ID you want to use, usually this is the same as the back package
 - artifactId – the name of your project
 - version – the version number 1.0-SNAPSHOT by default
 - package – the base package to put the main java file in

12

All options:

<https://maven.apache.org/archetype/maven-archetype-plugin/generate-mojo.html>

mvn artifact:generate

mvn compile

The Directory Structure

- **Default for maven projects**

```
my-app
- pom.xml
- src
  - main
    - java
      - com
        - qa
          - myapp
            - App.java
  - test
    - java
      - com
        - qa
          - myapp
            - AppTest.java
```

The POM

```
<project xmlns="...">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.qa</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>my-app</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>
```

14

`<project xmlns="...">` - namespace of your project

`<modelVersion>4.0.0` - this will always be same number!!!!!!

`<url>http://maven.apache.org` - you can give it and it will be shown in metadata

`<dependency>`

includes junit only for tests – so it will be used only in the place where its required (not in all project)

The groupId, artifactId, version and name were all generated from the values we gave maven

The model version is based on the archetype

A list of dependencies tells maven which additional jar files are required. The scope of `<test>` means that it is only included when the project is being tested, rather than when it is packaged up for final release.

Running Tests, Running the project

- **Maven has several built in goals which can be specified**
 - These are described in more detail in the exercise
 - We are interested in four for now
 - Clean – delete all the current compiled code
 - Test – run all the unit tests
 - Looks for tests stored in the test/ directory
 - Package – generate the jar / war files required
 - The compiled files are all stored in target/
 - Site – generate the maven documentation
 - HTML documentation in the /target/site directory

```
$ mvn clean
$ mvn compile
$ mvn test
$ mvn package
$ mvn site
```

15

To run my-app project

```
cd target/classes
```

```
java com.qa.App
```

Or

```
java -cp target/classes com.qa.App
```

The Eclipse Plugin

- To generate all the eclipse metadata run the following command

```
$ mvn eclipse:eclipse
```

- We can also clean out the metadata if required

```
$ mvn eclipse:clean
```

- Projects can be imported directly from the pom by using eclipses' import

Exercise

- **See the exercise guide for more details**
 - Install Maven
 - Pull down some archetypes
 - Test, compile and run your projects

To read more about Maven

Books:

- Apache Maven Cookbook (2015. link: <https://www.packtpub.com/application-development/apache-maven-cookbook>)
- Maven Crash Course (2017. link: <https://www.safaribooksonline.com/library/view/maven-crash-course/9781787124912/>)
- Learning Apache Maven (2015. link: <https://www.safaribooksonline.com/library/view/learning-apache-maven/9781771373661/>)
- Apache Maven Project books (2017. link: <https://maven.apache.org/articles.html>)

Tutorials:

- <https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html>
- <https://www.udemy.com/mavencrashcourse/>
<https://www.tutorialspoint.com/maven/index.htm>

Summary

- **What are build tools**
 - Why are they useful
 - Different build tools available
- **Introducing Maven**
 - What is Maven
 - Why do we like it
 - Maven Principles
- **Installing Maven**
 - Pulling down an archetype
 - Running a project