>CONFESS_2011

CONference For
Enterprise Software
Solutions_

Apache Maven 3

A practical guide

Mark Struberg,
INSO TU Vienna

What is Maven?

- Maven is a build management suite
- Maven is a project of the Apache Software Foundation
  -> http://maven.apache.org
- Maven has a great Community
  -> mailto://users@maven.apache.org
  -> archived at markmail.org , nable.com, ...
  -> IRC: irc.codehaus.org #maven
- A few Wiki pages
  -> https://cwiki.apache.org/MAVEN/maven-3x.html
  -> http://docs.codehaus.org/display/MAVEN/

Agenda - General

- Evolution of build systems
  - make
  - ant
  - maven1
  - maven2
  - maven3
- The daily maven #wtf
- chopping your projects the right way

Agenda – Maven3 basics

- Installing and Configuring Maven
- The default Project structure
- Maven build lifecycle
- Artifacts and Scopes
- The Maven POM explained
- Useful maven plugins

Agenda – Maven3 in depth

- Repository management
- Handling multi module builds
- POM aggregation vs composition
- Building a Site with Maven
- Performing Releases with Maven

Build System Evolution

Maven Evolution - the old days

- in the 80s
- running asm/cc on single files
- who likes to do it this way today? :)

Maven Evolution - make

- automatic 'dirty' file detection
- output-result chain – the first 'lifecycle'
- 'configurable' builds
- 'make clean all' instead of tons of custom parameters

Maven Evolution - Ant

- language independent
- OS independent
- platform independent
- written in Java
- executable XML
- introduces 'tasks'
- ant is 'imperative'

Maven Evolution - Maven1

- no libraries in your project anymore
- dependency management
- jar versioning
- maven is 'declarative'
- convention over configuration – 'best practice' approach
- default build lifecycle
- use plugins – don't repeat yourself!
- providing quality project information

Maven Evolution - Maven2

- plugins mainly written in Java
- support for plugins written in other languages or scripts
- multi module builds
- Container isolation for plugins (ClassLoaders)

Maven Evolution – Maven3 whatsnew

- most changes 'under the hood' (e.g. move to Java5)
- parallel builds
- better classpath separation
- no <reporting> section anymore (still compatible)
- maven1 repo layout support got dropped
- force of reproducable builds
- artifact resolution caching
- improved error reporting and warnings
- https://cwiki.apache.org/MAVEN/maven-3x-compatibility-notes.html

The daily Maven wtf

- There are some complaints about Maven doing things a certain way – but most of the time it's actually the smartest way
  - maven is downloading the Internet
  - XML too verbose
  - builds aren't predictable
  - maven release performs the tests 3 times
  - maven doesn't allow me to do complicated things
  - tbc

  - what are your complaints?

Doing tha Project Sushi

- how to slice your project modules is very important!
- each jar, war, ear, ... always gets an own module
- creating jars is cheap
- adding jars as dependencies is even cheaper
- separate your project into logical parts
- modules can form a tree

Maven 3 – basic concepts

Installing Maven

- download maven from http://maven.apache.org/download.html
- verify checksum with md5sum
- unzip and mv to /opt/apache/
- ln -s /opt/apache/apache-maven-3.0.3 /opt/apache/maven
- ln -s /opt/apache/maven/bin/mvn /usr/bin/mvn
- ln -s /opt/apache/maven/bin/mvnDebug /usr/bin/mvnDebug
- sudo vi /etc/mavenrc
  export MAVEN_OPTS="-client -Xmx512m -Xms128m -XX:MaxPermSize=128m"
  export M2_HOME=/opt/apache/maven

Configure Maven - settings.xml

- The global settings.xml in $M2_HOME/config/settings.xml
  -> proxy settings
  -> global mirror settings
  -> settings for all users
- The local settings.xml in ~/.m2/settings.xml
  -> user specific settings
  -> default profiles
  -> server credentials
- Encrypt your passwords in settings.xml !
  http://maven.apache.org/guides/mini/guide-encryption.html

The Maven CLI

- $> mvn -help
- $> mvn clean install
- $> mvn install:help
- $> mvn sql:execute -Pmigrationdata
- $> mvn test -Dmyfaces.version=2.0.5
- $> mvn deploy -T4

The Maven Lifecycles

- Maven provides 3 default lifecycles: clean, build, site

  http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html
- each project follows this lifecycles
- not all phases are always 'used'
- 'goals' can be bound to phases
- default lifecycle binding for a few packagings (jar, ear, war, pom, ...)

The Build Lifecycle

- validate
- compile
- test
- package
- integration-test
- verify
- install
- deploy

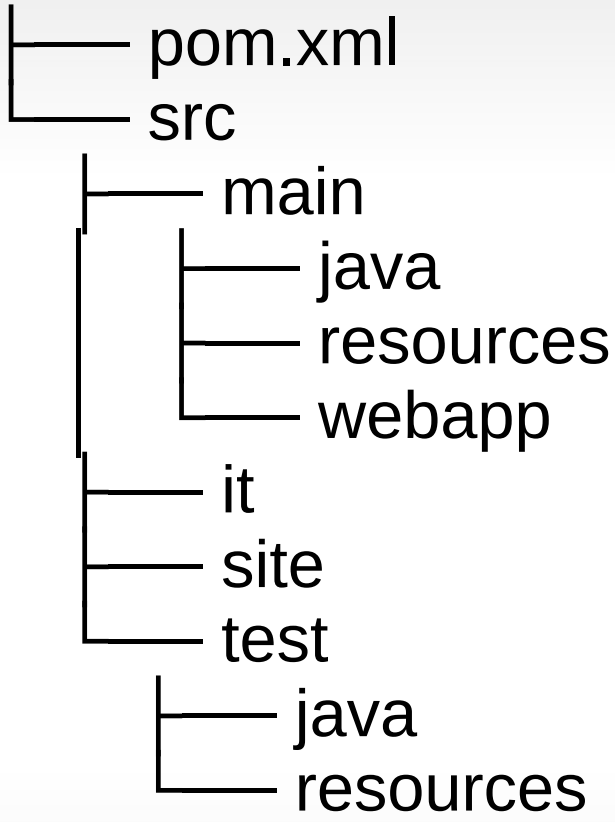
need more Lifecycles?

- usually the default lifecycles are all a user needs
- advanced users might introduce own lifecycles (see cobertura-maven-plugin as example)

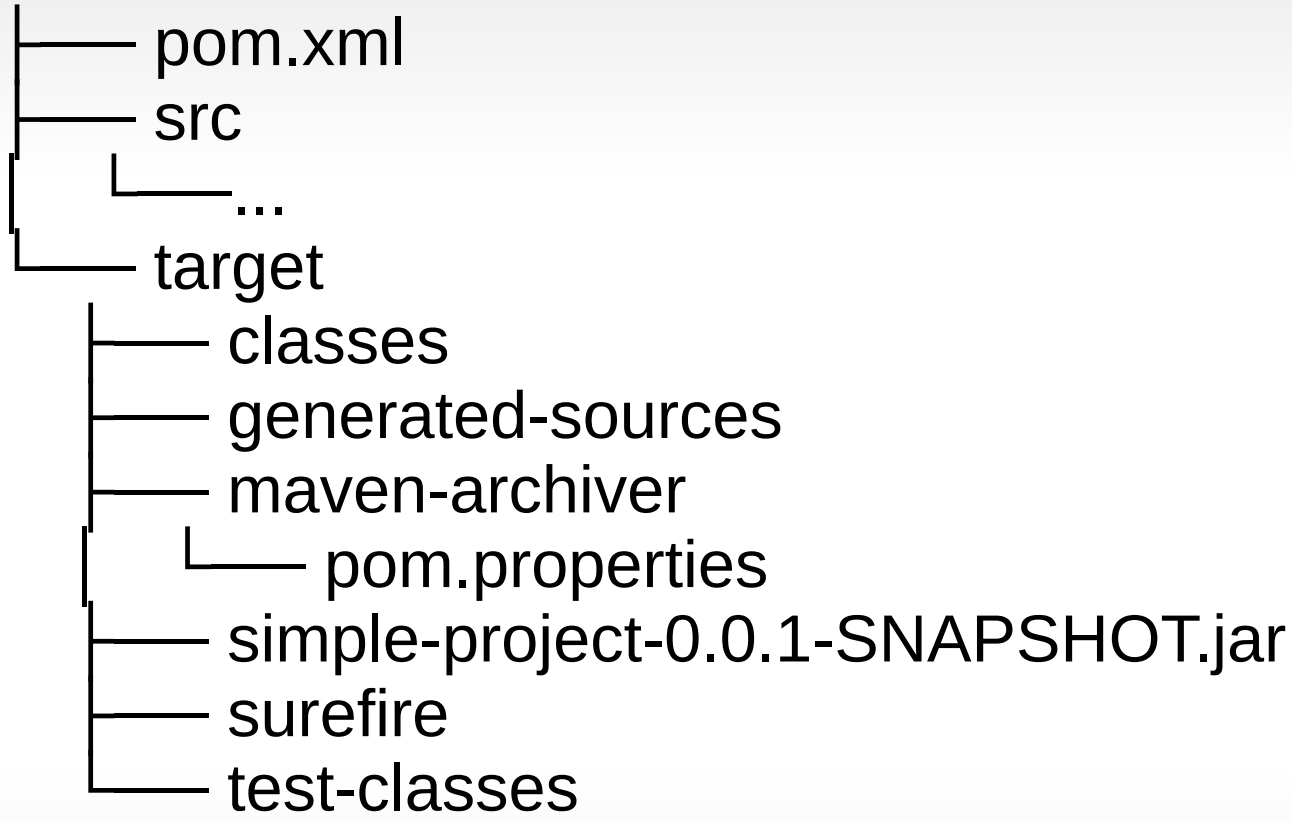
Maven Plugins

- Each Plugin contains multiple 'goals'
- goals can be invoked directly
  $>mvn myplugin:goalA
- goals can be bound to a lifecycle phase via <executions>
- every plugin has a default 'help' goal
  $>mvn myplugin:help
- List of maven3 compatible plugins:
  https://cwiki.apache.org/MAVEN/maven-3x-plugin-compatibility-matrix.html

Default directory layout

- 

Default Build Result Layout

- 

Maven Artifacts

- Artifacts are build results (files) which are stored in a maven repository
- Artifacts are identified by GAV
  <**g**roupId> + <**a**rtifactId> + <**v**ersion>
- Artifacts always contain information in a pom.xml
- maven always have a packaging, default is jar
- 'attached artifacts'

SNAPSHOT Artifacts

- now always get a timstamp in the repo
- <unique> is deprecated
- no release possible with a SNAPSHOT artifact

Repositories

- local repository in ~/.m2/repository
- direct mapping between GAV and directory path
- multiple upstream repositories
- can be switched in settings.xml via <localRepository>
- *lastUpdated
- _maven.repositories

The maven POM

- The GAV of the project
- the packaging of a project
- declare projects infrastructure
- using maven plugins
- dependency resolution

A simple maven project

- -> cd sources/simple_maven_project
- Needed tags
  - groupId
  - artifactId
  - groupId
  - name

Maven Archetypes

- used to create project blueprints
  $> mvn archetype:generate
- easy to create your own archetypes
  $> archetype:create-from-project

- internal stuff:
  META-INF/maven/archetype-metadata.xml
  containing a <archetype-descriptor>
- found via archetype-catalog.xml

POM Inheritance

- <modules>
  for defining child modules
  -> aggregation
  -> 'has a'
  -> only allowed in modules with packaging 'pom'
- <parent>
  for 'extending' another pom
  -> composition
  -> 'is a'

Distribution Management

- <distributionManagement>
  <**repository**>
  <id>mycomp.releases.https</id>
  <name>Release Distribution Repository</name>
  <url>https://mycomp.org/repos/releases</url>
  </repository>
  <**snapshotRepository**>
  <id>mycomp.snapshots.https</id>
  <name>Snapshot Repository</name>
  <url>https://mycomp.org/repos/snapshots</url>
  </snapshotRepository>
  <**site**>
  <id>mycomp.site</id>
  <url>scp://mycomp.org/${project.groupId}/${project.artifactId}/${project.version}</url>
  </site>
  </distributionManagement>

Source Code Management

- the <scm> syntax:
  scm:<scm_provider><delimiter><provider_specific_part>
- example:
  scm:svn:https://myserver.com/svn/myprj/trunk
  scm:git:file:///home/msx/tmp/olamy/scm-git-test/
- attention! some scms need the '/' at the end!
- <scm> contains 3 sub-tags
  - <connection> - will be usedfor read only access
  - <developerConnection> - will be usedfor read-write access
  - <url> - the http browsing URL (fishEye, etc)
- module name gets autom. appended for sub-modules

Using Profiles

- Profiles might contain additional settings which are not always needed
- Can be used to 'switch' between different setups

Plugin Goodies

maven-help-plugin

- useful utility plugin to get informations about your current build
- $> mvn help:effective-pom
- $> mvn help:all-profiles
- $> mvn help:effective-settings

maven-dependency-plugin

- $> mvn dependency:analyze
- $> mvn dependency:list | less
- $> mvn dependency:tree | less
- $> mvn dependency:copy
- $> mvn dependency:copy-dependencies
- $> mvn dependency:go-offline
- $> mvn dependency:sources
- $> mvn dependency:unpack

versions-maven-plugin

- helps with showing upgrade paths to new plugins and dependencies
- $>mvn versions:display-dependency-updates
- $>mvn versions:display-plugin-updates
- $>mvn versions:lock-snapshots
- $>mvn versions:use-releases

maven-assembly-plugin

- used to package (zip, tar.gz) files and provide them as attached artifact
- $> mvn assembly:single

maven-shade-plugin

- can copy packages from other dependencies to the current project
- can be used to 'shade' packages into a private area

maven-checkstyle-plugin

- can enforce checkstyle rules
- e.g.
  - no tabs allowed
  - certain file header (license, company info, etc)
  - certain code style

maven-antrun-plugin

- allows you to embed ant calls into your maven build
- meant for easy transition of ant projects to maven
- better to write plugins (even if written in ant)

Even more plugins needed?

- Mavens own core plugins: maven-xxx-plugin
  http://repo2.maven.org/maven2/org/apache/maven/plugins/
- Codehaus 'Mojo' plugins: xxx-maven-plugin
  http://repo2.maven.org/maven2/org/codehaus/mojo/

Maven 3 - advanced topics

Repository Management

- MANDATORY for Companies!
- -> Apache Archiva
- -> Sonatype Nexus
- "mirroring" by disabling the 'central' repository

Properties

- used to allow a reuse of configuration
- defined as <properties><myProp>myVal</myProp>...
- used in the pom as ${myProp}
- might get overwritten with -DmyProp=otherValue

A Company setup

- a-directional build structure
- modules should be sliced according to real needs

Maven Site

- new maven-site-plugin-3.0-beta-3
- src/site/site.xml defines the main structure
- easy to write sites in apt (Almost Plain Text)
- good resource are the master poms of maven and apache:
  http://repo2.maven.org/maven2/org/apache/maven/maven-parent/19/
  http://repo2.maven.org/maven2/org/apache/apache/9/

Releasing with maven

- $> mvn release:prepare
  -> test
  -> check that all SNAPSHOTS got removed
  -> checks that no modifications are dirty
  -> tags in the SCM
- $> mvn release:perform
  -> does a clean checkout into target/checkout
  -> runs the whole build
  -> tests again
  -> optionally signs the packages
  -> by default builds the site
  -> deploys to the <distributionManagement> locations

Maven Books

- http://maven.apache.org/articles.html
- From http://www.sonatype.com/books.html
  -> Maven by Example
  -> Maven, the complete reference
  -> Maven: The Definitive Guide
- Brett Porter, Maria Odea Ching:
  -> Apache Maven 2 - Effective Implementation
- Nicolas De Loof, Arnaud Héritier
  -> Apache Maven (in french only)

Legal

- Maven is a trademark of the Apache Software Foundation
- Nexus is a trademark of Sonatype
- Images taken from http://www.freephotobank.org, licensed under Creative Commons