

Maven – *or how to automate java builds,
tests and version management with open
source tools*

Erik Putrycz

Software Engineer, Apption Software

erik.putrycz@gmail.com

Outlook

- **What is Maven**
- Maven Concepts and Projects
- Integration and extensibility
- Conclusions
- Demo

Builds?



© Michael Heiss

Today, builds are often...

Build automation, version management, dependency management and testing automation are

- Complex: many tools, inconsistent accross projects, plenty of configuration setting and files, machine dependent paths, etc.
- Build + run the tests + run: “major milestone”

They should

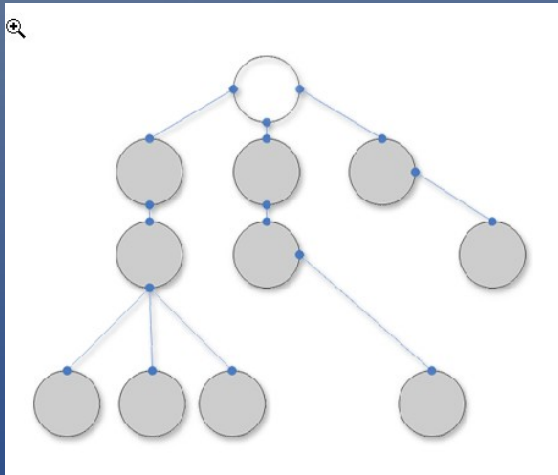
- Not be even worth a talk
- “out of the way” - let the focus on code

What is Maven

A build tool!

```
C:\WINDOWS\system32\cmd.exe
Downloading: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon/1.0-alpha-4/wagon-1.0-alpha-4.pom
3K downloaded
Downloading: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon-provider-api/1.0-alpha-4/wagon-provider-api-1.0-alpha-4.jar
45K downloaded
Downloading: http://repo1.maven.org/maven2/org/apache/maven/maven-artifact-manager/2.0-alpha-3/maven-artifact-manager-2.0-alpha-3.jar
32K downloaded
[INFO] Installing: install
[INFO] Installing C:\my-app\target\my-app-1.0-SNAPSHOT.jar to C:\Documents and Settings\Administrator\TOSHIBA\m2\repository\com\mycompany\app\my-app\1.0-SNAPSHOT\my-app-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 47 seconds
[INFO] Finished at: Fri Jun 24 16:24:10 PDT 2005
[INFO] Final Memory: 2M/5M
[INFO] -----
C:\my-app>
```

A dependency management tool! A documentation tool!



Objectives

- Make the development process visible or transparent
- Provide an easy way to see the health and status of a project
- Decreasing training time for new developers
- Bringing together the tools required in a uniform way
- Preventing inconsistent setups
- Providing a standard development infrastructure across projects
- Focus energy on writing applications

Popular build tools

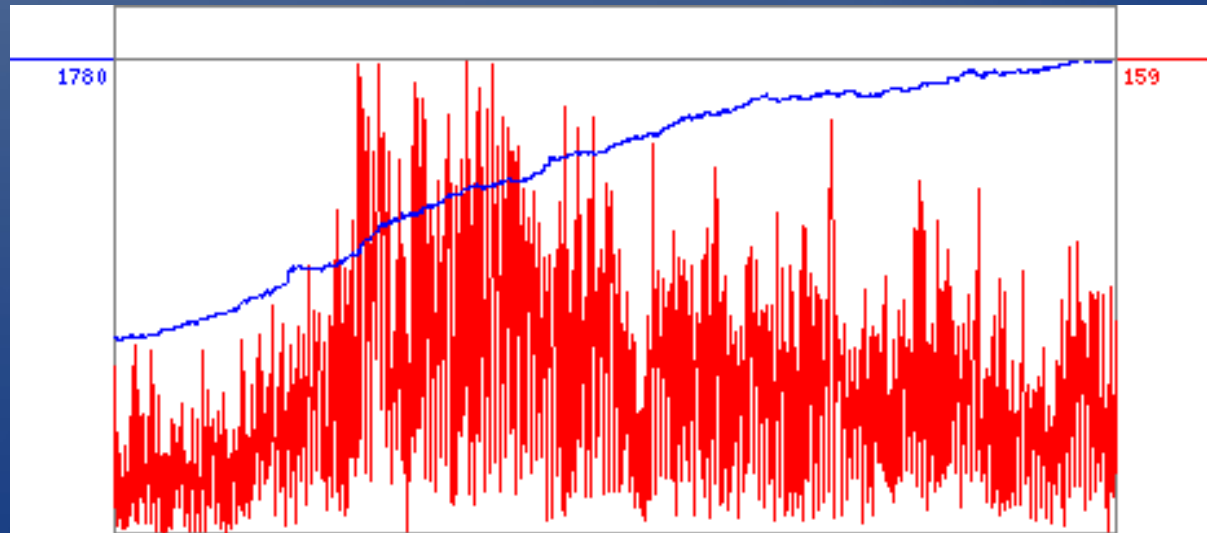
- Ant + Makefile
 - Highly configurable
 - Ant extensible via plugins
 - Script based
- No standard and consistent way for
 - Testing configuration
 - RCS
 - Dependencies
 - Documentation

What is different?

- Declarative based build vs scripted builds
- Reinforces good practices
 - Test automation
 - Integration of tests in the build process
 - Versioning
 - Release management
- Easy integration of build tools
- Consistency for project builds

Is Maven another obscure open source tool?

- 50% of Java projects build with Maven (Oreilly and others)
- 6-8 Million Java developers (Gartner)
- Maven central (dependencies): 58065 artefacts
- `users@maven.apache.org` – over 50 posts a day



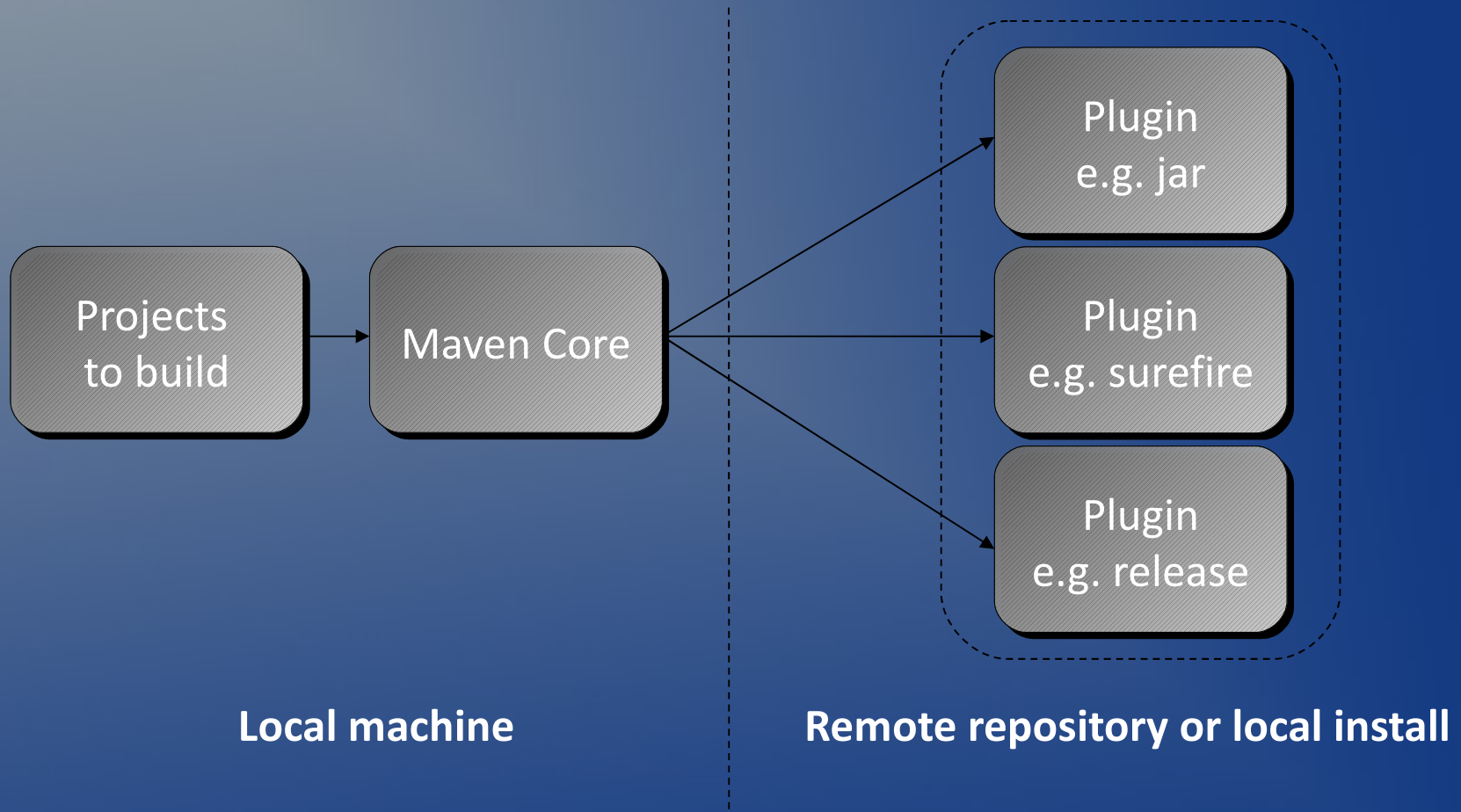
Experience

- Large project @ NRC
 - Integration of many tools+languages+testing
- Converted several open source projects from ant builds to maven
 - Increased the number of unit tests
 - Encouraged developers to write tests
 - Project is part of “maven central” and the ecosystem of dependencies - can be easily reused in other projects

Outlook

- What is Maven
- **Maven Concepts and Projects**
- Integration and extensibility
- Conclusions
- Demo

Maven architecture



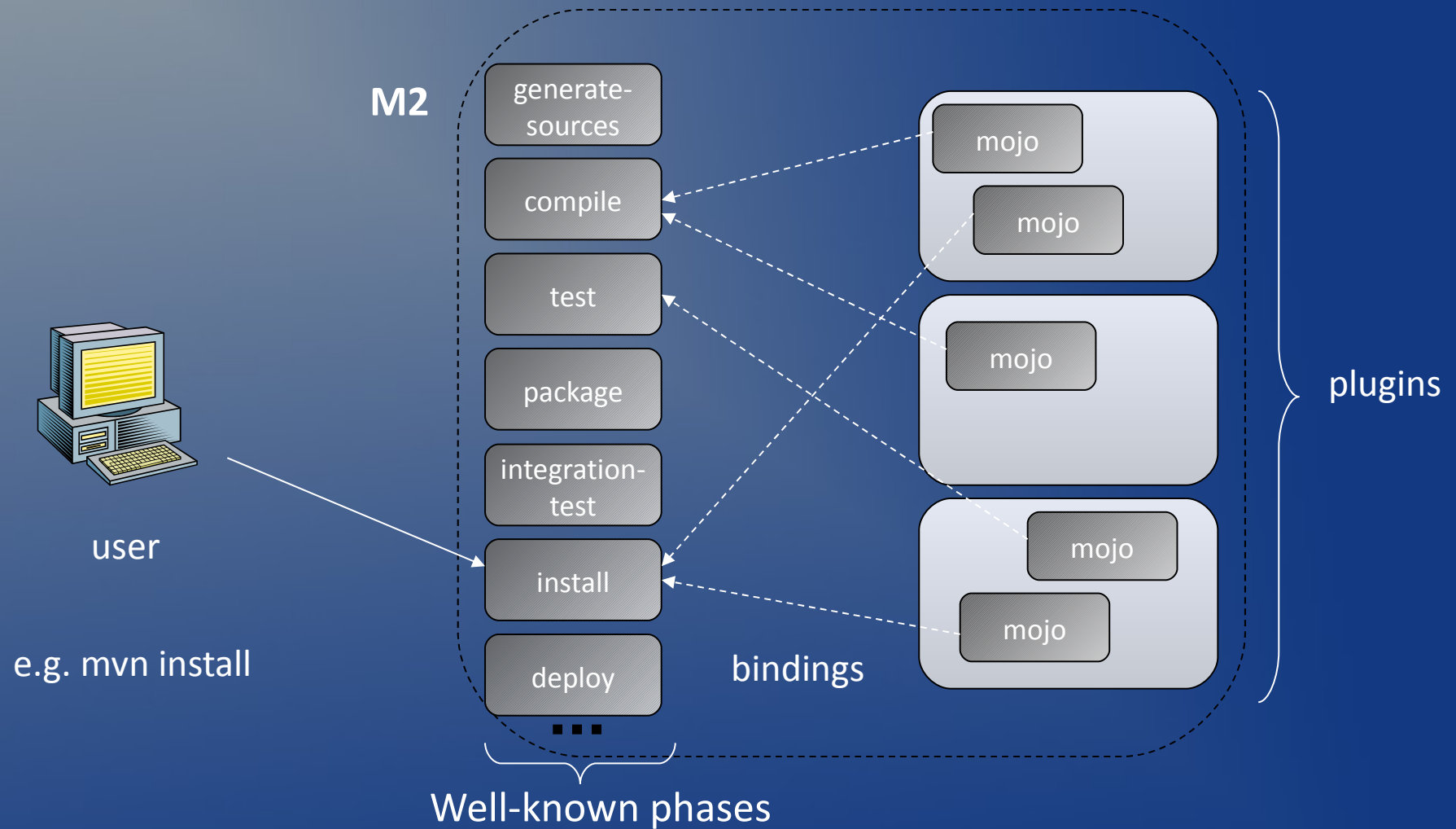
Standard project metadata format

- POM = Project Object Model = pom.xml
- Contains metadata about the project
 - Location of directories, Developers/Contributors, Issue tracking system, Dependencies, Repositories to use, etc
- Example:

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.codehaus.cargo</groupId>
  <artifactId>cargo-core-api-container</artifactId>
  <name>Cargo Core Container API</name>
  <version>0.7-SNAPSHOT</version>
  <packaging>jar</packaging>
  -----
  <dependencies/>
  <build/>
  [...]
```

↑ Minimal POM

Standard way to build applications (1/4)



Standard way to build applications (2/4)

- Default lifecycle
 - *validate* - validate the project is correct and all necessary information is available
 - *compile* - compile the source code of the project
 - *test* - test the compiled source code using a suitable unit testing framework.
 - *package* - take the compiled code and package it in its distributable format, such as a JAR.

Standard way to build applications (3/4)

- *integration-test* - process and deploy the package if necessary into an environment where integration tests can be run
- *verify* - run any checks to verify the package is valid and meets quality criteria
- *install* - install the package into the local repository, for use as a dependency in other projects locally
- *deploy* - done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

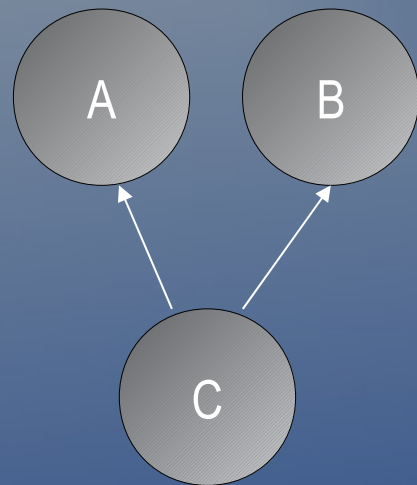
Standard way to build applications (4/4)

- The lifecycle depends on the project type (packaging)
 - Defined in pom.xml (pom, jar, ear, war, etc)
 - Ex: `<packaging>jar</packaging>`
- User can modify lifecycles by adding a goal to a phase:

```
<plugin>
  <groupId>com.mycompany.example</groupId>
  <artifactId>touch-maven-plugin</artifactId>
  <executions>
    <execution>
      <phase>process-test-resources</phase>
      <configuration>[...]</configuration>
      <goals>
        <goal>timestamp</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

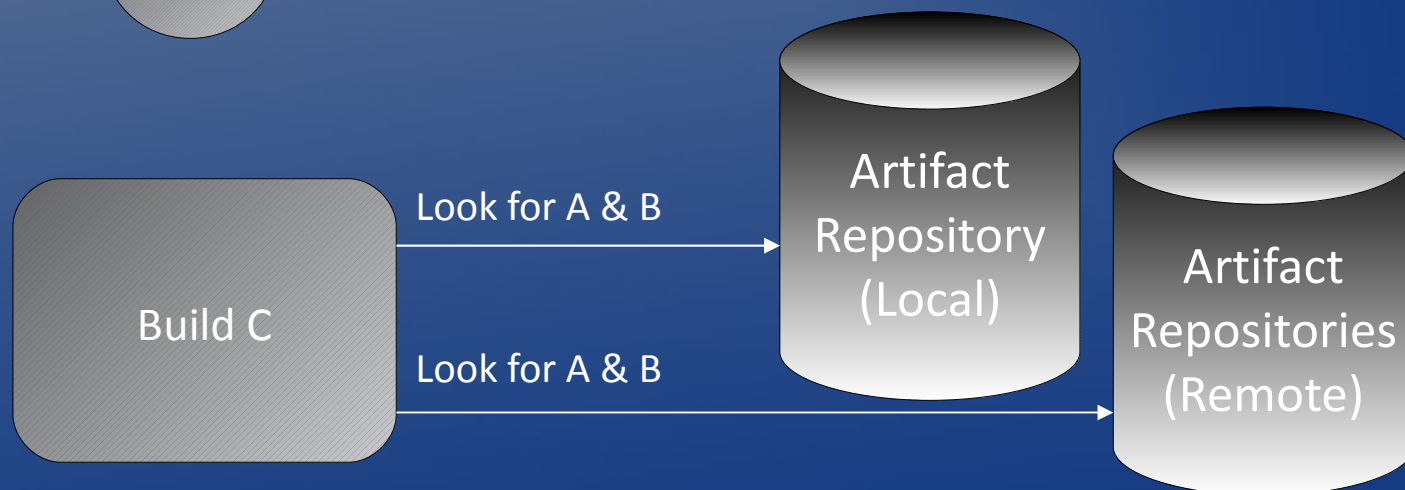
Dependency management (1/2)

- Maven uses binary dependencies



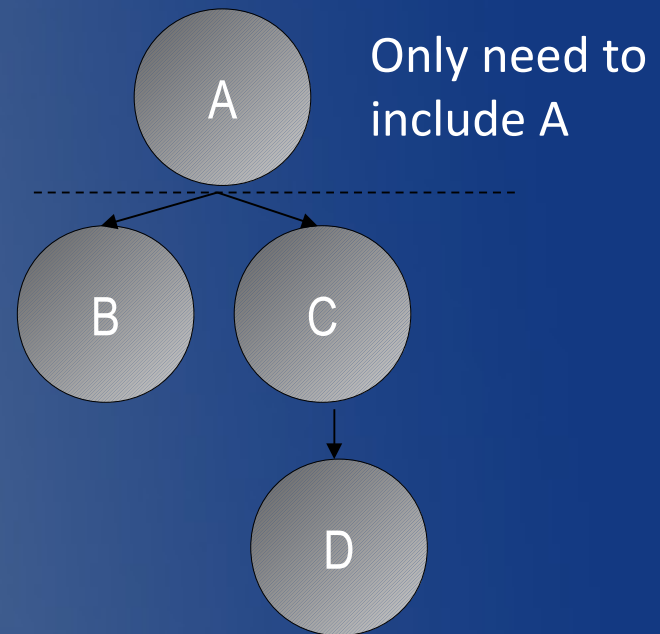
```
<dependencies>
  <dependency>
    <groupId>com.acme</groupId>
    <artifactId>B</artifactId>
    <version>[1.0,)</version>
    <scope>compile</scope>
  </dependency>
</dependencies>
```

« any version after 1.0 »



Dependency management (2/2)

- Transitive dependencies
 - Possibility to exclude some deps
 - Need good metadata
 - Ideally projects should be split
- SNAPSHOT handling
 - Always get latest
- Automatic dep updates
 - By default every day



Versions in Maven

- SNAPSHOT
 - Development version
 - New version availability is checked at every build and downloaded if necessary
 - Snapshot dependencies not allowed for releases
- Fixed version
 - Downloaded once to the local repository

Making a release

```
$ mvn release:prepare [-DdryRun=true]
```

Checks SCM for modifications

Checks for use of SNAPSHOT dependencies or plugins

Runs `$ mvn clean integration-test`

Requests release info:

version numbers

Creates new POMs:

`pom.xml` for tag

`pom.xml` for next version

`release-pom.xml`

Creates tag in SCM

```
$ mvn release:perform
```

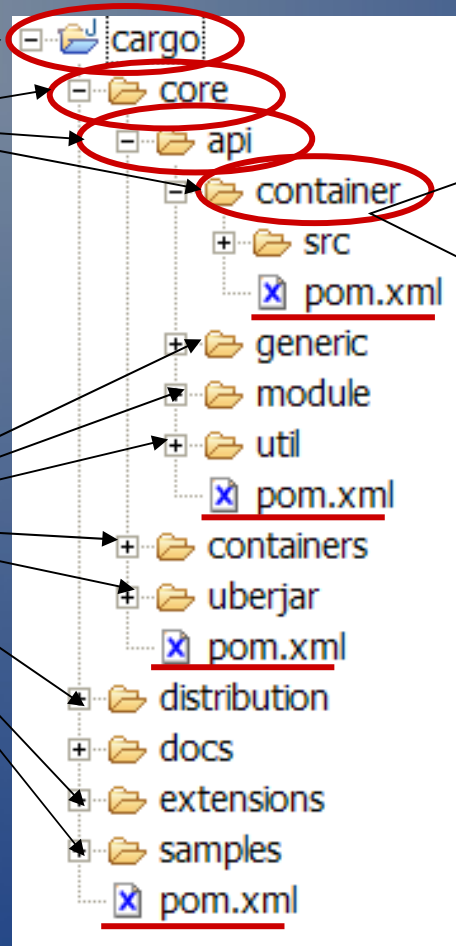
Uses `release-pom.xml`, deploys

project, generates site, etc.

Standard directory organization

4 nested projects

Other projects

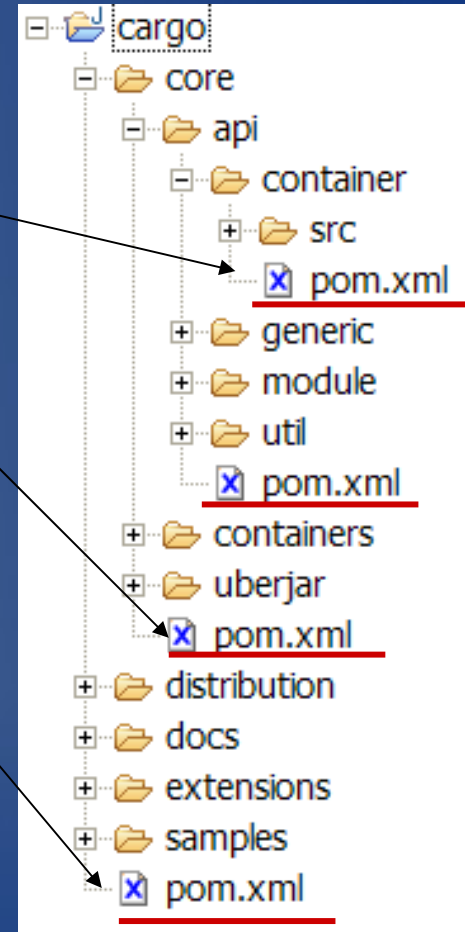


- src/
 - main/
 - java/
 - resources/
 - webapp/
 - application/
 - groovy/
 - test/
 - java/
 - resources/
 - cactus/
 - site/

Multi-module builds

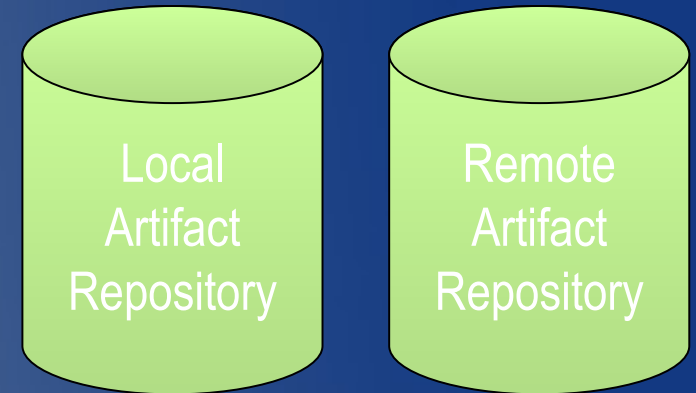
- Projects and subprojects
- Run « mvn » at parent level
 - E.g. `mvn install` in `cargo/core/api`
 - E.g. `mvn install` in `cargo/core`
 - E.g. `mvn install` in `cargo/`
- Declare children projects in parents:

```
<modules>  
  <module>core</module>  
  <module>extensions</module>  
  <module>samples</module>  
</modules>
```



Artifact repositories (1/2)

- Used to store all kind of artifacts
 - JARs, EARs, WARs, NBMs, EJBs, ZIPs, plugins, ...
- All project interactions go through the repository
 - No more relative paths!
 - Easy to share between teams

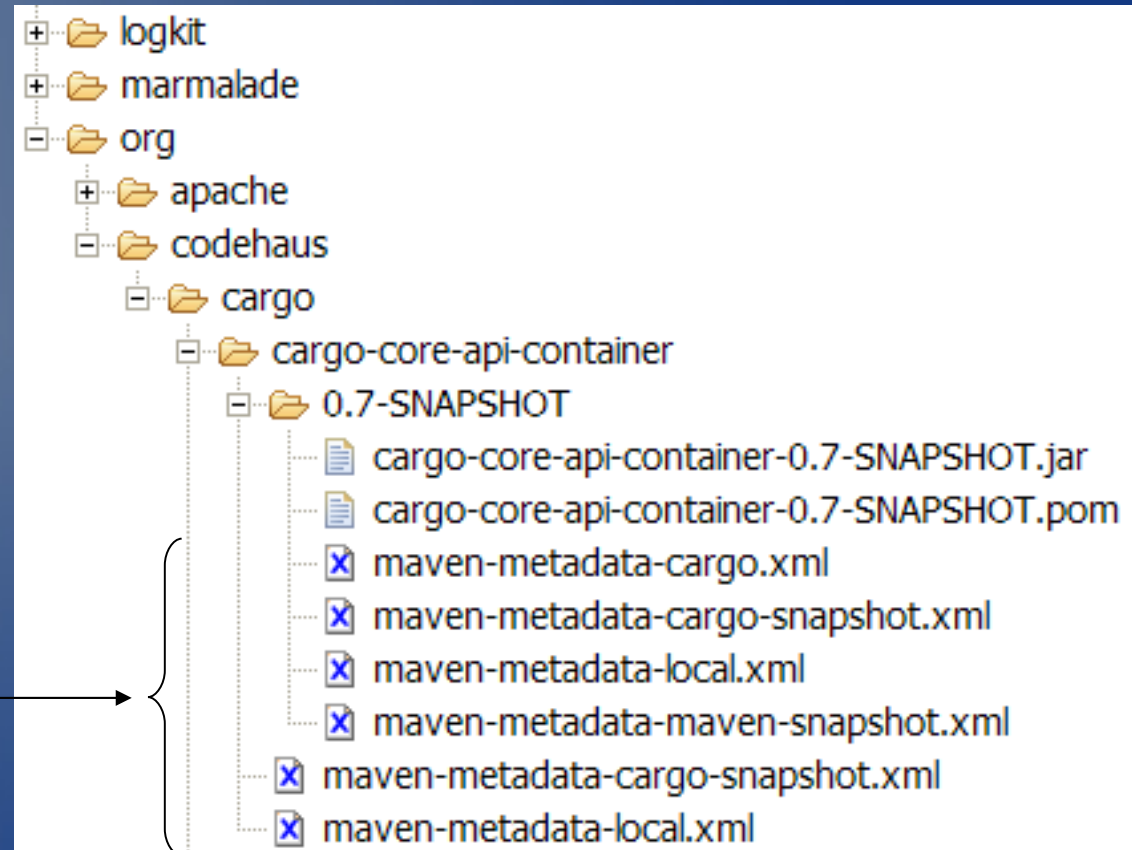


e.g. <http://ibiblio.org/maven2>

```
<repositories>
  <repository>
    <id>maven2-snapshot</id>
    <releases>
      <enabled>>true</enabled>
    </releases>
    <name>Maven Central Development Repository</name>
    <url>http://snapshots.maven.codehaus.org/maven2</url>
    <layout>legacy|default</layout>
  </repository>
</repositories>
```


Artifact repositories (2/2)

- Hierarchical structure
- Automatic plugin download
- Plugins are read directly from the repository
- Configurable strategies for checking the remote repositories for updates
 - Daily check by default for plugin and ranges updates
- Remote repositories contain Metadata information
 - Releases, latest, and more to come



Private Repositories

- Everything is not open source (?!)
- Private repository
 - Directory with http access and file copy through any protocol (ssh/scp/ftp/smb/file...)
- Repository managers
 - 3 main projects: Archiva, Nexus, Artifactory
 - Proxy
 - Search and management

Archiva

Maven Archiva :: Browse Repository - Mozilla Firefox

File Edit View History del.icio.us Bookmarks Tools Help GBookmarks http: va pro:

Find


- Search
- Find Artifact
- Browse

Manage

- Reports
- User Management
- Appearance

Administration

- Settings
- Managed Repositories
- Proxied Repositories

 **Hibernate**

Info Dependencies Dependency Tree Used By Mailing Lists Reports

[top] / [org](#) / [hibernate](#) / [hibernate](#) / **3.2.2.ga**

Relational Persistence for Java

Group ID org.hibernate
Artifact ID hibernate
Version 3.2.2.ga
Packaging jar

POM Dependency Snippet

```
<dependency>  
  <groupId>org.hibernate</groupId>  
  <artifactId>hibernate</artifactId>  
  <version>3.2.2.ga</version>  
</dependency>
```

Other Details

URL <http://www.hibernate.org>
Organisation [JBoss Inc.](#)
License [GNU LESSER GENERAL PUBLIC LICENSE](#)
Issue Tracker [Jira](#)

SCM

Connection scm:svn:http://anonhibernate.labs.jboss.com/trunk/Hibernate3
Dev. Connection scm:svn:https://hibernate.labs.jboss.com/repos/hibernate
Viewer <http://cvs.sourceforge.net/viewcvs.py/hibernate/Hibernate3/>

Done

Environment-dependent builds (1/2)

- Based on profiles
 - Located in pom.xml, in profiles.xml or in settings.xml

```
<profiles>
  <profile>
    <id>tomcat5x</id>
    <activation>
      <activeByDefault>>true</activeByDefault>
    </activation>
    <properties>
      <containerId>tomcat5x</containerId>
      <downloadUrl>..jakarta-tomcat-5.0.30.zip</downloadUrl>
    </properties>
  </profile>
  <profile>
    <id>orion2x</id>
    <properties>
      <containerId>orion2x</containerId>
      <downloadUrl>..orion2.0.5.zip</downloadUrl>
    </properties>
  </profile>
</profiles>
```

Profile that is always active

[...]

Environment-dependent builds (2/2)

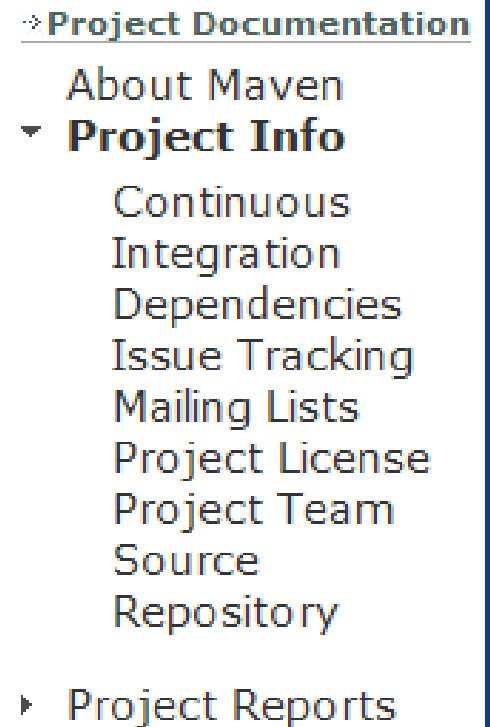
- Different activation conditions
 - JDK version, OS, property defined, existence of file or directory
- Profiles can also modify plugin configurations and other POM elements
 - Merged with the main pom.xml content
- Profiles can be selected on the command line:

```
mvn -P orion2x,resin3x install
```

Site and reports (1/4)

- Lots of reports
 - Project information (mailing lists, SCM, dependencies, etc)
 - PMD, Checkstyle, Javadoc, etc

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-javadoc-plugin</artifactId>
    </plugin>
    [...]
  </plugins>
</reporting>
```



❖ **Project Documentation**

- About Maven
- ▼ **Project Info**
 - Continuous Integration
 - Dependencies
 - Issue Tracking
 - Mailing Lists
 - Project License
 - Project Team
 - Source Repository
- ▶ Project Reports

Site and reports (2/4)

- Accepts several input formats
 - Almost Plain Text (Wiki like)
 - Xdoc (Maven 1.0 compatible)
 - FAQ (Maven 1.0 compatible)
 - Docbook

```
mvn site
```

```
+-- src/  
  +- site/  
    +- apt/  
      +- index.apt  
    +- xdoc/  
      +- other.xml  
    +- fml/  
      +- general.fml  
      +- faq.fml  
    +- site.xml
```

Site and reports (3/4)

```
-----  
The APT format  
-----  
Jason van Zyl  
-----  
2009-03-02  
-----
```

The APT format

~~~~~

APT stands for "Almost Plain Text". APT is a markup language that takes the hassle out of documentation writing by striving for simplicity. Its syntax resembles plain-text more than it resembles most other markup formats (such as HTML).

This document provides some examples of available APT formatting.

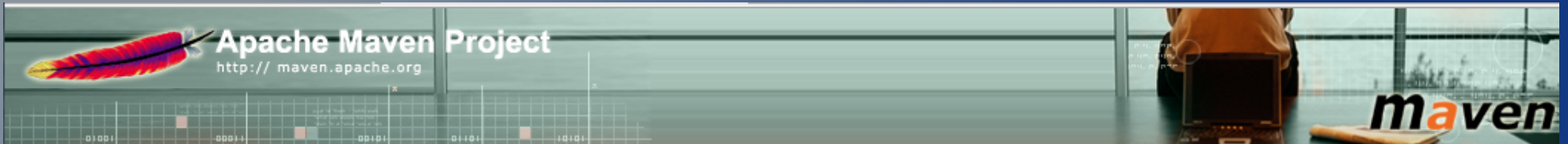
\* Important Notice

The information contained in this document corresponds to the original APT format as published by {{{http://www.xmlmind.com/}Xmlmind}}.

...



# Site and reports (4/4)



Apache > Doxia Site Last Published: 2009-05-12

→ **About Doxia**

- What is Doxia?  
Overview
- ▶ What's new in 1.1?  
Downloads  
FAQ

→ **Documentation**

- ▼ Format References
  - ▼ **Apt Format**
    - Doxia Apt Enhancements
  
    - FML Format
    - Xdoc Format
  
- Doxia Modules Guide
- Doxia Macros Guide
- Writing Books
- Issues & Gotchas
- External Resources

→ **Developer Docs**

- Developer Centre
- Doxia 1.1.x
- Doxia Sitetools 1.1.x
- Doxia 1.0.x
- Doxia Sitetools 1.0.x
- Doxia Tools

→ **Project Documentation**

- ▶ Project Information

## The APT format

APT stands for "Almost Plain Text". APT is a markup language that takes the hassle out of documentation writing by striving for simplicity. Its syntax resembles plain-text more than it resembles most other markup formats (such as HTML).

This document provides some examples of available APT formatting.

→ **Important Notice**

The information contained in this document corresponds to the original APT format as published by [Xmlmind](#). In version 1.1 Maven Doxia has applied several modifications to this original format, see this separate [document](#) for a detailed description. Notable differences are highlighted below with a [\[Change\]](#) link.

The following sections contain formatted text that demonstrates the use of APT to create paragraphs, headers, sections, lists, code samples, figures, tables, rules, breaks, and text level elements such as font styles, anchors, and special characters. Boxes containing text in typewriter-like font are examples of APT source.

→ **Document structure**

A short APT document is contained in a single text file. A longer document may be contained in a ordered list of text files. For instance, first text file contains section 1, second text file contains section 2, and so on.

→ **Note:**

Splitting the APT document in several text files on a section boundary is not mandatory. The split may occur anywhere. However doing so is recommended because a text file containing a section is by itself a valid APT document.

# Outlook

- What is Maven
- Maven Concepts and Projects
- **Integration and extensibility**
- Conclusions
- Demo

# IDE Integration

- Netbeans + IDEA
  - Built-in
  - Netbeans delegates major actions to maven (test, compile, run)
- Eclipse
  - 2 main plugins: IAM and M2Eclipse
  - Manage classpath and source folders

# Plugins

- “Official” plugins
  - Core (compiler, tests)
  - Packaging (jar, war, etc.)
  - Reporting (Documentation, tests, etc.)
  - Tools (ant, version management, etc.)
  - IDE (Eclipse, IDEA, Netbeans)
- Many other plugins available on third party repositories

# SCM Integration

- Fully implemented:
  - Bazaar, CVS, Mercurial, Perforce, StarTeam, Subversion, CM Synergy
- Partially implemented:
  - ClearCase, File system, Visual Source Safe
- Release process creates tags for each version and puts the new development version in the main branch
- Change plugin – documents changes between versions

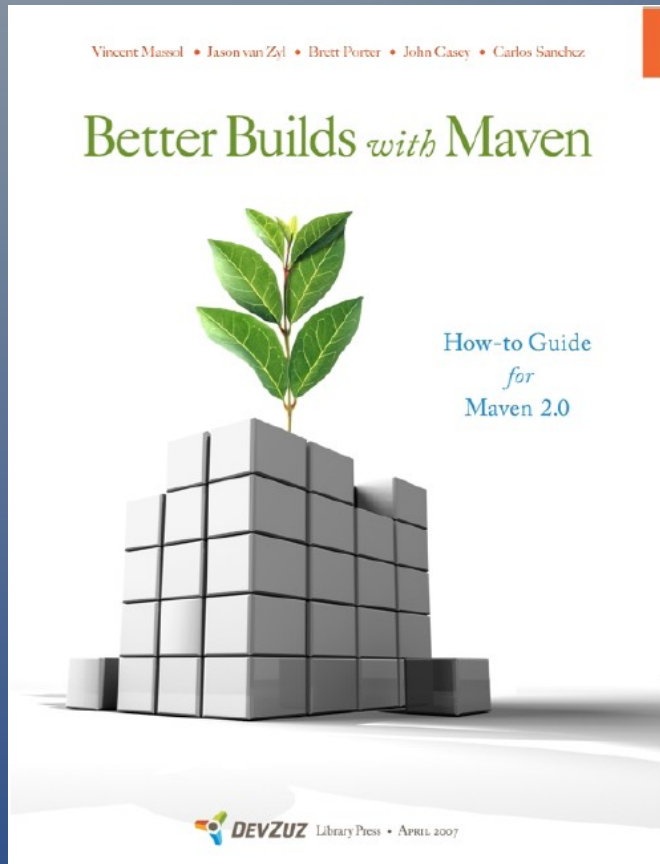
# Outlook

- What is Maven
- Maven Concepts and Projects
- Integration and extensibility
- **Conclusions**
- Demo

# Current status and future

- Maven 2.0
  - Most important milestone – better dependency management
- Maven 2.1
  - Several issues about the plugin versions solved
    - Fixed version of plugins by default
- Maven 3.0
  - Refactoring of the code for easier understanding and better extensibility
  - Complete backward compatibility

# Free Maven Books



**Better Builds with Maven**  
[www.mergere.com](http://www.mergere.com)

**Maven: The Definitive Guide (alpha)**  
[www.sonatype.com/book](http://www.sonatype.com/book)

Sonatype | Build Success for your Enterprise

home training partners news & events company contact blog

### Everybody Builds

It's true. Everybody builds. Building your software shouldn't be painful and it doesn't have to be. We are experts, with decades of combined experience, who specialize in setting up sound development infrastructures that yield results. Let us help your team be more productive and consistent.



### Joyful Metamorphosis

All IT departments are acutely aware of the liability of build script crud, accumulated over time. Learn about "inner platform effect" and how Sonatype strategies helped migrate thousands of lines of legacy code over to Apache Maven, a healthy, maintainable and extensible infrastructure for build management. [Read more ...](#)

### Outstanding Training

We provide training for Maven related technologies with courses for managers, users, teams, and Maven trainers.

We provide public training sessions in addition to onsite sessions that can be customized to your projects and development practices.

[▶ Learn more ...](#)

### Maven: The Definitive Guide

We have produced a free Maven guide for users that is updated and released on a weekly basis.

We are working with the founder of the Maven project, and core Maven committers to produce an unrivaled technical resource for Maven users.

[▶ Read the book ...](#)

### Our Partners Network

We see ourselves as facilitators and enablers of Maven related technologies. We pride ourselves on creating a healthy network of training, service, support, and product partners:



©2007 Sonatype, Inc. [info@sonatype.com](mailto:info@sonatype.com)



# Conclusions

- Significant learning curve
  - Different paradigms from ant or Makefile
- Promotes better reuse through a large community
  - No more libraries on a fixed path
- Lowers effort for infrastructure setup
  - Test runner + reports + builds
- Consistency accross projects
  - Same testing configuration

# Outlook

- What is Maven
- Maven Concepts and Projects
- Integration and extensibility
- Conclusions
- **Demo**

# Questions ?

Some slides © Vincent Massol 2005