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

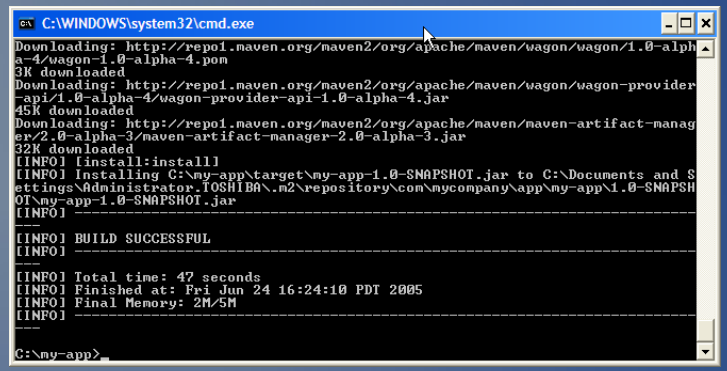
- Complex: many tools, inconsistent accross projects, plenty of configuration setting and files, machine dependent paths, etc.
- Build + run the tests + run: “major milestone”

They should

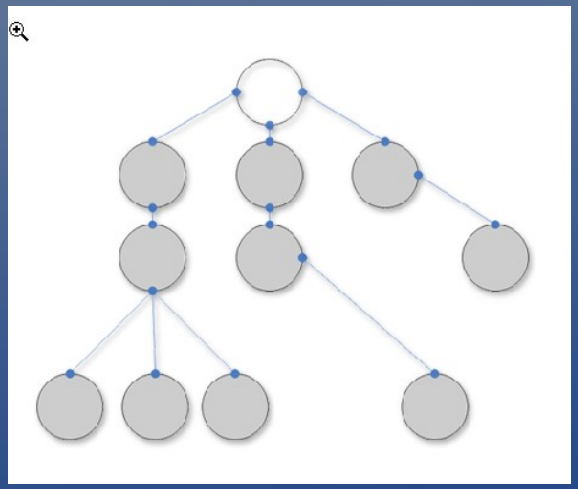
- Not be even worth a talk
- “out of the way” - let the focus on code

What is Maven

A build tool!



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

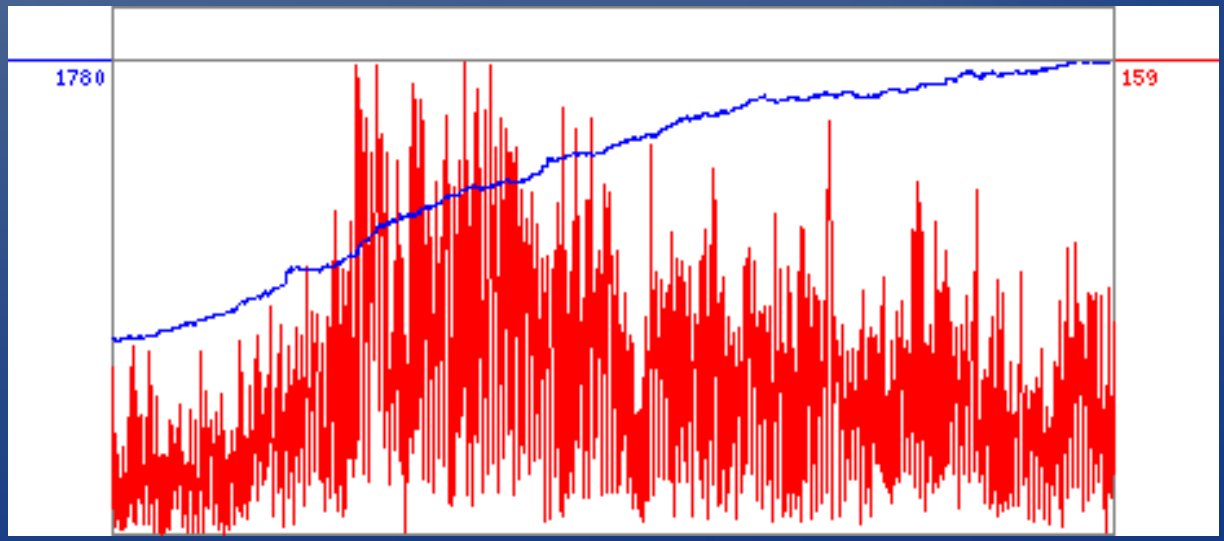
- Ant + Makefile
  - Highly configurable
  - Ant extensible via plugins
  - Script based
- No standard and consistent way for
  - Testing configuration
  - RCS
  - Dependencies
  - Documentation

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



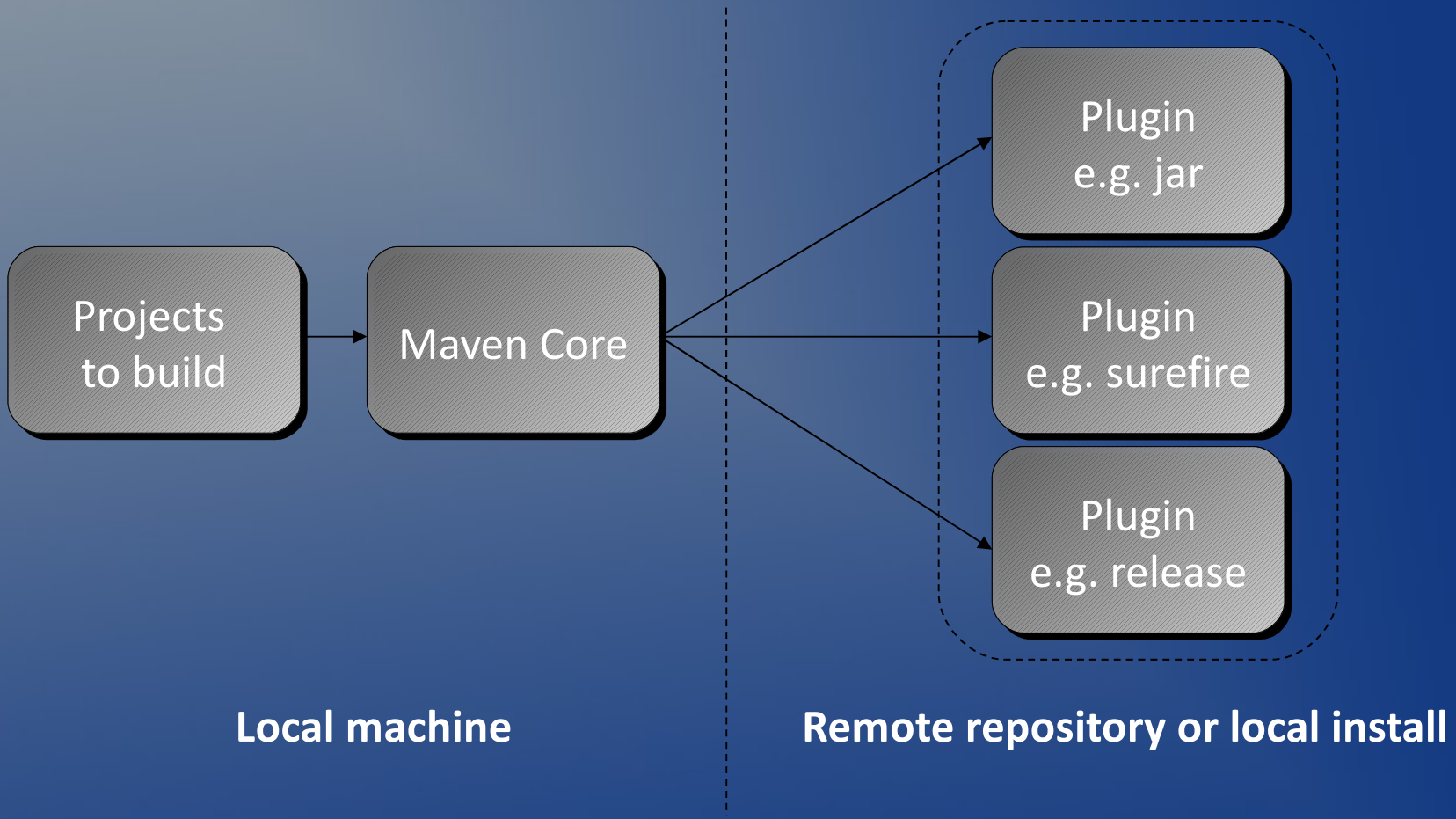
Experience

- 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



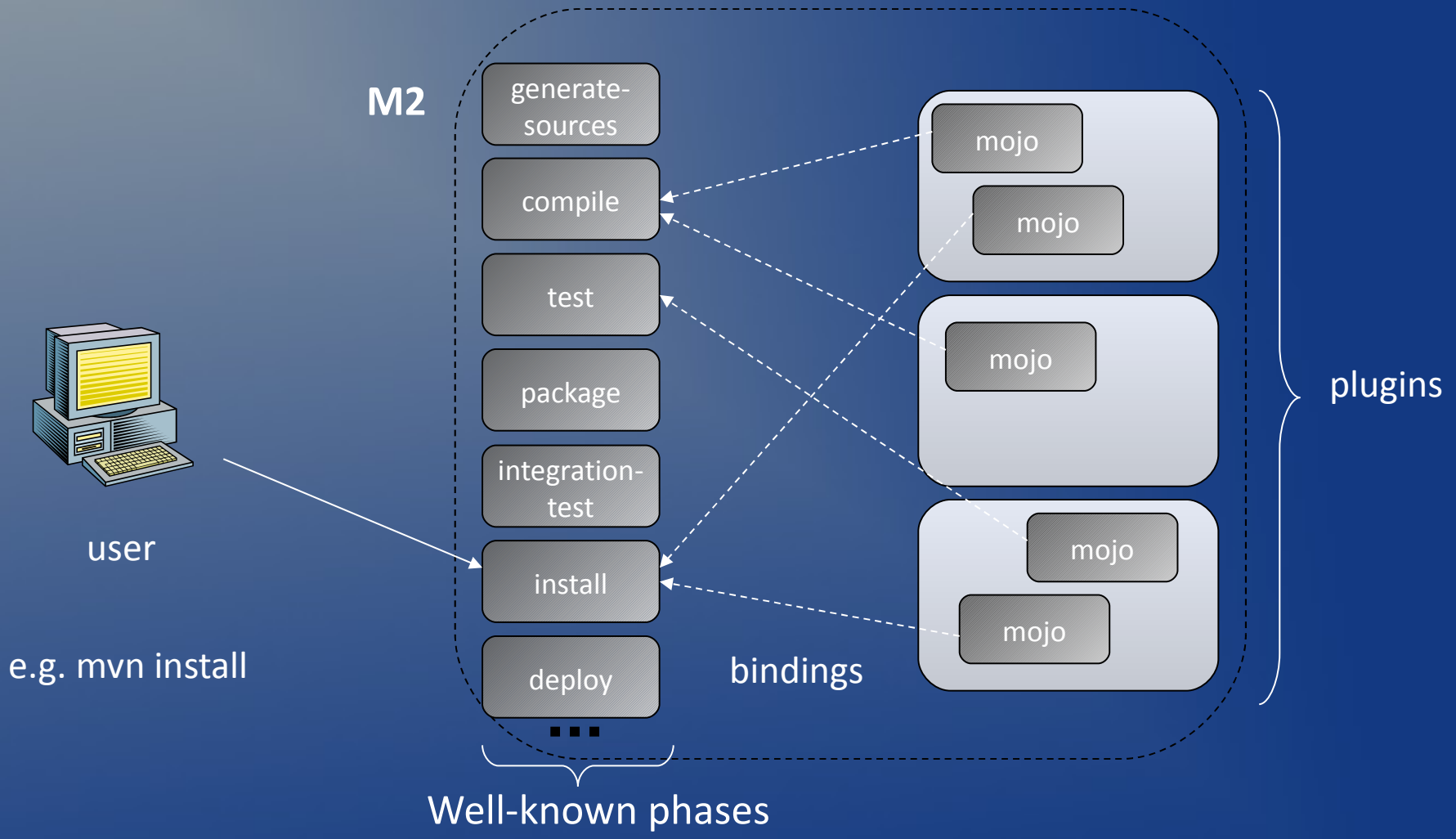
Standard project metadata format

- 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.

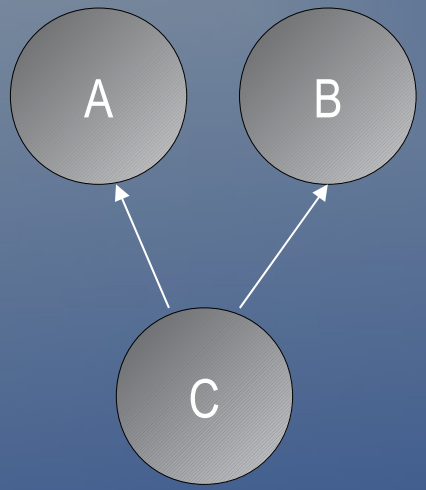
Standard way to build applications (4/4)

- 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>
```

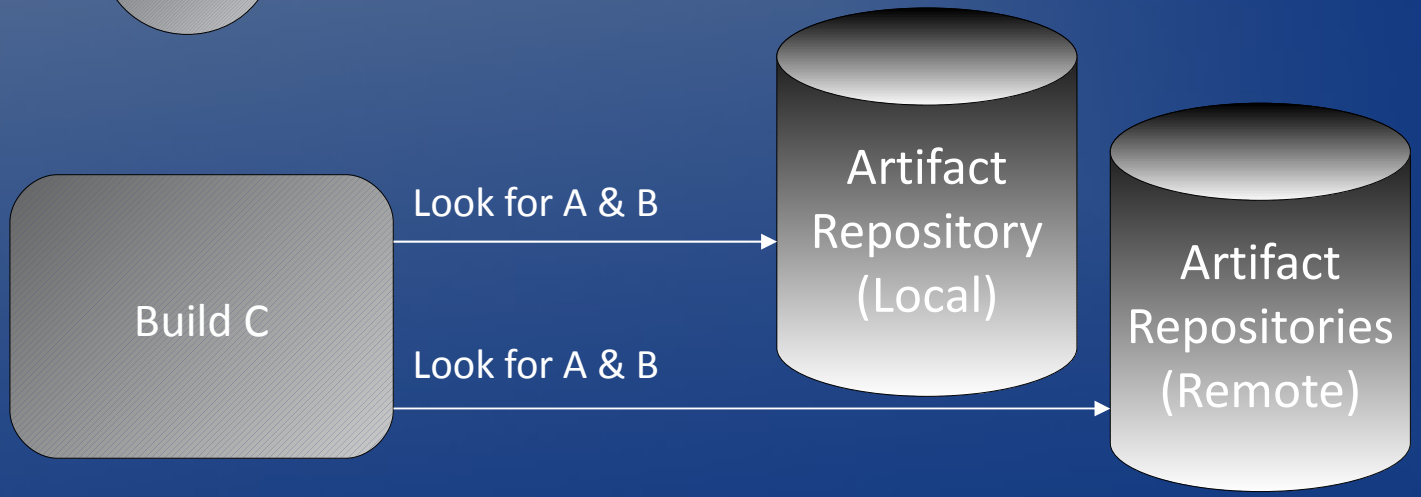
Dependency management (1/2)

- Maven uses binary dependencies



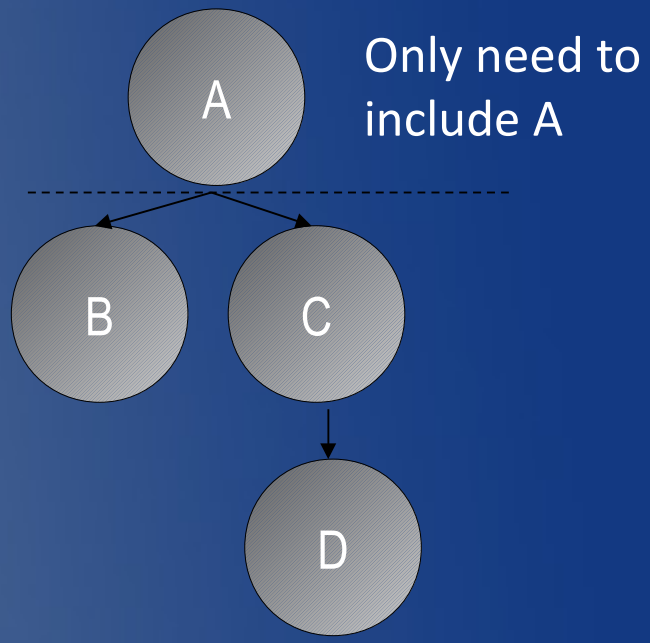
« any version after 1.0 »

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



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

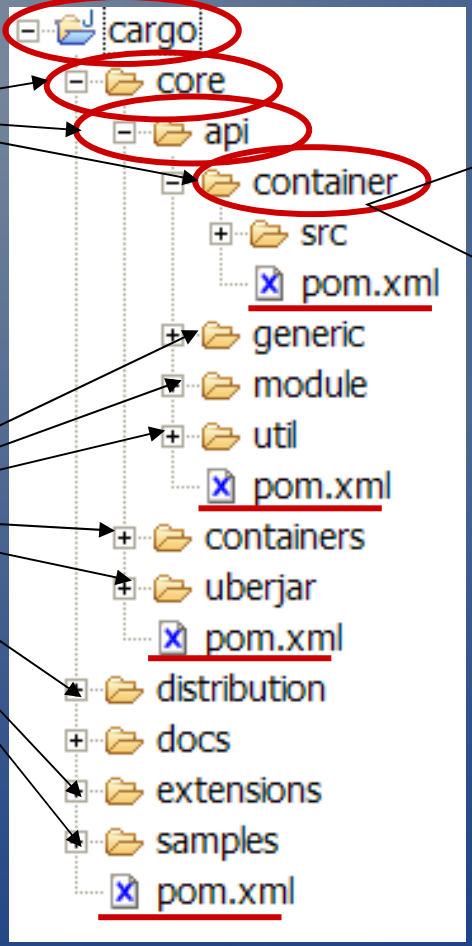
Making a release

$ 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

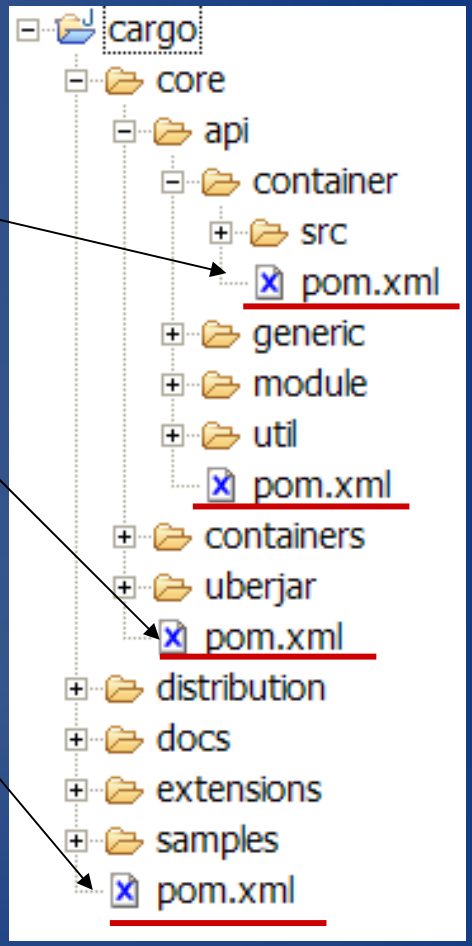


- src/
  - main/
    - java/
    - resources/
    - webapp/
    - application/
    - groovy/
  - test/
    - java/
    - resources/
    - cactus/
  - site/

Multi-module builds

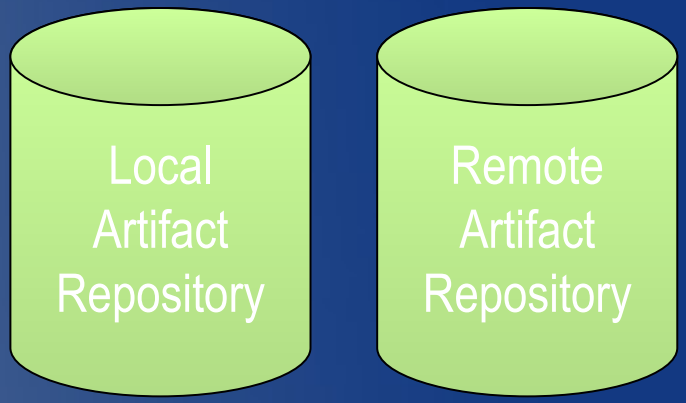
- Projects and subprojects
- Run « `mvn` » at parent level
  - E.g. **`mvn install`** in `cargo/core/api`
  - E.g. **`mvn install`** in `cargo/core`
  - E.g. **`mvn install`** in `cargo/`
- Declare children projects in parents:

```
<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

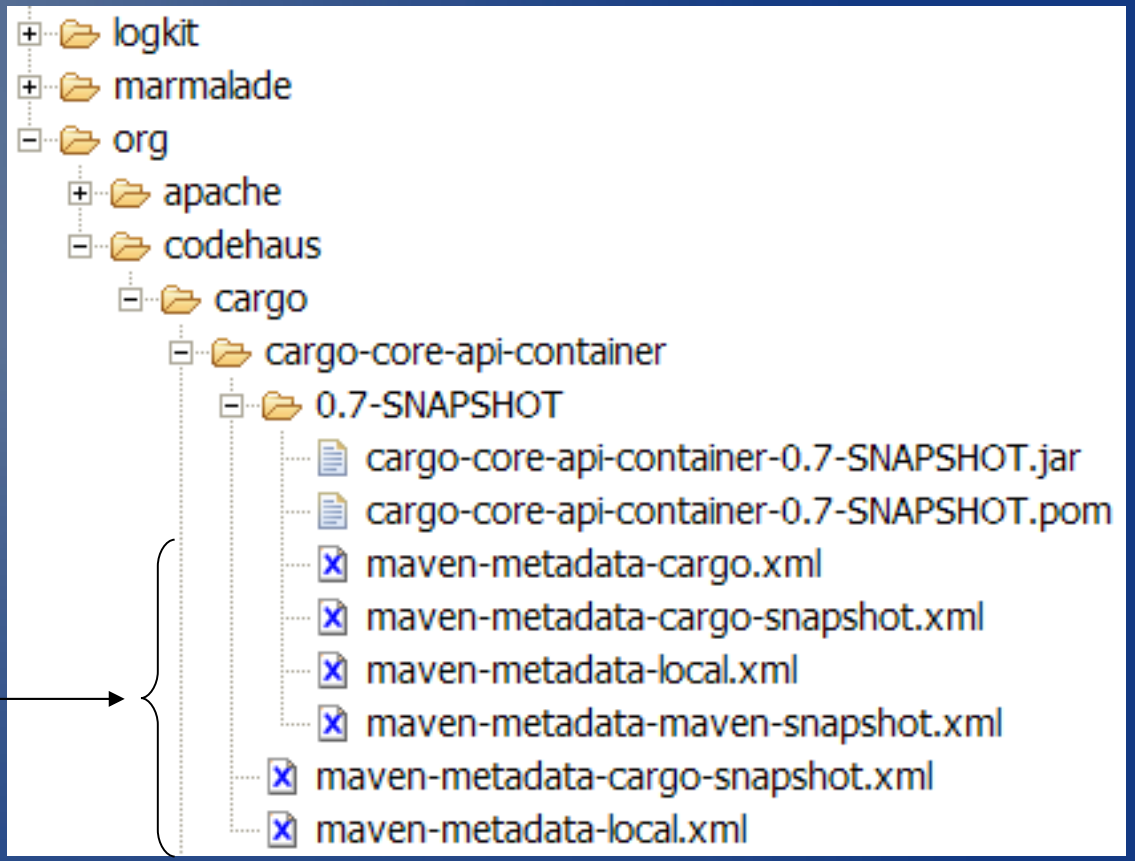


e.g. http://ibiblio.org/maven2

```
<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)

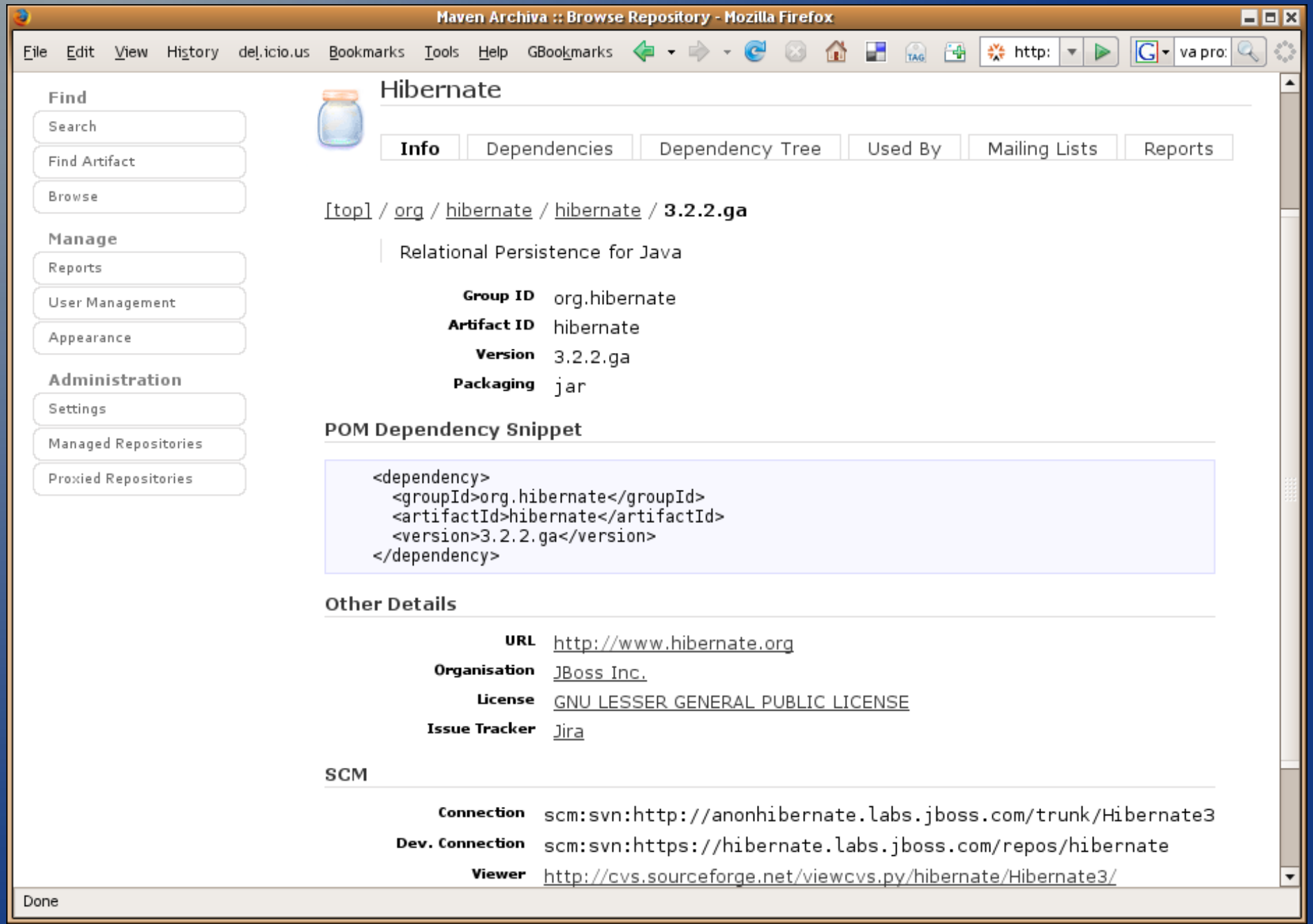
- Hierarchical structure
- Automatic plugin download
- Plugins are read directly from the repository
- Configurable strategies for checking the remote repositories for updates
  - Daily check by default for plugin and ranges updates
- Remote repositories contain Metadata information
  - Releases, latest,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

Archiva


Maven Archiva :: Browse Repository - Mozilla Firefox
File Edit View History del.icio.us Bookmarks Tools Help GBookmarks
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>
[…]
```

Profile that is always active

Environment-dependent builds (2/2)

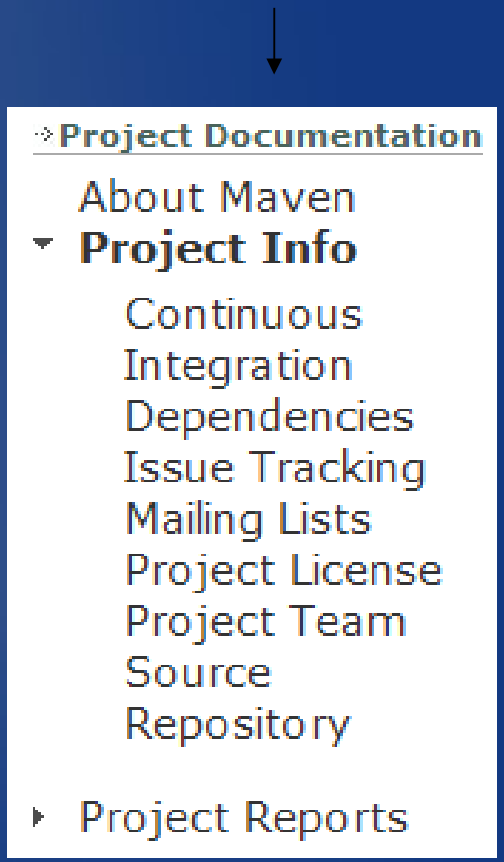
- Different activation conditions
  - JDK version, OS, property defined, existence of file or directory
- Profiles can also modify plugin configurations and other POM elements
  - Merged with the main pom.xml content
- Profiles can be selected on the command line:

```
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>
```



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

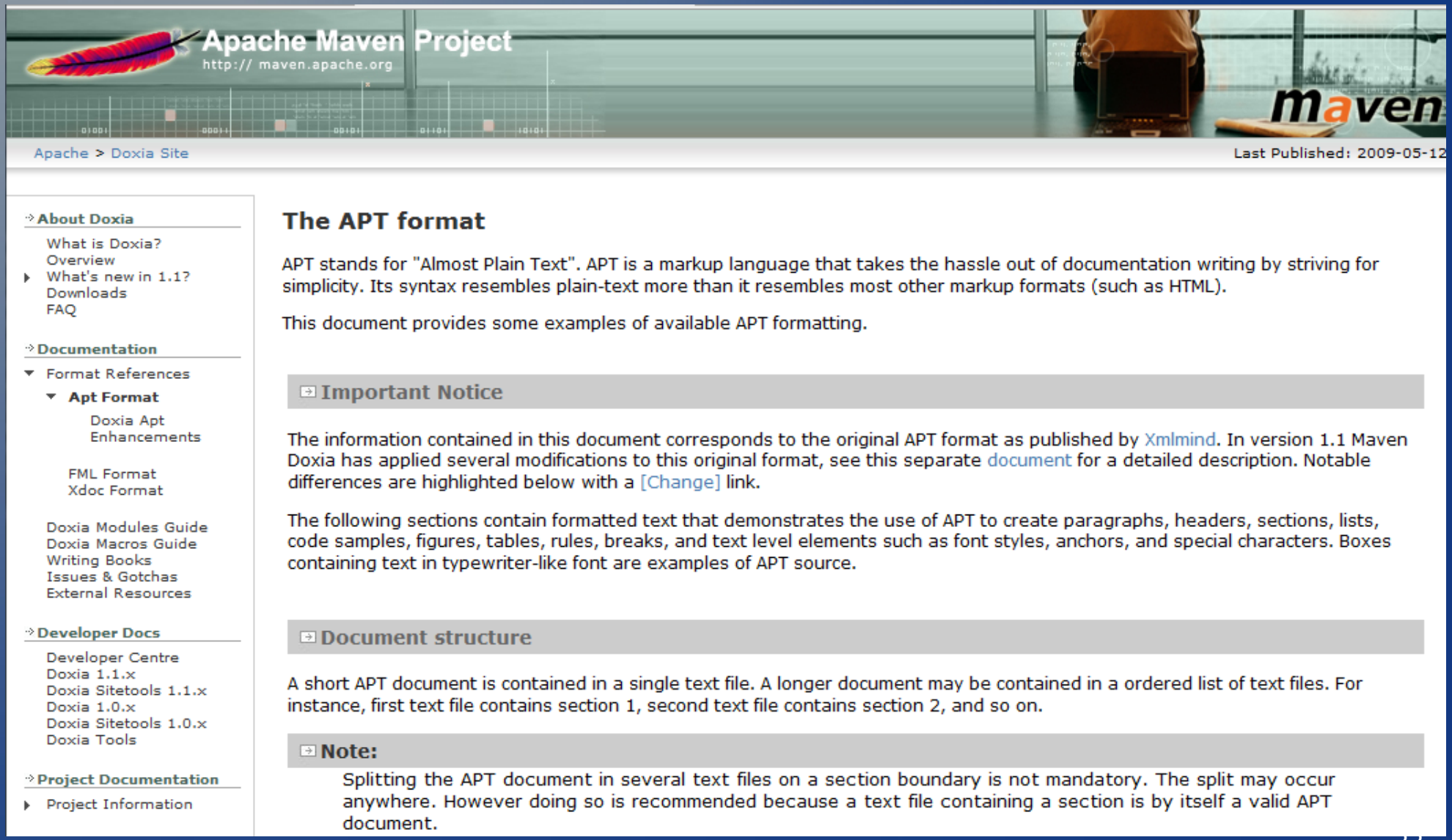
```
mvn site
```

```
+- 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)



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

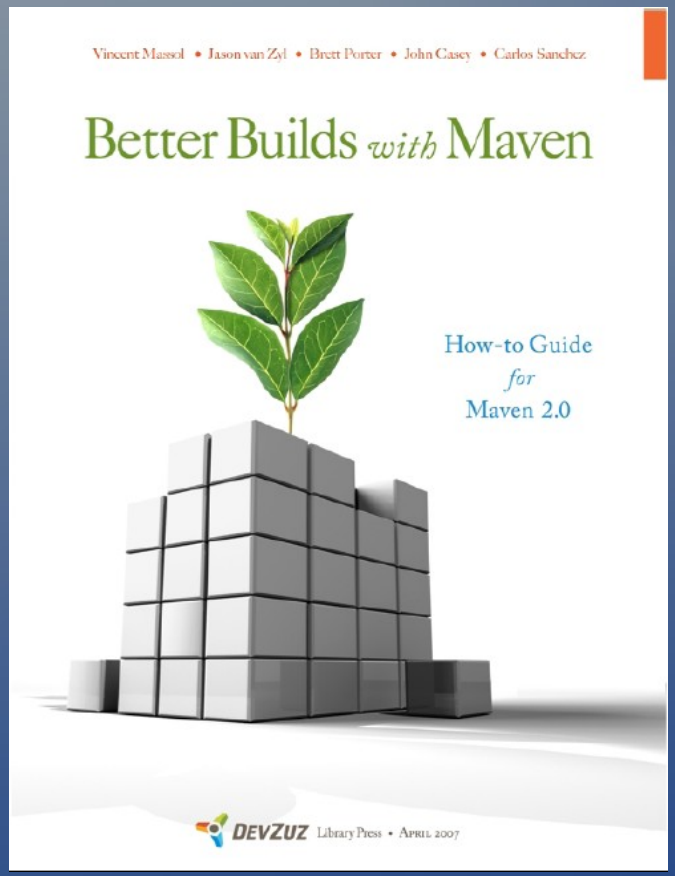
Outlook

- 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

Maven: The Definitive Guide (alpha)
www.sonatype.com/book

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