Apache Maven 2

Didier DONSEZ

Université Joseph Fourier – Grenoble 1

PolyTech Grenoble – LIG/ERODS

Didier.Donsez@imag.fr

Didier.Donsez@ieee.org

Summary

- Motivation
- Installation
- Lifecycle
- Maven project structure
- Plugins Maven
- Developing a plugin
- Bibliographie et Webographie

Motivations

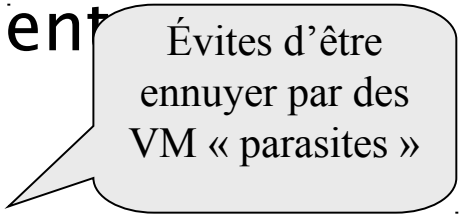
- What is ANT
  - Replaces make (for Java development)
    - NAnt for .NET
  - Task scheduler (grouped in targets)
    - Target graph dependency
  - Many tasks ready to use
- Problems
  - No « standard » project structure
  - No « standard » lifecycle of a project
  - No « standard » metadatas for the project
    - scm, website, ML, …
  - No separation of concerns
  - External libraries to retrieve for project and for tasks
    - problème de la MAJ des versions

Motivations of Maven

- Abstract project model (POM)
  - Object oriented, inheritance
  - Separation of concerns
- Default lifecycle
  - Default state (goals) sequence
    - plugins depend on states
- Structure « standard » of project
  - Standard naming conventions for variables (src.dir, …)
- Automatic handling of dependencies between projects
  - Chargement des MAJ
- Project repositories
  - public or private, local or remotes
  - caching and proxy
- Extensible via external plugins
  - which are Maven projects of course!

Installation

- Download binary distribution
  - http://maven.apache.org
- Unzip inside directory
- Positionner les variables d'environnement
  - `set JAVA_HOME=c:\j2sdk1.x.y`
  - `set MAVEN_HOME=c:\maven-2.z.w`
  - `set PATH=%JAVA_HOME%\bin;%MAVEN_HOME%\bin`
  - `mvn -version`
  - `mvn --help`
- (eventually) Configure ~/.m2/settings.xml
  - repositories, plugins repositories, proxies, …
- Integration with IDE (Eclipse, NetBeans, IDEA, …)
  - http://m2eclipse.sonatype.org/update/



mvn --help

```
usage: mvn [options] [<goal(s)>] [<phase(s)>]
Options:
 -q,--quiet                    Quiet output - only show errors
 -C,--strict-checksums         Fail the build if checksums don't match
 -c,--lax-checksums            Warn if checksums don't match
 -P,--activate-profiles        Comma-delimited list of profiles to activate
 -ff,--fail-fast               Stop at first failure in reactorized builds
 -fae,--fail-at-end            Only fail the build afterwards; allow all non-impacted builds to
                               continue
 -B,--batch-mode               Run in non-interactive (batch) mode
 -fn,--fail-never              NEVER fail the build, regardless of project result
 -up,--update-plugins          Synonym for cpu
 -N,--non-recursive            Do not recurse into sub-projects
 -npr,--no-plugin-registry     Don't use ~/.m2/plugin-registry.xml for plugin versions
 -U,--update-snapshots         Forces a check for updated releases and snapshots on remote
                               repositories
 -cpu,--check-plugin-updates   Force upToDate check for any relevantregistered plugins
 -npu,--no-plugin-updates      Suppress upToDate check for any relevant registered plugins
 -D,--define                   Define a system property
 -X,--debug                    Produce execution debug output
 -e,--errors                   Produce execution error messages
 -f,--file                     Force the use of an alternate POM file.
 -h,--help                     Display help information
```

Project model (POM is Project Object Model)

- Project description independant from actions to perform
  - Object oriented → model inheritance
- Exemple

```
<project>
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.mycompany.app</groupId>
 <artifactId>my-app</artifactId>
 <version>1.0.0-SNAPSHOT</version>
 <packaging>jar</packaging>
 <dependencies>
  <dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>3.8.1</version>
   <scope>test</scope>
  </dependency>
 </dependencies>
 ...
</project>
```

Unique identifier of the project: Identify product artifact

type du projet:
pom, jar, war, ear, bundle, …

project deps (artifact)
it's the $CLASSPATH

id of a dependency
versions can be range

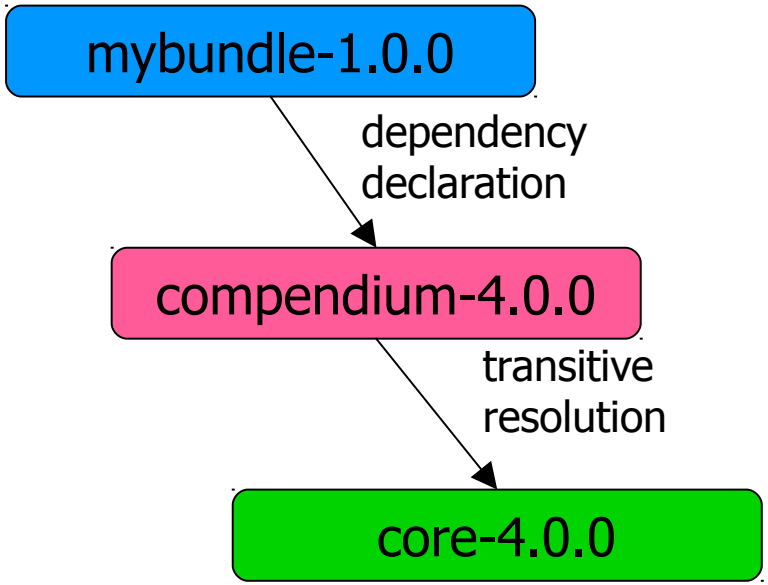
scope of the dependency related to the lifecycle's state (compile, provided, runtime, test)

next to come …

Dependencies

- Useful for artifacts and plugins
- transitive resolution

```
<project>  ...
  <groupId>com.mycompany.app</groupId>
  <artifactId>mybundle</artifactId>
  <version>1.0.0</version> ...
  <dependencies>
    <dependency>
      <groupId>org.osgi</groupId>
      <artifactId>compendium</artifactId>
      <version>4.0.0</version>
</dependency>
 </dependencies>
... </project>
```



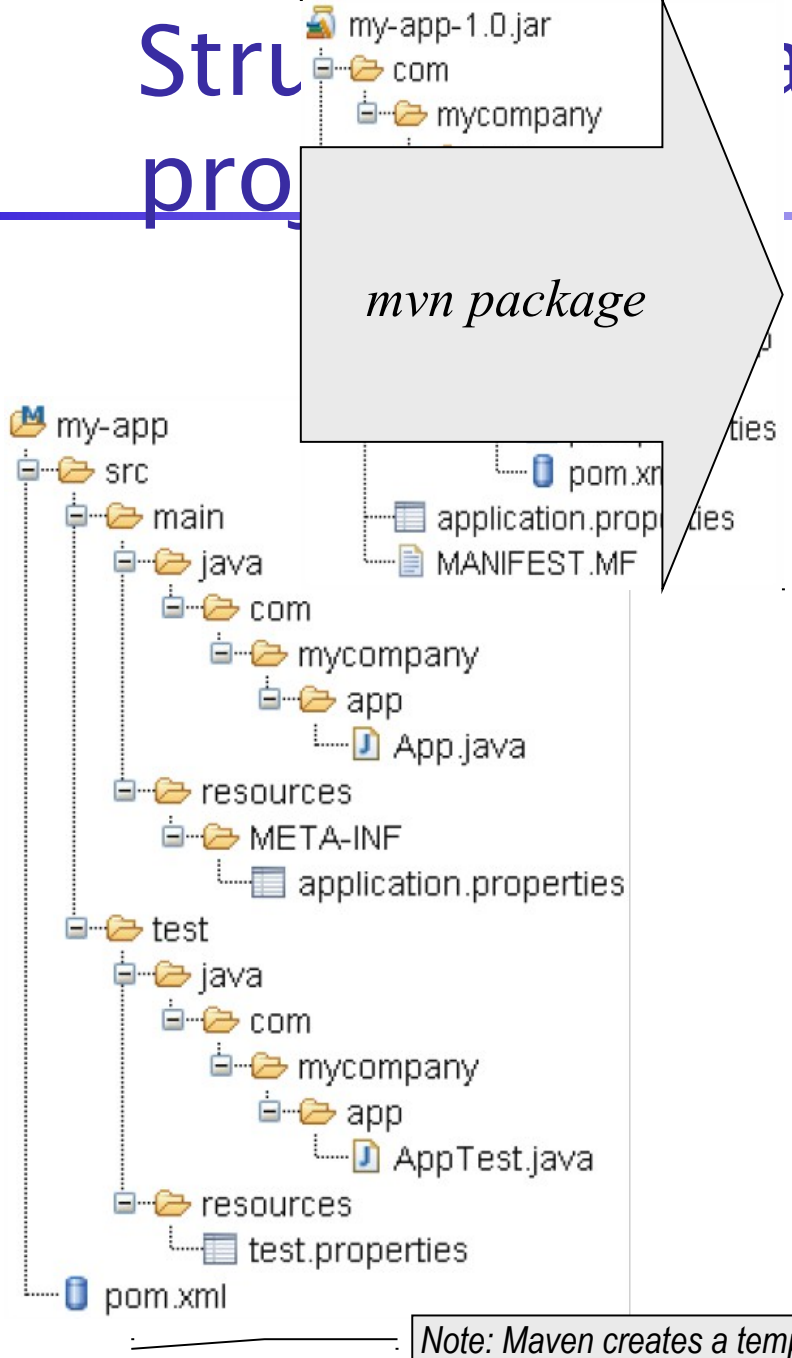
- Used to build the CLASSPATH
  - To compile, to execute the tests, to execute

Scope of dependencie

- **5 scopes possible related to the project classpath**
  - **compile** (default)
    - available with any classpath
    - Transistive vers les projets dépendants
  - **provided**
    - compilation and test classpaths
    - Not transitive.
  - **runtime**
    - runtime and test classpaths.
  - **test**
    - test compilation and execution phases.
  - **system**
    - similar to provided but the artifact is always available and is not looked up in a repository.
  - **import**
    - only used on a (gement> section.
- Portée transistive

	compile	provided	runtime	test
compile	compile	-	runtime	-
provided	provided	provided	provided	-
runtime	runtime	-	runtime	-
test	test	-	test	-

Stru ndard » of a pro

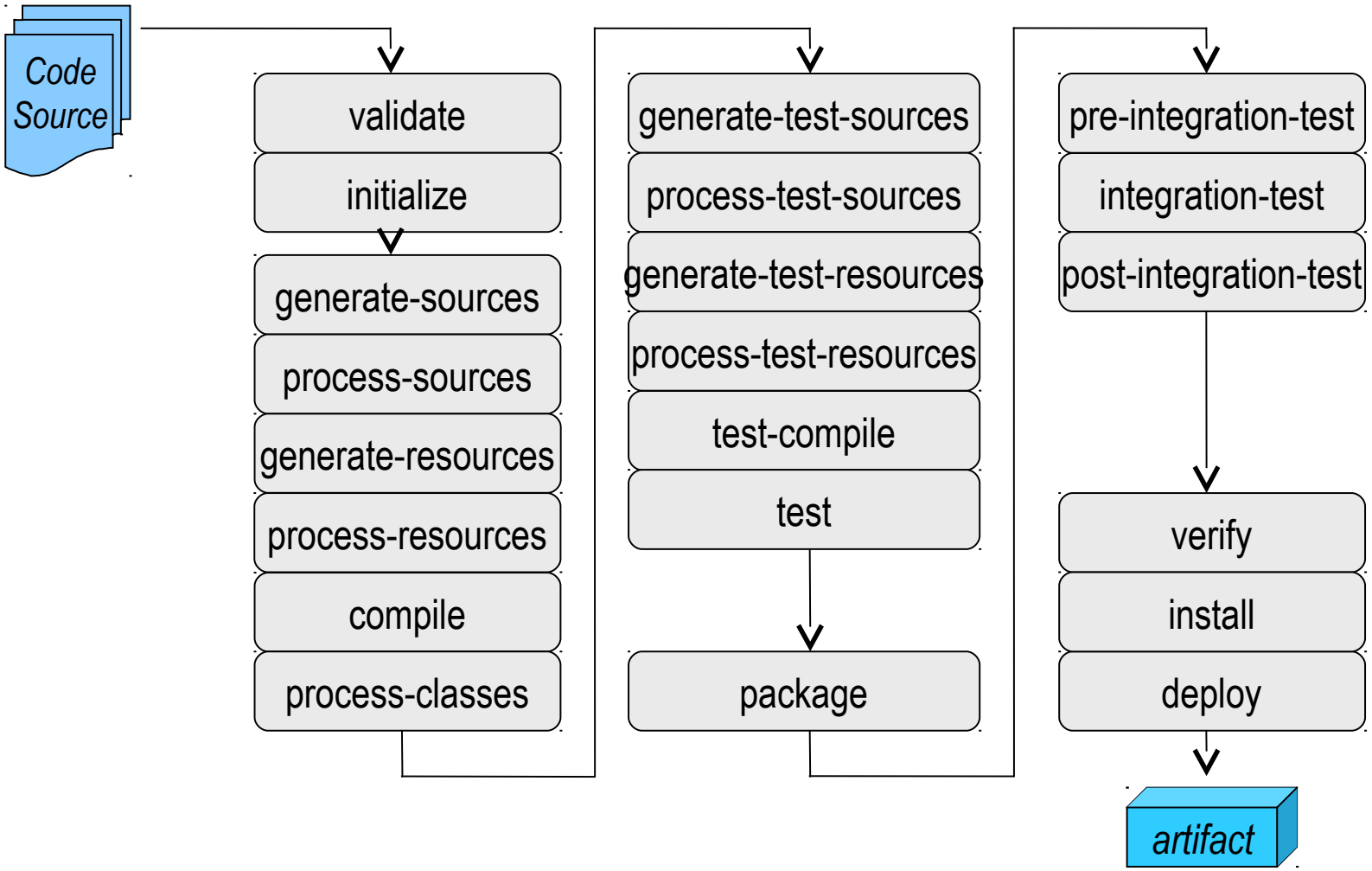


Note: Maven adds the POM to the packaged artifact

Note: Maven creates a temporary repository in ${basedir}/target shared by all plugins

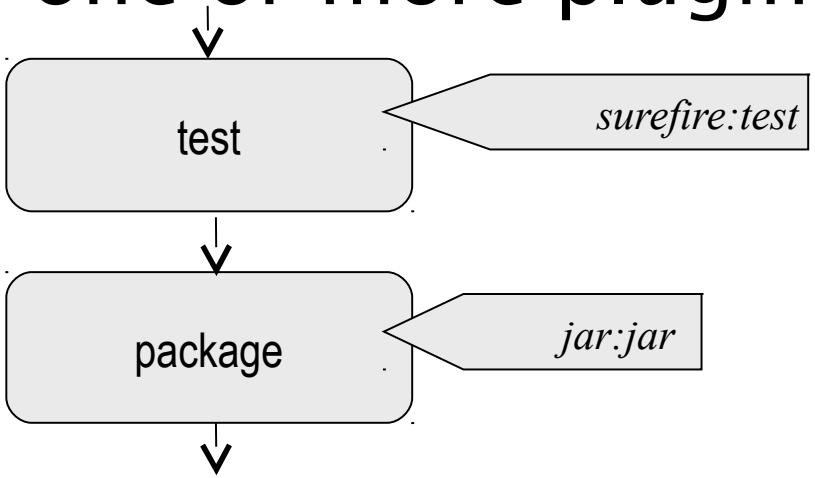
Default lifecycle of a project

- 21 phases



Phases et Goals

- Each phase is associated with one or more goals or one or more plugins



- Note
  - *mvn resources:resources compiler:compile resources:testResources compiler:testCompile surefire:test jar:jar* equals *mvn package*
- Other lifecycles were defined
  - clean = pre-clean → clean → post-clean
  - site = pre-site → site → post-site → site-deploy
  - …

Reminder
Version numbering

- Scheme

  <major>.<mini>[.<micro>][-<qualifier>[-<buildnumber>]]
- Increment
  - Major : major change
    - (possibly) break retro-compatibility
  - Mini : add features
    - retro-compatibility compatibility
  - Micro : maintenance corrective (bug fix)
- Qualifiers
  - `SNAPSHOT` (Maven) : version in evolution
  - `alpha1` : version alpha (very unstable and incomplete)
  - `beta1, b1, b2` : version beta (unstable)
  - `rc1, rc2` : release candidate
  - `m1, m2` : milestone
  - `ea` : early access
  - `20081014123459001` : build date
- Version ordering
    - Différent de l'ordre lexicographique
  - `1.1.1 < 1.1.2 < 1.2.2`
  - `1.1.1-SNAPSHOT < 1.1.1`
  - `1.1.1-alpha1 < 1.1.1-alpha2 < 1.1.1-b1 < 1.1.1-rc1 < 1.1.1-rc2 < 1.1.1`

Versioning

- Snapshot
  - A **snapshot in Maven** is an artifact which has been prepared using the **most recent** sources available. … **Specifying a snapshot version** for a dependency means that Maven will look for new versions of that dependency without you having to manually specify a new version.
  - mvn -U command line option to force the search for updates.

- Dependencies
  - Specify version ranges

```
<dependency>
  <groupId>org.codehaus.plexus</groupId>
  <artifactId>plexus-utils</artifactId>
  <version>[1.1,)</version>
</dependency>
```

Range	Meaning
(,1.0]	Less than or equal to 1.0
[1.2,1.3]	Between 1.2 and 1.3 (inclusive)
[1.0,2.0)	Greater than or equal to 1.0, but less than 2.0
[1.5,)	Greater than or equal to 1.5
(,1.1),(1.1,)	Any version, except 1.1

Common plugins

- Core
  - clean, compiler, deploy, install, resources, site, surefire, verifier
- Packaging
  - ear, ejb, jar, rar, war, bundle (OSGi)
- Reporting
  - changelog, changes, checkstyle, clover, doap, docck, javadoc, jxr, pmd, project-info-reports, surefire-report
- Tools
  - ant, antrun, archetype, assembly, dependency, enforcer, gpg, help, invoker, one (interop Maven 1), patch, plugin, release, remote-resource, repository, scm
- IDEs
  - eclipse, netbeans, idea
- Others
  - exec, jdepend, castor, cargo, jetty, native, sql, taglist, javacc, obr …

Plugin configuration

- Specify more parameters
- Example

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.5</source>
        <target>1.5</target>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <configuration>
        <archive>
          <manifest>
            <mainClass>${artifactId}.Main</mainClass>
            <addClasspath>true</addClasspath>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
```

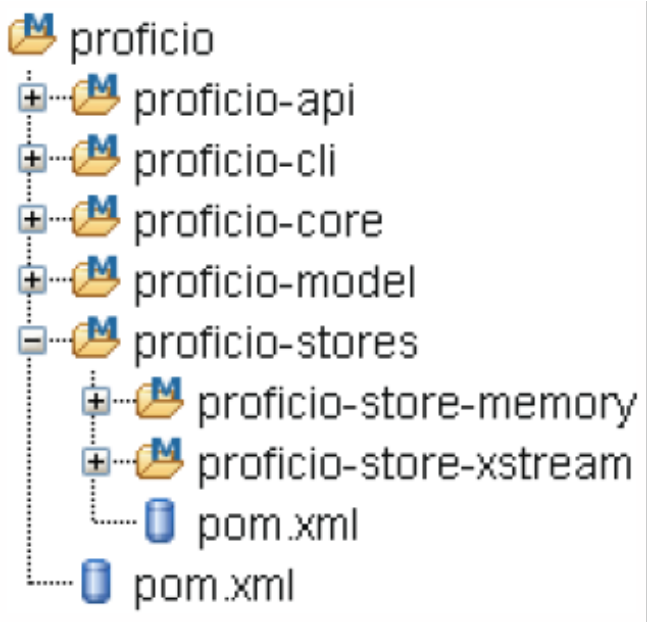
Profiles

- Motivation
  - Better project portability (compared to environment
    - different JVM, Java versions, servers JEE, DBMS,

  → To create project differentiations (=profils)
- Element <profile>
  - Defines the variations to the plugins
- Profile activation
  - Default Profile
  - Using properties (systems, JDK, …)
  - Using its identifier
    - mvn --activate-profiles felix,equinox clean install

Project hierarchies

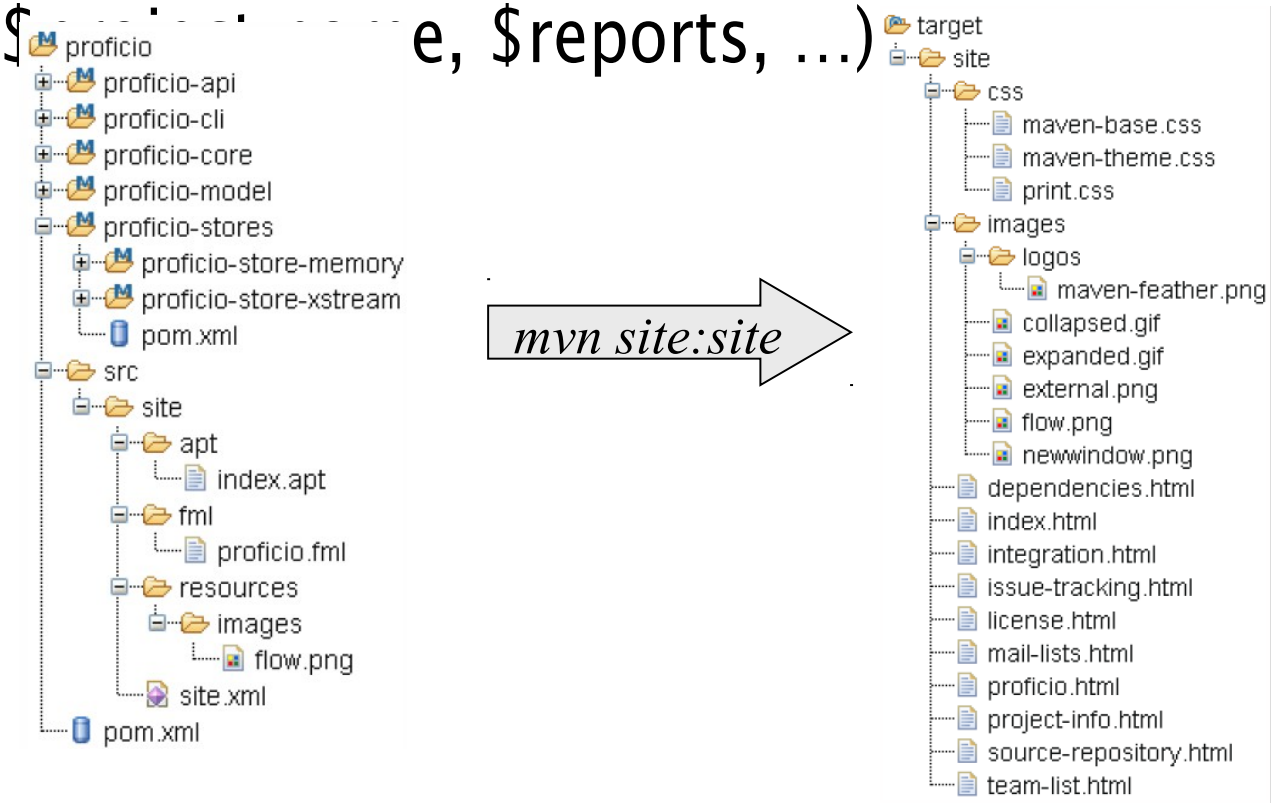
- Motivations
  - Organize development in sub-projects
    - With N levels (N>=1)
- Technique
  - Create a super POM (type *pom*) for each nesting level
    - Place shared plugins/goals at the same level
  - Subprojects (called modules) inherit from this super pom
- Example



- Command
  - mvn clean install
    - Global construction

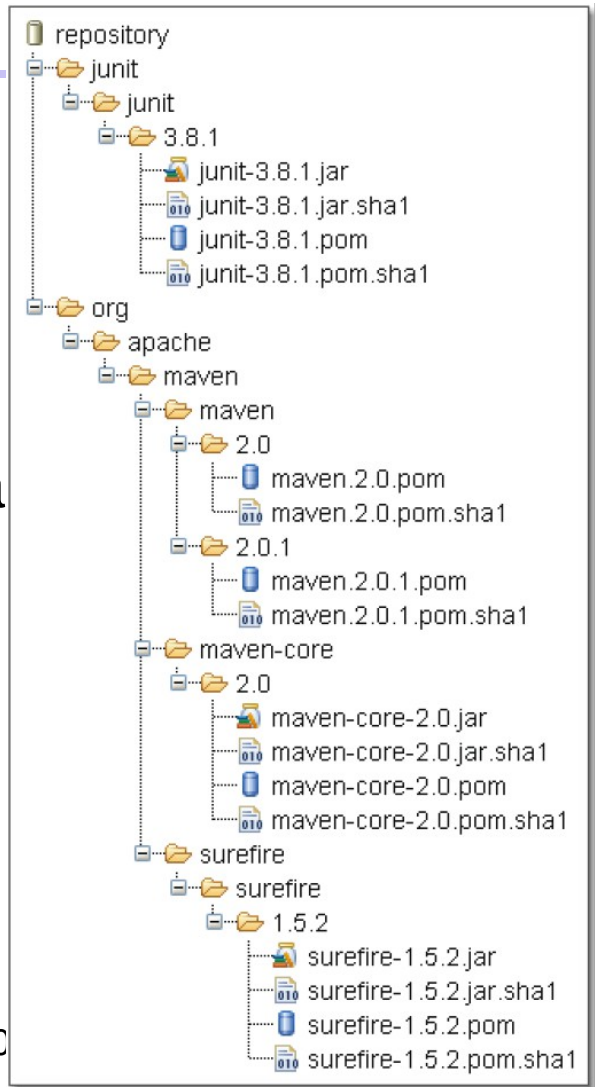
Documentation Web

- Transform several doc formats
  - XDOC, APT (Almost Plain Text), FML (FAQ ML), DocBook Simple, Twiki, Confluence
  - Sorce documentation can use project variables ($[illegible]e, $reports, ...)

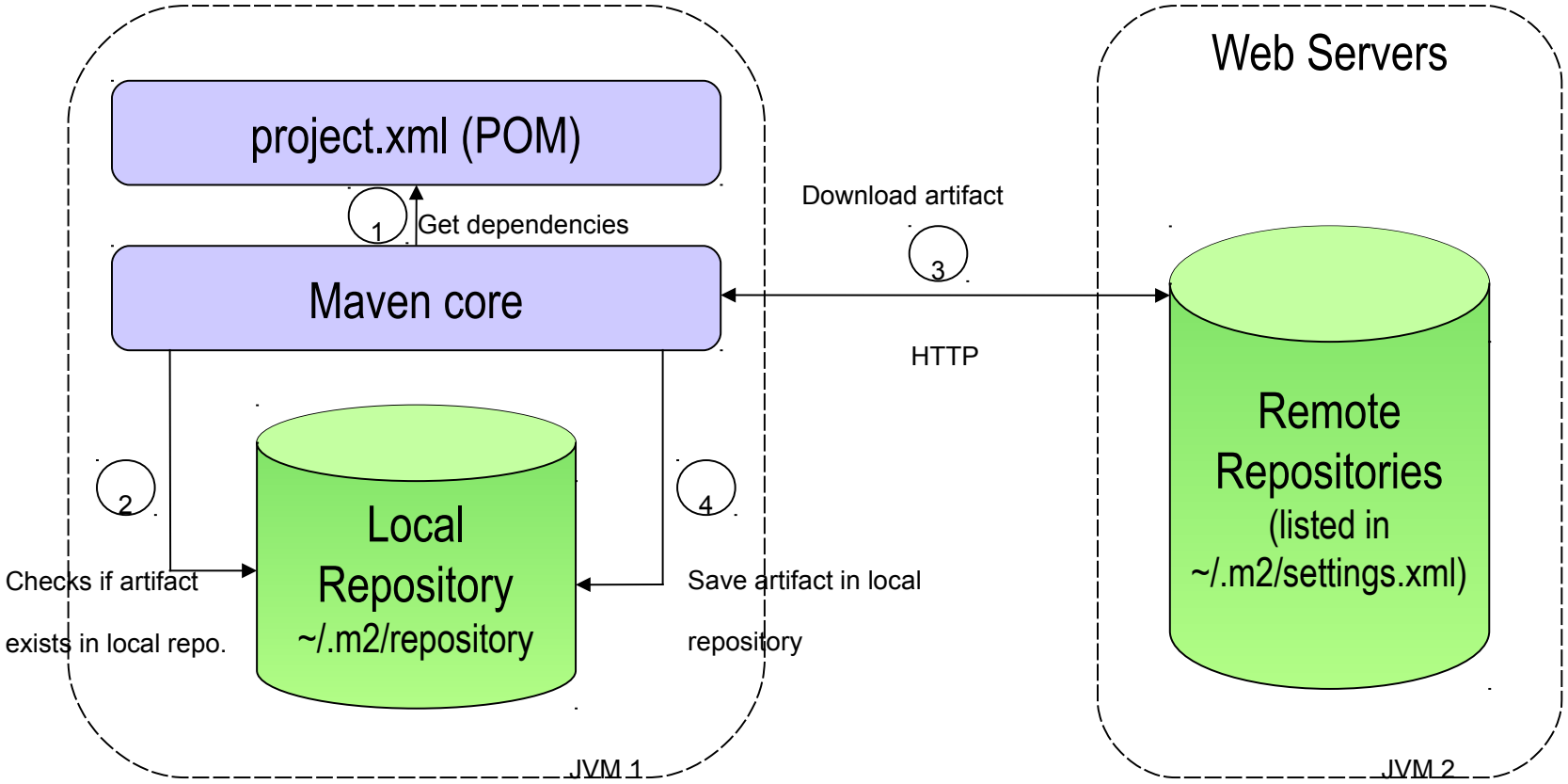


Project repository

- Local ~/.m2/repository
  - Projects (artifacts) installed locally
    - mvn install
    - mvn install:install-file
  - Caches of projects (artifacts) downloa from remote repositories
    - Listed in POMs and settings.xml
- Remote repositories
  - Enterprise repo's
  - Cache of repositories
  - Main ones
    - http://repo1.maven.org/maven2/
      - Plus de 20000 artifacts décrits (en 20
- Structure
  - Hierchical naming conventions
    - ${groupId}.replace('.','/') / ${articfactId} / ${version}



Dependency resolution



R1: Local repository update is done daily (unless `mvn -U`)
R2: Plugins and artifacts are updated in the same way

by Vincent Massol

Construction-time variable replacement

- Motivations
  - Instantiate resource values during the process-resources phase
- Example de POM

```
… <build>
 <filters>
  <filter>src/main/filters/filter.properties</filter>
 </filters>
 <resources>
  <resource>
   <directory>src/main/resources</directory>
   <filtering>true</filtering>
  </resource>
 </resources>
</build>
```

```
# src/main/filters/filter.properties
my.filter.value=Hello !
```

```
# src/main/resources/application.properties
message=${my.filter.value}
application.name=${project.name}
application.version=${project.version}
```

Archetype

- Initial build of a Maven project
  - Depends on the type T of the project
    - T= quickstart, archetype, bundles, j2ee-simple, marmalade-mojo, mojo, plugin, plugin-site, portlet, profiles, simple, site, site-simple, webapp, …
- Example
  - mvn archetype:create
    mode interactif
  - mvn archetype:create
    -DgroupId=demo.maven
    -DartifactId=hello
    -Dversion=0.1.0-SNAPSHOT
    -DarchetypeGroupId=org.apache.maven.archetypes
    -DarchetypeArtifactId=maven-archetype-quickstart

Customized archetypes

- Possible to create custom archetypèes
  - → from scratch

    mvn archetype:create
    -DarchetypeGroupId=org.apache.maven.archetypes
    -DarchetypeArtifactId=maven-archetype-archetype
    -DgroupId=com.mycompany
    -DartifactId=my-archetype
  - → from pre-existing archetype
- Development
  - Based on the Velocity template engine (http://velocity.apache.org/)

Plugin development

- Plugin = { <goal,MOJO> }
- MOJO = Maven POJO
  - Annotations XDocLet

- Languages
  - Java and Groovy (for scripting)
  - Others as well (ruby..)

- Deployment
  - Artifact Maven
  - Use the same deployment mechanism (version, dépendances, …)
  - Plugin repositories
    - http://maven.apache.org/plugins/, http://repository.codehaus.org/

Plugin development Example (i)

```
package sample.plugin;
import org.apache.maven.plugin.AbstractMojo;
import org.apache.maven.plugin.MojoExecutionException;
/**
 * Says "Hi" to the user.
 * @goal sayhi
 * @phase compile
 */
public class GreetingMojo extends AbstractMojo {
    /** The greeting to display.
     * @parameter alias="message" expression="Hello, world (from ${project.groupId}:${
     {project.artifactId})" */
    private String greeting;

/** The classpath.
   * @parameter expression="${project.compileClasspathElements}"
  * @required
  * @readonly */
  private List classpathElements;
    public void execute() throws MojoExecutionException {
      getLog().info(greeting);
```

phase and goal during which execute() is invoked

parameter used in <configuration>

Integer, …, String, List, Properties, Map, Object, File, URL, …

paramètre issue du pom

Plugin development
Example (ii)

- Dans le POM

```
<build>
  <plugins>
    <plugin>
      <groupId>sample.plugin</groupId>
      <artifactId>maven-hello-plugin</artifactId>
      <configuration>
          <message>Welcome</message>
      </configuration>
    </plugin>
  </plugins>
</build>
```

- Execution

```
mvn sample.plugin:maven-hello-plugin:sayhi
```

Plugins and Lifecycle

- MOJO attached to a lifecycle phase
  - @nnotations doclet
- Customiized lifecycles
  - Overloading META-INF/plexus/components.xml,

Call ANT tasks from Maven

- Motivations
  - recover existing projects before the migration
  - Execute legacy tasks missing equivalent plugins
    - Note: think about using Macro ANT !
- Example with plugin org.apache.maven.plugins:maven-antrun-plugin

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-antrun-plugin</artifactId>
  <executions>
    <execution>
      <phase>generate-sources</phase>
      <configuration>
<tasks unless="maven.test.skip">
      <!-- Place any ant task here. You can add anything
    you can add between <target> and </target> in a build.xml.-->
        <echo message="To skip me, just call mvn -Dmaven.test.skip=true"/>
      <exec dir="${basedir}"
            executable="${basedir}/src/main/sh/do-something.sh" failonerror="true">
        <arg line="arg1 arg2 arg3 arg4" />
      </exec>
    </tasks>
      </configuration>
      <goals>
        <goal>run</goal>
      </goals>
```

Migrate ANT project to Maven

- 2 options for the project structure
  - Reorganize manually
    - src → src/main/java, src/test/java, doc → src/site
    - classes → target/classes, build → target, …
  - Configure default parameters of POM according to the structure of the ANT project
- Defines dependencies
  - using <classpath …>

Antlib for Maven

- Tasks Maven for ANT projects
  - Manipulate artifacts from an ANT project
    - Transitive dependency management
      - scope recognition and SNAPSHOT handling
  - Deploy artifacts into a Maven repository
  - Analyze pom.xml
- Example

```
<artifact:dependencies pathId="dependency.classpath">
   <dependency groupId="javax.servlet" artifactId="servlet-api"
         version="2.4" scope="provided" />
   ...
</artifact:dependencies>
<javac ...>
   <classpath refid="dependency.classpath" />
   ...
</javac>
```

Maven and other languages

- Maven is very Java-centric
- Projects for other enviroments and languages
  - .NET, …
  - JNI, C, C++, C#, PHP, JavaScript, GWT, Basic, ..
- Project structure
  - src/main/java
  - src/main/c
  - src/main/cpp
  - src/main/cs
  - src/main/php
  - src/main/vb
  - …
- Plugins
  - maven-antrun-plugin (org.apache.maven.plugins:)
  - native-maven-plugin (org.codehaus.mojo:)

Misc

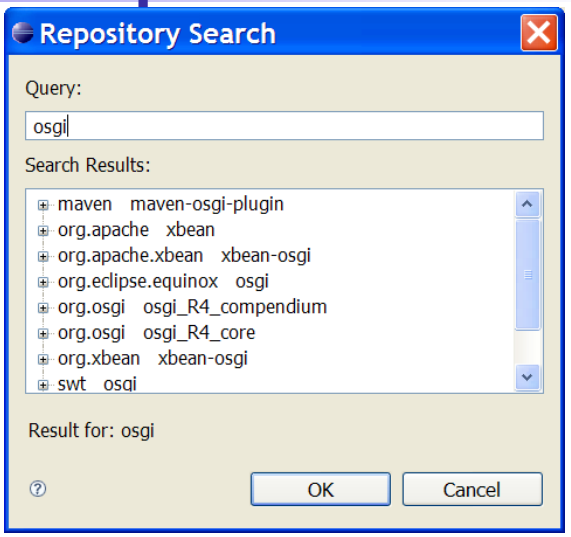
- Maven SCM
  - Plugin offering common API for main SCM
  - Commands
    - Changelog – command to show the source code revisions
    - Checkin – command for commiting changes
    - Checkout – command for getting the source code
    - Diff – command for showing the difference of the working copy with the remote ones
    - Edit – command for starting edit on the working copy
    - Status – command for showing the scm status of the working copy
    - Tag – command for tagging the certain revision
    - UnEdit – command for to stop editing the working copy
    - Update – command for updating the working with the latest changes
    - Validate – validates the scm information on the pom
  - Supported SCM
    - Subversion, CVS, Starteam, Clearcase, Perforce, bazaar
- Maven Continuum
  - continuous integration (JEE-based) server for building Java based projects.
    - Schedulable projects: Maven 1, Maven 2, Ant, Shell scripts
    - Notifications : Mail and IM (IRC, Jabber, MSN)
- Maven Archiva
  - Repository manager (search, security, reporting, …)
- Maven Wagon
  - Tool to send artifacts on a varity of repositories (remote or local ones)
    - File, HTTP, HTTP lightweight, FTP, SSH/SCP, WebDAV, SCM

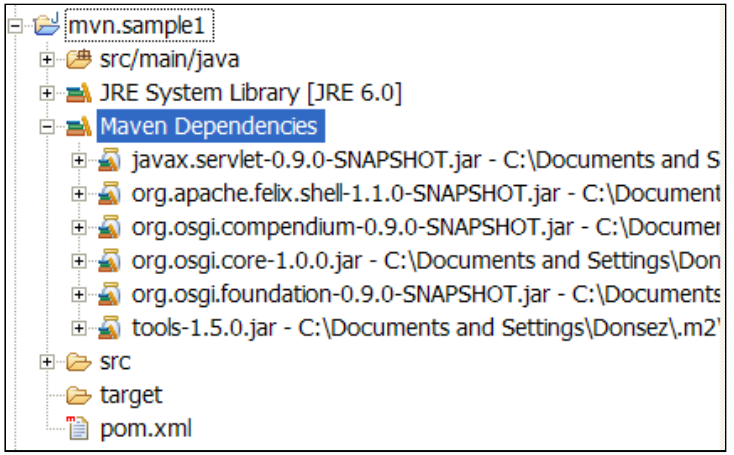
Misc

- Apache Ivy
  - Dependency Manager (for Ant projects)
    - Sous gestionnaire pour des dépôts Maven (locaux ou distants)

Plugin Maven pour Eclipse

- Search for dependencies
  - from local or remote repositories
- Wizard to edit the POM
- Graph dependency
  - helps to discover (potential) conflicts





- Add dependencies to POM
- Add POM dep's to .classpath
  - org.maven.ide.eclipse.MAVEN2_CLASSPATH_CONTAINER
- Execute main phases : clean, test, install, …

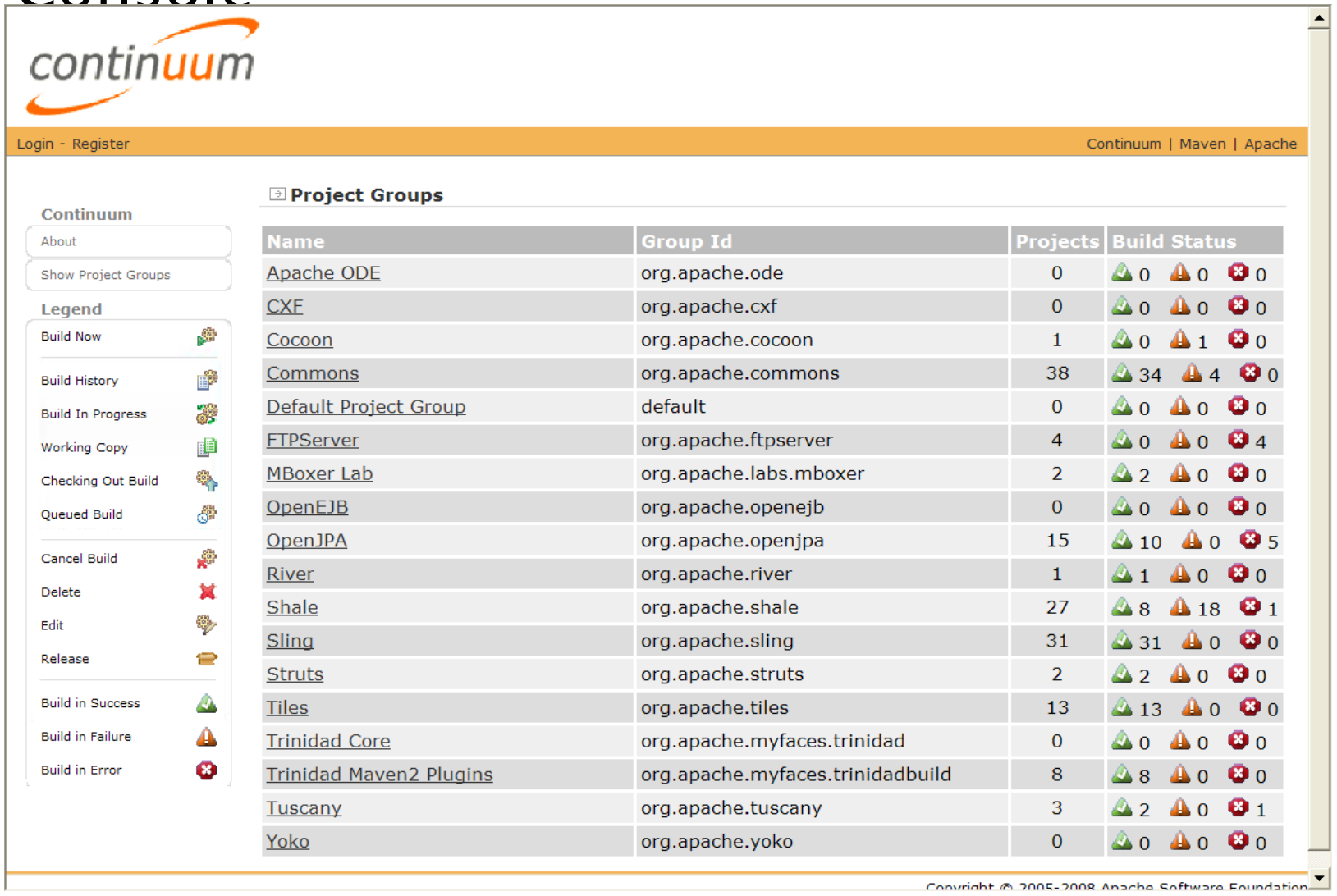
Maven Continuum

http://maven.apache.org/continuum/

- continuous integration (JEE-based) server for building Java based projects.
  - Schedulable projects: Maven 1, Maven 2, Ant, Shell scripts
  - Notifications : Mail and IM (IRC, Jabber, MSN)
  - Release management
  - SCM support
    - CVS, Subversion, Clearcase, Perforce, Starteam, Visual Source Safe, CM Synergy, Bazaar, Mercurial
  - External access with XMLRPC

Maven Continuum

- Console



Misc

- Search artifacts
  - http://www.mvnrepository.com/

Misc

- Tree Surgeon (Maven for .NET ?)
  - http://www.codeplex.com/treesurgeon
  - http://confluence.public.thoughtworks.org/display/TREE/Tree+Surgeon
  - « Tree Surgeon is an Open Source tool by the software company of Martin Fowler, Thoughtworks" "It is a tool that automates the process of establishing a directory structure with source code stubs and supporting infrastructure in a consistent manner. (...) It supports tools like NAnt and NUnit by generating build files and unit tests as part of the automated process. . Far from the same functional level of Maven,it represents an excellent base to industrialize generation of .NET project skeletons»

Bibliography and Webography

- Web
  - Site Maven, http://maven.apache.org
- Examples et exercises
  - http://www-adele.imag.fr/users/Didier.Donsez/cours/tpmvn
- Also :
  - http://www-adele.imag.fr/users/Didier.Donsez/cours/coursjavaoutil.pdf

Bibliographie et Webographie
http://maven.apache.org/articles.html

- Books
  - Maven: The Definitive Guide
    - http://www.sonatype.com/book/maven-user-guide.pdf
  - John Casey, Vincent Massol, Brett Porter, Carlos Sanchez, Jason van Zyl, Better Builds with Maven, Publisher Mergere Library Press, March 2006
    - (free PDF online)
    - Excellent, but prior knowledge of Maven is useful!
  - Vincent Massol, Tim O'Brien, Maven: A Developer's Notebook, Publisher O'Reilly, July 2005
  - French FAQ
    - http://java.developpez.com/faq/maven/

Developping a plugin

- Develop a Velocity plugin
  - and also for DVSL (Declarative Velocity Style Language)
- Questions
  - Which phase to attach this plugin to ?
  - …

Reminder on Apache Velocity

- Template language (VTL)
  - Next to CPP syntax
    - Macros #set, #foreach() ... #end, #if () ...#elseif () ...#else ...#end, #include(...), #parse(...)
    - DVSL (Declarative Velocity Style Language) #match() ... #end
    - Variables $var or ${var}
- Usage
  - Web page generation
  - Code source generation (generative programming), …
- Example

```
// generated at $date
package ${pkgName};
public interface ${itfName}MBean {
#foreach ($attribute in ${attributesList})
    /** setter for the attribute ${attribute} */
    public void set${attribute}(String new${attribute});
    /** getter for the attribute ${attribute} */
    public String get${attribute}();
#end
    /** reset all the attributes */
    public void reset();
}
```

```
public static void main(String [] args) {
  Velocity.init();
  VelocityContext vc = new VelocityContext();
  vc.put("date", new Date());
  vc.put("itfName", "Config"); …
  Template template
     = Velocity.getTemplate(args[0]);
  OutputStreamWriter osw =
                new StringWriter(System.out);
  template.merge(vc, osw);
}
```