Maven

N. Vinot

Viveris Technologies

28 juin 2011

Sommaire

Présentation

Origines

Projet apparu en 2002

- Géré par la fondation Apache
- Développé en grande partie par Sonatype

Versions

3ème version

- v1 (2002) : flop total, jamais utilisé
- v2 (2005) : refonte intégrale, est devenu LA référence Java
- v3 (2010) : adaptation à d'autres langages, plugins. . .

Présentation

Origines

Projet apparu en 2002

- Géré par la fondation Apache
- Développé en grande partie par Sonatype

Versions

3ème version

- v1 (2002) : flop total, jamais utilisé
- v2 (2005) : refonte intégrale, est devenu LA référence Java
- v3 (2010) : adaptation à d'autres langages, plugins...

Objectifs

Constat de l'équipe Maven

Pourquoi fait-on tous pareil mais :

1. Manuellement
2. Jamais de la même manière

Principes fondamentaux de Maven :

1. Homogénéiser les projets
2. Automatiser la chaîne de développement
3. Intégrer des outils

Objectifs

Constat de l'équipe Maven

Pourquoi fait-on tous pareil mais :

1. Manuellement
2. Jamais de la même manière

Principes fondamentaux de Maven :

1. Homogénéiser les projets
2. Automatiser la chaîne de développement
3. Intégrer des outils

Objectifs

Constat de l'équipe Maven

Pourquoi fait-on tous pareil mais :

1. Manuellement
2. Jamais de la même manière

Principes fondamentaux de Maven :

1. Homogénéiser les projets
2. Automatiser la chaîne de développement
3. Intégrer des outils

Objectifs

Constat de l'équipe Maven

Pourquoi fait-on tous pareil mais :

1. Manuellement
2. Jamais de la même manière

Principes fondamentaux de Maven :

1. Homogénéiser les projets
2. Automatiser la chaîne de développement
3. Intégrer des outils

Objectifs

Constat de l'équipe Maven

Pourquoi fait-on tous pareil mais :

1. Manuellement
2. Jamais de la même manière

Principes fondamentaux de Maven :

1. Homogénéiser les projets
2. Automatiser la chaîne de développement
3. Intégrer des outils

Objectifs

Constat de l'équipe Maven

Pourquoi fait-on tous pareil mais :

1. Manuellement
2. Jamais de la même manière

Principes fondamentaux de Maven :

1. Homogénéiser les projets
2. Automatiser la chaîne de développement
3. Intégrer des outils

Objectifs

Constat de l'équipe Maven

Pourquoi fait-on tous pareil mais :

1. Manuellement
2. Jamais de la même manière

Principes fondamentaux de Maven :

1. Homogénéiser les projets
2. Automatiser la chaîne de développement
3. Intégrer des outils

Homogénéité des projets

Avant Maven

- Organisation des sources différentes
- Nommage des projets différents
- Gestion des versions différentes

Et pourtant les projets dépendent les uns des autres !

Après Maven

« Convention over configuration »

Convention plutôt que configuration

Homogénéité des projets

Avant Maven

- Organisation des sources différentes
- Nommage des projets différents
- Gestion des versions différentes

Et pourtant les projets dépendent les uns des autres !

Après Maven

« Convention over configuration »

Convention plutôt que configuration

Homogénéité des projets

Avant Maven

- Organisation des sources différentes
- Nommage des projets différents
- Gestion des versions différentes

Et pourtant les projets dépendent les uns des autres !

Après Maven

« Convention over configuration »

Convention plutôt que configuration

Organisation inter-projets

Notion d'assembly $\sim$ projet, livrable, dll, exe...

Chaque assembly est décrite par un fichier `pom.xml`

Chaque assemly est identifiée par une coordonnée Maven

- `groupId` :
  identifiant de l'équipe de développement
- `artifactId` :
  identifiant du projet
- `version` :
  normalisée, M.m.r(-SNAPSHOT)
- `packaging` :
  type de l'assembly (pom, jar, war, ear...)

Exemple de POM

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>fr.grouperatp.ratp.satan</groupId>
    <artifactId>satan-parent</artifactId>
    <version>2.0.2.1-SNAPSHOT</version>
    <packaging>pom</packaging>
</project>
```

Organisation intra-projet

Standard Directory Layout

- `pom.xml`
- `src/main/java` :
  les sources Java
- `src/main/resources` :
  les ressources (conf, XSD, XML...)
- `src/test/java` :
  les sources Java des tests unitaires
- `src/test/resources` :
  les ressources des tests unitaires
- `src/main/webapp` :
  site internet applicatif
- `src/site` :
  site internet projet

Organisation intra-projet

Standard Directory Layout

- `pom.xml`

- `src/main/java` :
  les sources Java
- `src/main/resources` :
  les ressources (conf, XSD, XML...)

- `src/test/java` :
  les sources Java des tests unitaires
- `src/test/resources` :
  les ressources des tests unitaires

- `src/main/webapp` :
  site internet applicatif
- `src/site` :
  site internet projet

Organisation intra-projet

Standard Directory Layout

- `pom.xml`
- `src/main/java` :
  les sources Java
- `src/main/resources` :
  les ressources (conf, XSD, XML...)
- `src/test/java` :
  les sources Java des tests unitaires
- `src/test/resources` :
  les ressources des tests unitaires
- `src/main/webapp` :
  site internet applicatif
- `src/site` :
  site internet projet

Organisation intra-projet

Standard Directory Layout

- `pom.xml`
- `src/main/java` :
  les sources Java
- `src/main/resources` :
  les ressources (conf, XSD, XML...)
- `src/test/java` :
  les sources Java des tests unitaires
- `src/test/resources` :
  les ressources des tests unitaires
- `src/main/webapp` :
  site internet applicatif
- `src/site` :
  site internet projet

Organisation intra-projet

Standard Directory Layout

- `pom.xml`
- `src/main/java` :
  les sources Java
- `src/main/resources` :
  les ressources (conf, XSD, XML...)
- `src/test/java` :
  les sources Java des tests unitaires
- `src/test/resources` :
  les ressources des tests unitaires
- `src/main/webapp` :
  site internet applicatif
- `src/site` :
  site internet projet

Organisation intra-projet

Nommage des fichiers

- `src/main/java/**/*.java` :
  compilé et intégré dans l'application finale
- `src/main/resources/**/*` :
  intégré dans l'application finale

- `src/test/java/**/*.java` :
  compilé pendant les tests unitaires
- `src/test/java/**/Test*.java`,
  `src/test/java/**/*Test.java`,
  `src/test/java/**/*TestCase.java` :
  exécuté à la recherche de tests unitaires
- `src/test/resources/**/*` :
  intégré dans le classpath lors des tests unitaires

- `src/main/webapp/**/*` :
  intégré dans le WAR final

Organisation intra-projet

Nommage des fichiers

- `src/main/java/**/*.java` :
  compilé et intégré dans l'application finale
- `src/main/resources/**/*` :
  intégré dans l'application finale

- `src/test/java/**/*.java` :
  compilé pendant les tests unitaires
- `src/test/java/**/Test*.java`,
  `src/test/java/**/*Test.java`,
  `src/test/java/**/*TestCase.java` :
  exécuté à la recherche de tests unitaires
- `src/test/resources/**/*` :
  intégré dans le classpath lors des tests unitaires

- `src/main/webapp/**/*` :
  intégré dans le WAR final

Organisation intra-projet

Nommage des fichiers

- `src/main/java/**/*.java` :
  compilé et intégré dans l'application finale
- `src/main/resources/**/*` :
  intégré dans l'application finale

- `src/test/java/**/*.java` :
  compilé pendant les tests unitaires
- `src/test/java/**/Test*.java`,
  `src/test/java/**/*Test.java`,
  `src/test/java/**/*TestCase.java` :
  exécuté à la recherche de tests unitaires
- `src/test/resources/**/*` :
  intégré dans le classpath lors des tests unitaires

- `src/main/webapp/**/*` :
  intégré dans le WAR final

Organisation intra-projet

Nommage des fichiers

- `src/main/java/**/*.java` :
  compilé et intégré dans l'application finale
- `src/main/resources/**/*` :
  intégré dans l'application finale

- `src/test/java/**/*.java` :
  compilé pendant les tests unitaires
- `src/test/java/**/Test*.java`,
  `src/test/java/**/*Test.java`,
  `src/test/java/**/*TestCase.java` :
  exécuté à la recherche de tests unitaires
- `src/test/resources/**/*` :
  intégré dans le classpath lors des tests unitaires

- `src/main/webapp/**/*` :
  intégré dans le WAR final

Gestion des dépendances

« Projet A » dépend de « Projet B » pour fonctionner

Avant Maven

- Installation « sauvage »
  - C:/Program Files/libB/libB.dll
  - /trunk/A/lib/libB.dll
- Utilisation « sauvage »
  - Makefile : gcc -l C:/Program Files/libB/libB.dll
  - IDE : /trunk/A/lib/libB.dll

Gestion des dépendances

« Projet A » dépend de « Projet B » pour fonctionner

Avant Maven

- Installation « sauvage »
  - `C:/Program Files/libB/libB.dll`
  - `/trunk/A/lib/libB.dll`
- Utilisation « sauvage »
  - Makefile : `gcc -l C:/Program Files/libB/libB.dll`
  - IDE : `/trunk/A/lib/libB.dll`

Gestion des dépendances

« `Projet A` » dépend de « `Projet B` » pour fonctionner

Avant Maven

- Installation « sauvage »
  - `C:/Program Files/libB/libB.dll`
  - `/trunk/A/lib/libB.dll`
- Utilisation « sauvage »
  - `Makefile : gcc -l C:/Program Files/libB/libB.dll`
  - IDE : `/trunk/A/lib/libB.dll`

Gestion des dépendances

« Projet A » dépend de « Projet B » pour fonctionner

Avant Maven

- Installation « sauvage »
  - C:/Program Files/libB/libB.dll
  - /trunk/A/lib/libB.dll
- Utilisation « sauvage »
  - Makefile : gcc -l C:/Program Files/libB/libB.dll
  - IDE : /trunk/A/lib/libB.dll

Problèmes

- Intégration d'une nouvelle plate-forme ?
- Dépendance de B avec C ?
- Nouvelle version de B ?

Gestion des dépendances

Avec Maven

- Utilisation des coordonnées Maven
- Gestion transitive
- Publication dans des dépôts

Gestion des dépendances

Avec Maven

- Utilisation des coordonnées Maven
- Gestion transitive
- Publication dans des dépôts

Solutions

- Intégration d'une nouvelle plate-forme ?
- Dépendance de `B` avec `C` ?
- Nouvelle version de `B` ?

Gestion des dépendances

Avec Maven

- Utilisation des coordonnées Maven
- Gestion transitive
- Publication dans des dépôts

Solutions

- Intégration d'une nouvelle plate-forme ? 
- Dépendance de `B` avec `C` ? 
- Nouvelle version de `B` ? 

Gestion des dépendances

```
<dependencies>
    <dependency>
        <groupId>commons-lang</groupId>
        <artifactId>commons-lang</artifactId>
        <version>2.5</version>
    </dependency>
    <dependency>
        <groupId>commons-io</groupId>
        <artifactId>commons-io</artifactId>
        <version>1.4</version>
    </dependency>
</dependencies>
```

Dépôt central Maven : http://repo1.maven.org/maven2/

home » org.hibernate » hibernate-core

Hibernate Core

The core functionality of Hibernate

tags:

Available versions

Version	Type	Download
4.0.0.Alpha3	alpha	Binary (3.8 MB)
4.0.0.Alpha2	alpha	Binary (3.4 MB)
4.0.0.Alpha1	alpha	Binary (3.3 MB)
3.6.4.Final	release	Binary (3.0 MB)
3.6.3.Final	release	Binary (3.0 MB)
3.6.2.Final	release	Binary (3.0 MB)
3.6.1.Final	release	Binary (3.0 MB)
3.6.0.Final	release	Binary (3.0 MB)
3.6.0.CR2	release candidate	Binary (3.0 MB)
3.6.0.CR1	release candidate	Binary (3.0 MB)
3.6.0.Beta4	beta	Binary (3.0 MB)
3.6.0.Beta3	beta	Binary (3.0 MB)
3.6.0.Beta2	beta	Binary (3.0 MB)
3.6.0.Beta1	beta	Binary (2.9 MB)
3.5.6-Final	release	Binary (2.5 MB)

home » org.hibernate » hibernate-core » 3.6.4.Final

Hibernate Core

The core functionality of Hibernate

Artifact	Download (JAR) **(3.0 MB)**
POM File	View
HomePage	
Organization	
Issue Tracker	

Apache Maven | Apache Ivy (Ant) | Grape (Groovy) | Apache Buildr | SBT (Scala)

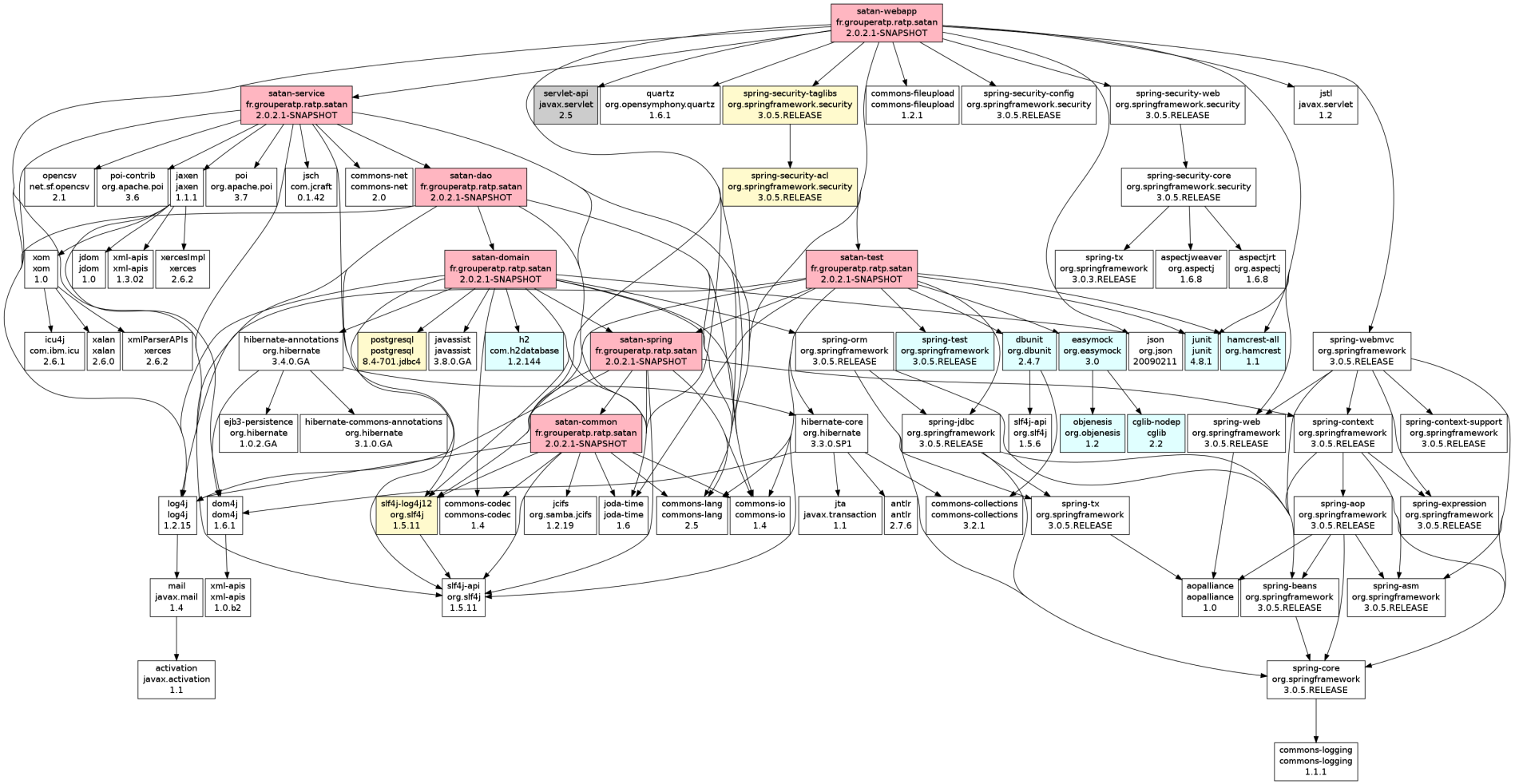
```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>3.6.4.Final</version>
</dependency>
```

This artifact depends on ...

Group	Artifact	Version
ant	ant	1.6.5
antlr	antlr	
cglib	cglib	
com.h2database	h2	

Comment faire ceci sans Maven ?

Projet SATAN ⇒ 77 dépendances, 9 niveaux de transition



SATAN webapp

Cycle de vie d'un projet

Constat fait par les concepteurs de Maven :

- Toujours les mêmes buts
- dans le même ordre
- sur chaque projet
  - Récupérer les dépendances
  - Compiler
  - Exécuter les tests unitaires
  - Générer l'assembly
- Mais aucun outillage. . .

Cycle de vie d'un projet

Constat fait par les concepteurs de Maven :

- Toujours les mêmes buts
- dans le même ordre
- sur chaque projet
  - Récupérer les dépendances
  - Compiler
  - Exécuter les tests unitaires
  - Générer l'assembly
- Mais aucun outillage. . .

Cycle de vie d'un projet

Constat fait par les concepteurs de Maven :

- Toujours les mêmes buts
- dans le même ordre
- sur chaque projet
  - Récupérer les dépendances
  - Compiler
  - Exécuter les tests unitaires
  - Générer l'assembly
- Mais aucun outillage...

Cycle de vie d'un projet

Constat fait par les concepteurs de Maven :

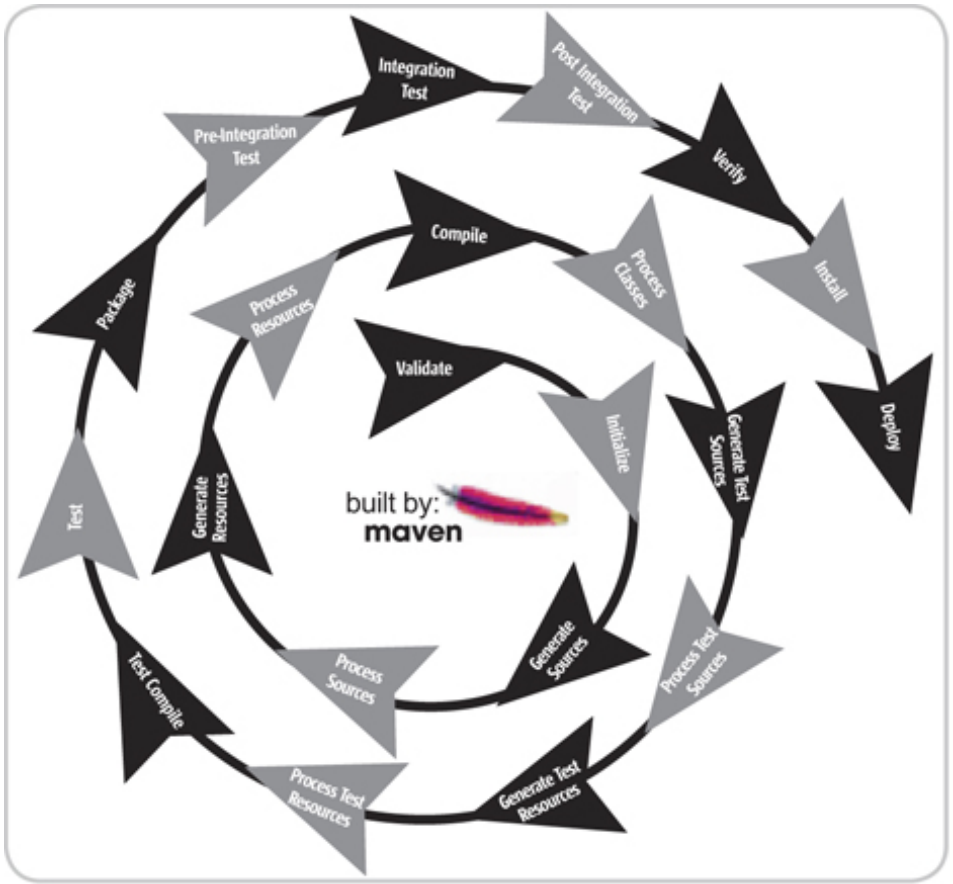
- Toujours les mêmes buts
- dans le même ordre
- sur chaque projet
  - Récupérer les dépendances
  - Compiler
  - Exécuter les tests unitaires
  - Générer l'assembly
- Mais aucun outillage...

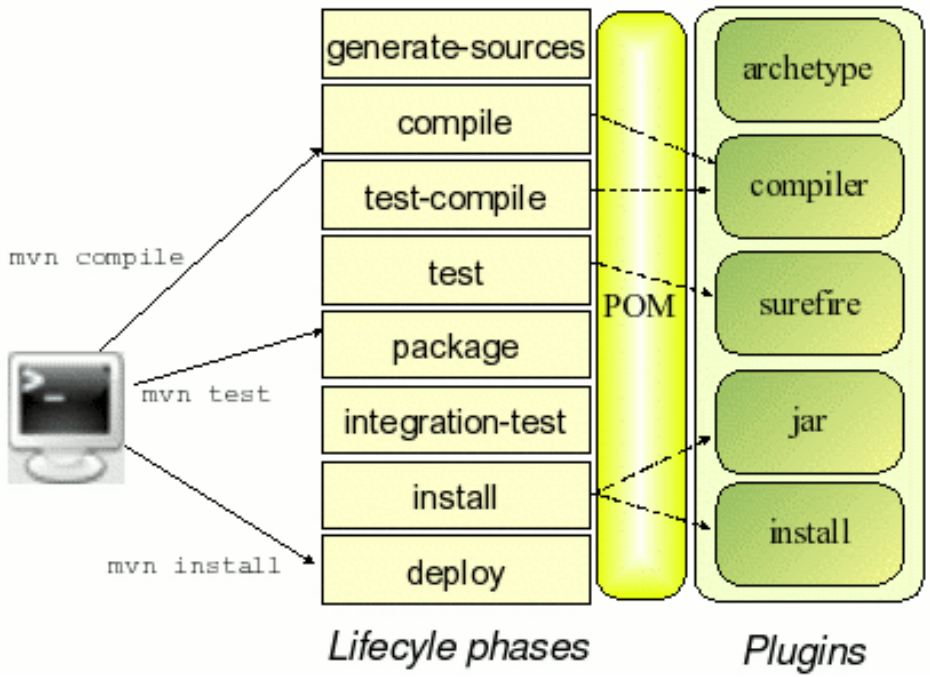
Solution :

Cycle de vie standardisé

Cycle de vie d'un projet

Cycle de vie de référence





Démonstration

Création d'un projet Maven

Outillage

Outillage non nécessaire et pourtant totalement indispensable :

- Rapport de tests unitaires
- Couverture de code des tests unitaires
- Javadoc
- Détection de bugs
- Respect des normes de codage
- Intégration continue
- Livraison
- Bugtracker
- Site web
- ...

Outillage

Outillage non nécessaire et pourtant totalement indispensable :

- Rapport de tests unitaires
- Couverture de code des tests unitaires
- Javadoc
- Détection de bugs
- Respect des normes de codage
- Intégration continue
- Livraison
- Bugtracker
- Site web
- ...

Tout est centralisé dans Maven

Intégration outils

```
<scm>
    <connection>
        scm:svn:http://dingo/svn/RATPSatan/trunk/satan-parent/
    </connection>
</scm>
<ciManagement>
    <system>Jenkins</system>
    <url>
        http://ci-linux/jenkins/job/RATP_SATAN/
    </url>
</ciManagement>
<issueManagement>
    <system>Mantis</system>
    <url>
        http://bugtracker/
    </url>
</issueManagement>
```

Intégration outils

```
<plugins>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
    </plugin>
    <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>cobertura-maven-plugin</artifactId>
    </plugin>
    <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>findbugs-maven-plugin</artifactId>
    </plugin>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-pmd-plugin</artifactId>
    </plugin>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-site-plugin</artifactId>
    </plugin>
</plugins>
```

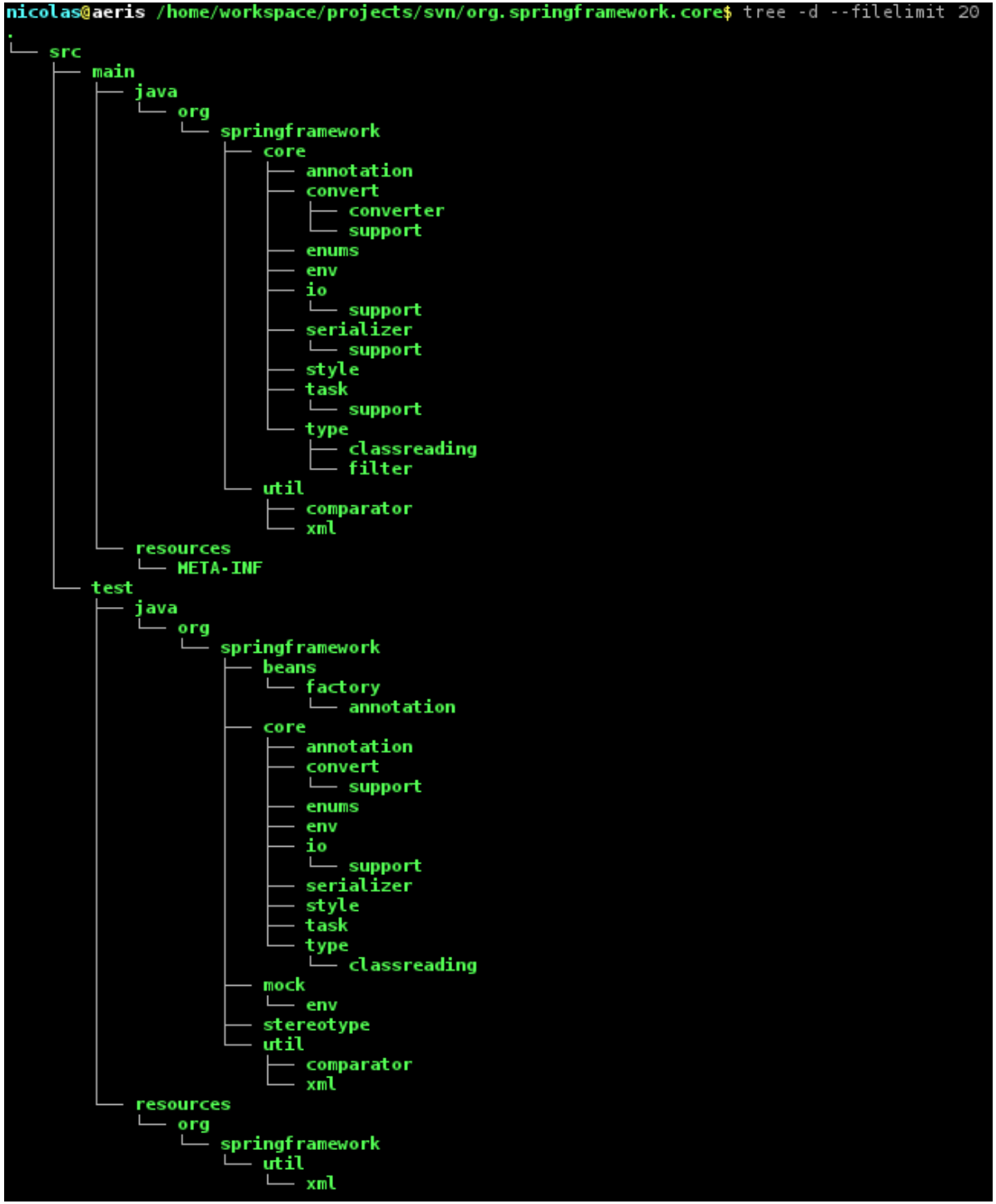
Démonstration

Génération site projet

Release

Remplit ses objectifs ?

```
nicolas@aeris /home/workspace/projects/github/hibernate/hibernate-core$ tree -d --filelimit 20
.
└── src
    ├── main
    │   ├── antlr
    │   ├── java
    │   │   └── org
    │   │       └── hibernate [39 entries exceeds filelimit, not opening dir]
    │   ├── javadoc
    │   │   └── images
    │   └── resources
    │       └── org
    │           └── hibernate
    │               └── ejb
    └── test
        ├── java
        │   └── org
        │       └── hibernate
        │           ├── cache
        │           ├── connection
        │           ├── dialect
        │           │   └── resolver
        │           ├── engine
        │           │   └── query
        │           ├── id
        │           │   ├── enhanced
        │           │   └── uuid
        │           ├── jdbc
        │           │   └── util
        │           ├── jmx
        │           ├── property
        │           ├── sql
        │           ├── subclassProxyInterface
        │           ├── type
        │           │   └── descriptor
        │           │       ├── java
        │           │       └── sql
        │           └── util
        └── resources
```


nicolas@aeris /home/workspace/projects/svn/org.springframework.core$ tree -d --filelimit 20
.
└── src
    ├── main
    │   ├── java
    │   │   └── org
    │   │   └── springframework
    │   │   ├── core
    │   │   │   ├── annotation
    │   │   │   ├── convert
    │   │   │   │   ├── converter
    │   │   │   │   └── support
    │   │   │   ├── enums
    │   │   │   ├── env
    │   │   │   ├── io
    │   │   │   │   └── support
    │   │   │   ├── serializer
    │   │   │   │   └── support
    │   │   │   ├── style
    │   │   │   ├── task
    │   │   │   │   └── support
    │   │   │   ├── type
    │   │   │   │   ├── classreading
    │   │   │   │   └── filter
    │   │   │   └── util
    │   │   │   ├── comparator
    │   │   │   └── xml
    │   │   └── resources
    │   │   └── META-INF
    │   └── test
    │   ├── java
    │   │   └── org
    │   │   └── springframework
    │   │   ├── beans
    │   │   │   └── factory
    │   │   │   └── annotation
    │   │   ├── core
    │   │   │   ├── annotation
    │   │   │   ├── convert
    │   │   │   │   └── support
    │   │   │   ├── enums
    │   │   │   ├── env
    │   │   │   ├── io
    │   │   │   │   └── support
    │   │   │   ├── serializer
    │   │   │   ├── style
    │   │   │   ├── task
    │   │   │   ├── type
    │   │   │   │   └── classreading
    │   │   │   ├── mock
    │   │   │   │   └── env
    │   │   │   ├── stereotype
    │   │   │   └── util
    │   │   │   ├── comparator
    │   │   │   └── xml
    │   │   └── resources
    │   │   └── org
    │   │   └── springframework
    │   │   └── util
    │   │   └── xml


nicolas@aeris /home/workspace/projects/svn/maven3/maven-core$ tree -dL 7 --filelimit 6
.
└── src
    ├── main
    │   ├── java
    │   │   └── org
    │   │   └── apache
    │   │   └── maven [16 entries exceeds filelimit, not opening dir]
    │   ├── mdo
    │   └── resources
    │   ├── META-INF
    │   │   └── plexus
    │   └── org
    │   └── apache
    │   └── maven
    │   ├── messages
    │   └── project
    ├── site
    │   ├── apt
    │   │   └── scripting-support
    │   └── resources
    │   └── design
    └── test
        ├── java
        │   └── org
        │   └── apache
        │   └── maven [9 entries exceeds filelimit, not opening dir]
        ├── projects
        │   ├── lifecycle-executor
        │   │   ├── mojo-configuration
        │   │   ├── project-basic
        │   │   ├── project-with-additional-lifecycle-elements
        │   │   │   └── src
        │   │   │   ├── main
        │   │   │   └── test
        │   │   ├── project-with-inheritance
        │   │   ├── project-with-multiple-executions
        │   │   │   └── src
        │   │   │   └── main
        │   │   └── project-with-plugin-level-configuration-only
        │   │   └── src
        │   ├── lifecycle-listener
        │   │   └── lifecycle-listener-dependency-injection
        │   ├── plugin-manager
        │   │   ├── mng-5003-plugin-realm-cache
        │   │   ├── project-contributing-system-scope-plugin-dep
        │   │   ├── project-with-inheritance
        │   │   └── project-with-plugin-classpath-ordering
        │   │   └── sub
        │   │   └── repo
        │   ├── project-builder
        │   │   ├── it0063
        │   │   │   └── jdk
        │   │   │   ├── jre
        │   │   │   └── lib
        │   │   └── mng-3023
        │   │   ├── consumer
        │   │   └── dependency
        │   └── project-dependencies-resolver
        │   ├── it0063
        │   │   └── jdk
        │   │   ├── jre
        │   │   └── lib
        │   └── project-with-exclusions
        └── remote-repo
            └── org
                ├── apache
                │   └── maven
                │   ├── its
                │   ├── maven
                │   ├── maven-plugin-api
                │   └── plugins

Built by: maven

Tomcat Maven Plugin

The Tomcat Maven Plugin provides goals to manipulate WAR projects within the Tomcat servlet container.

Goals Overview

The goals for this plugin come in two categories:

- Goals to manipulate deployed projects within Tomcat
- Goals to obtain information from Tomcat

Usage

Instructions on how to use the Tomcat Maven Plugin can be found on the usage page.

Examples

To provide you with better understanding of some usages of the Tomcat Maven Plugin, you can take a look into the following examples:

- Adding System Properties
- How to deploy projects to Tomcat

The Apache POI Project
http://poi.apache.org

Apache POI - the Java API for Microsoft Documents

6 June 2011 - POI 3.8 beta 3 available

The Apache POI team is pleased to announce the release of 3.8 beta 3. This includes a large number of bug fixes and enhancements.

A full list of changes is available in the change log. People interested should also follow the dev mailing list to track further progress.

See the downloads page for more details.

29 October 2010 - POI 3.7 available

The Apache POI team is pleased to announce the release of 3.7. This includes a large number of bug fixes, and some enhancements (especially text extraction). See the full release notes for more details.

A full list of changes is available in the change log. People interested should also follow the dev mailing list to track further progress.

See the downloads page for more details.

Mission Statement

The Apache POI Project's mission is to create and maintain Java APIs for manipulating various file formats based upon the Office Open XML standards (OOXML) and Microsoft's OLE 2 Compound Document format (OLE2). In short, you can read and write MS Excel files using Java. In addition, you can read and write MS Word and MS PowerPoint files using Java. Apache POI is your Java Excel solution (for Excel 97-2008). We have a complete API for porting other OOXML and OLE2 formats and welcome others to participate.

OLE2 files include most Microsoft Office files such as XLS, DOC, and PPT as well as MFC serialization API based file formats. The project provides APIs for the OLE2 Filesystem (POIFS) and OLE2 Document Properties (HPSF).

Office OpenXML Format is the new standards based XML file format found in Microsoft Office 2007 and 2008. This includes XLSX, DOCX and PPTX. The project provides a low level API to support the Open Packaging Conventions using openxml4j.

For each MS Office application there exists a component module that attempts to provide a common high level Java api to both OLE2 and OOXML document formats. This is most developed for Excel workbooks (SS=HSSF+XSSF). Work is

Main
Welcome

Get Maven
Download
Release Notes (3.0.3)
Release Notes (2.2.1)
Release Notes (2.0.11)
License

IDE Integration
Eclipse
Netbeans

About Maven
What is Maven?
Features
FAQ (official)
FAQ (unofficial)
Powered By

Documentation
Maven Plugins
Index (category)
Running Maven
User Centre
Plugin Developer Centre
Maven Repository Centre
Maven Developer Centre
Books and Resources
Wiki

Community
Community Overview

Welcome to Apache Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

If you think that Maven could help your project, you can find out more information about in the "About Maven" section of the navigation. This includes an in-depth description of what Maven is, a list of some of its main features, and a set of frequently asked questions about what Maven is.

Learning about Maven

This site is separated into the following sections, depending on how you'd like to use Maven:

- **Run Maven**
  Information for those needing to build a project that uses Maven
- **Use Maven**
  Information for those wanting to use Maven to build their project, including a "10 minute test" that gives a practical overview of Maven's main features in just 10 minutes
- **Write Maven Plugins**
  Information for those who may or may not be using Maven, but want to provide a plugin for shared functionality or to accompany their own product or toolset
- **Improve the Maven Repository**
  Information for those who may or may not use, but are interested in getting project metadata into the repository
- **Develop Maven**
  Information for those who are currently developers, or who are interested in contributing to the Maven project itself

Search Maven Sites
Search

Get Maven 3.0.3
Released: 3 March 2011

Maven 3.0.3
Release Notes, System Requirements, Installation Instructions

ApacheCon NA 2011

Get Maven Ant Tasks
Released: 22 Jul 2010

Maven Tasks for Ant 2.1.1
Release Notes, Documentation

Looking for Artifacts?

Search Central and other Public Repositories.

Looking for Repository Managers?

Scope des dépendances

Toutes les dépendances n'ont pas le même usage :

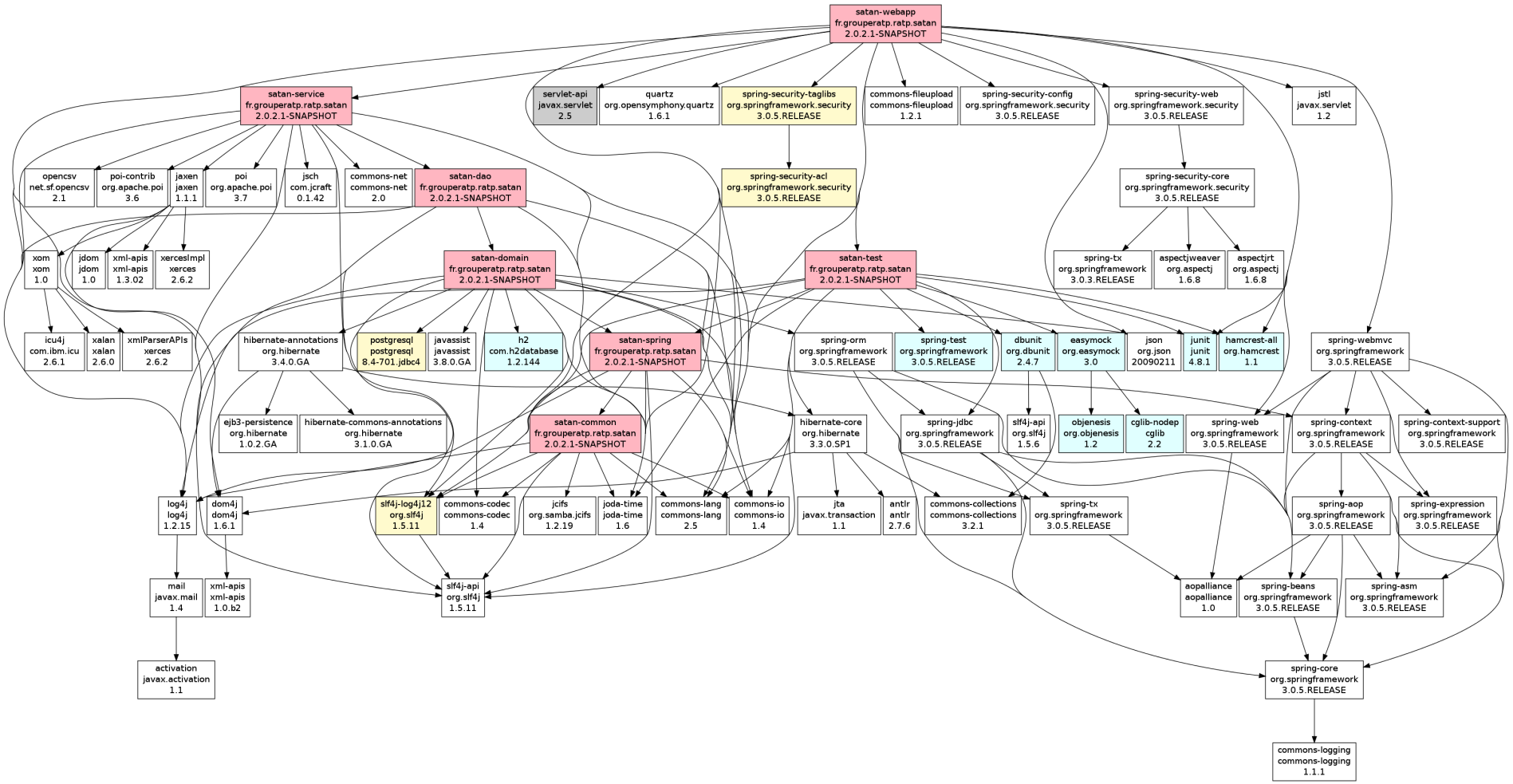
- L'application finale ne doit pas contenir les dépendances de tests unitaires
- Des dépendances peuvent être fournies par l'environnement d'exécution (cas des conteneurs de servlet et des serveurs d'application)
- Des dépendances ne sont nécessaires qu'à l'exécution et non à la compilation
- ...

Scope des dépendances

6 scopes de dépendances disponibles :

- `compile` :
  compilation, test, execution... (par défaut)
- `provided` :
  fournie par l'environnement d'exécution
- `runtime` :
  nécessaire uniquement à l'exécution
- `test` :
  nécessaire uniquement pour les tests unitaires
- `system` :
  comme `provided` mais gérer manuellement (rarement utilisé)
- `import` :
  importe les dépendances d'une dépendance POM (rarement utilisé)

Scope des dépendances



SATAN webapp

Héritage / Aggrégation

Maven permet la composition des POM de 2 manières :

- Héritage :
  un POM hérite des caratéristiques du POM parent
  composition ascendante
- Aggrégation :
  un POM va gérer X sous-POM
  composition descendante ($\sim$ sous-projets)

Exemples

- Je dois utiliser la configuration imposée par mon entreprise $\Rightarrow$ héritage
- Je dois développer une application 3-tiers $\Rightarrow$ aggrégation
- Je dois développer une application client/server $\Rightarrow$ aggrégation

Héritage / Aggrégation

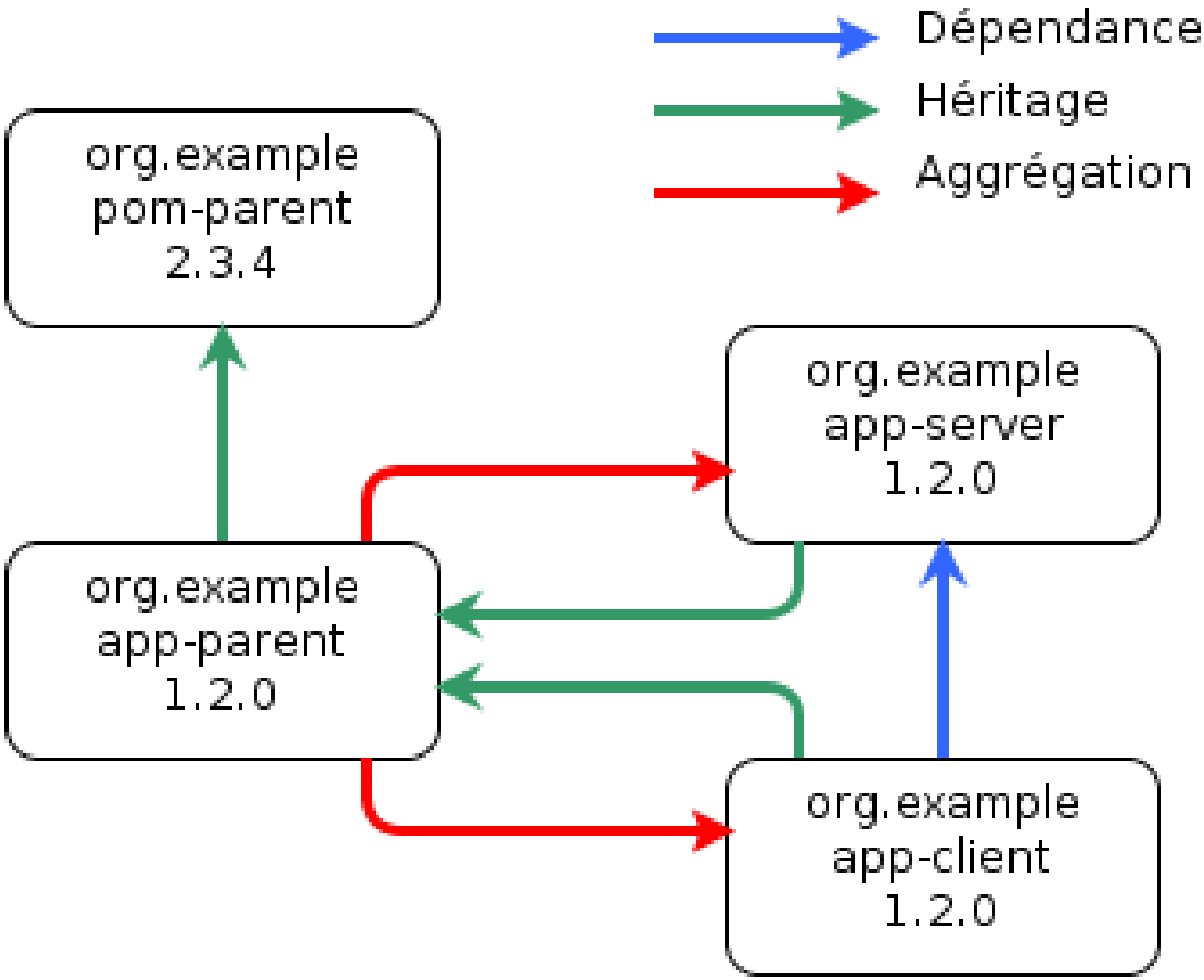
Maven permet la composition des POM de 2 manières :

- Héritage :
  un POM hérite des caratéristiques du POM parent
  composition ascendante
- Aggrégation :
  un POM va gérer X sous-POM
  composition descendante ($\sim$ sous-projets)

Exemples

- Je dois utiliser la configuration imposée par mon entreprise $\Rightarrow$ héritage
- Je dois développer une application 3-tiers $\Rightarrow$ aggrégation
- Je dois développer une application client/server $\Rightarrow$ aggrégation

Héritage / Aggrégation



Héritage / Aggrégation

```
<project>
    <groupId>org.example</groupId>
    <artifactId>pom-parent</artifactId>
    <version>2.3.4</version>
    <packaging>pom</packaging>
</project>
```

Héritage / Aggrégation

```
<project>
    <artifactId>app-parent</artifactId>
    <version>1.2.0</version>
    <packaging>pom</packaging>

    <parent>
        <groupId>org.example</groupId>
        <artifactId>pom-parent</artifactId>
        <version>2.3.4</version>
    </parent>

    <modules>
        <module>app-server</module>
        <module>app-client</module>
    </modules>
</project>
```

Héritage / Aggrégation

```
<project>
    <artifactId>app-client</artifactId>

    <parent>
        <groupId>org.example</groupId>
        <artifactId>app-parent</artifactId>
        <version>1.2.0</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>${project.groupId}</groupId>
            <artifactId>app-server</artifactId>
            <version>${project.version}</version>
        </dependency>
    </dependencies>
</project>
```

Aggrégation de dépôts

Des dépôts publics :

- Maven « centrale » : `http://repo1.maven.org/maven2/`
- Codehaus : `http://repository.codehaus.org/`
- Java (Oracle) : `http://download.java.net/maven/2/`
- ...

Des dépôts privés :

- Releases
- Snapshots

Des dépôts locaux

- `~/.m2`

Aggrégation de dépôts

Des dépôts publics :

- Maven « centrale » : `http://repo1.maven.org/maven2/`
- Codehaus : `http://repository.codehaus.org/`
- Java (Oracle) : `http://download.java.net/maven/2/`
- ...

Des dépôts privés :

- Releases
- Snapshots

Des dépôts locaux

- `~/.m2`

Comment aggréger tous les dépôts pour le développeur final ?

Aggrégation de dépôts

Des dépôts publics :

- Maven « centrale » : `http://repo1.maven.org/maven2/`
- Codehaus : `http://repository.codehaus.org/`
- Java (Oracle) : `http://download.java.net/maven/2/`
- ...

Des dépôts privés :

- Releases
- Snapshots

Des dépôts locaux

- `~/.m2`

Comment aggréger tous les dépôts pour le développeur final ?

⇒ Récursivité + proxy

Aggrégation de dépôts

3 possibilités :

- Utiliser la configuration par défaut = uniquement le dépôt « centrale »
  - Limitation des librairies disponibles
  - Publication des releases ?
- Ajouter des dépôts dans le POM
  - Perte de performances (404 à répétition)
  - Publication des releases ?
- Utiliser un proxy Maven (Nexus, Archiva, Artifactory)
  - Centralisation des développements
  - Publication des releases
  - Contrôle des dépendances (Release Manager)

Aggrégation de dépôts

3 possibilités :

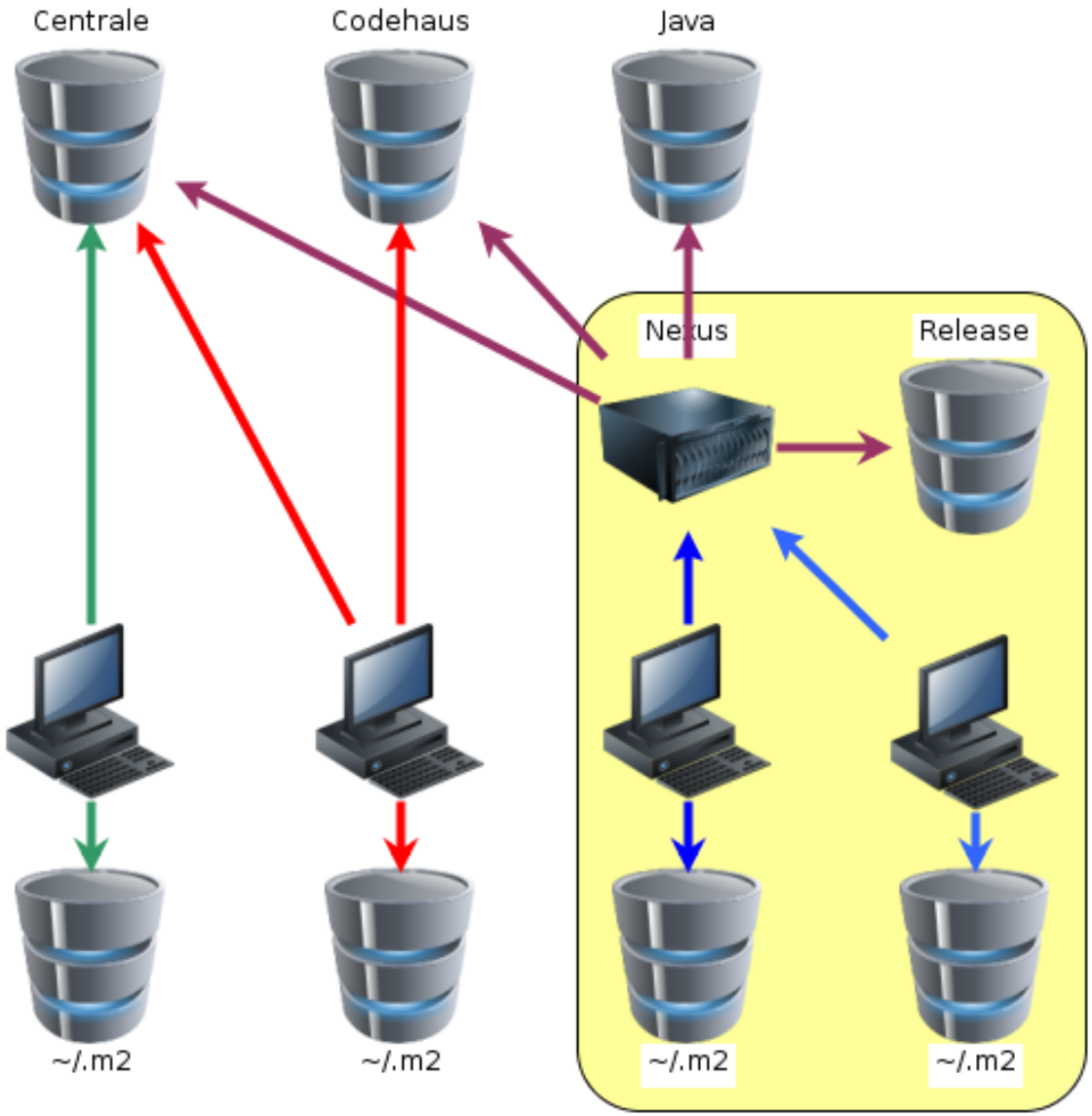
- Utiliser la configuration par défaut = uniquement le dépôt « centrale »
  - Limitation des librairies disponibles
  - Publication des releases ?
- Ajouter des dépôts dans le POM
  - Perte de performances (404 à répétition)
  - Publication des releases ?
- Utiliser un proxy Maven (Nexus, Archiva, Artifactory)
  - Centralisation des développements
  - Publication des releases
  - Contrôle des dépendances (Release Manager)

Aggrégation de dépôts

3 possibilités :

- Utiliser la configuration par défaut = uniquement le dépôt « centrale »
  - Limitation des librairies disponibles
  - Publication des releases ?
- Ajouter des dépôts dans le POM
  - Perte de performances (404 à répétition)
  - Publication des releases ?
- Utiliser un proxy Maven (Nexus, Archiva, Artifactory)
  - Centralisation des développements
  - Publication des releases
  - Contrôle des dépendances (Release Manager)

Aggrégation de dépôts



Démonstration

Nexus

Jenkins

Références
Documentation

Documentation

- Documentation officielle :
  `http://maven.apache.org/guides/index.html`
- Maven, the Complete Reference (306 pages) :
  `http://www.sonatype.com/books/mvnref-book/reference/public-book.html`
- Plugin M2Eclipse :
  `http://m2eclipse.sonatype.org/`
- Sites des plugins
- Recherche dans les dépôts :
  `http://mvnrepository.com/`

Questions ?