

Project Management: Make It Easy with Maven

Hoang Huu Hanh
(11.10.2004)

Contents

- What is Maven and How it works
- Installing and Using Maven
- A Simple Example
- Continuous Integration with Maven
- Customizing Maven

Contents

- What is Maven and How it works
- Installing and Using Maven
- A Simple Example
- Continuous Integration with Maven
- Customizing Maven

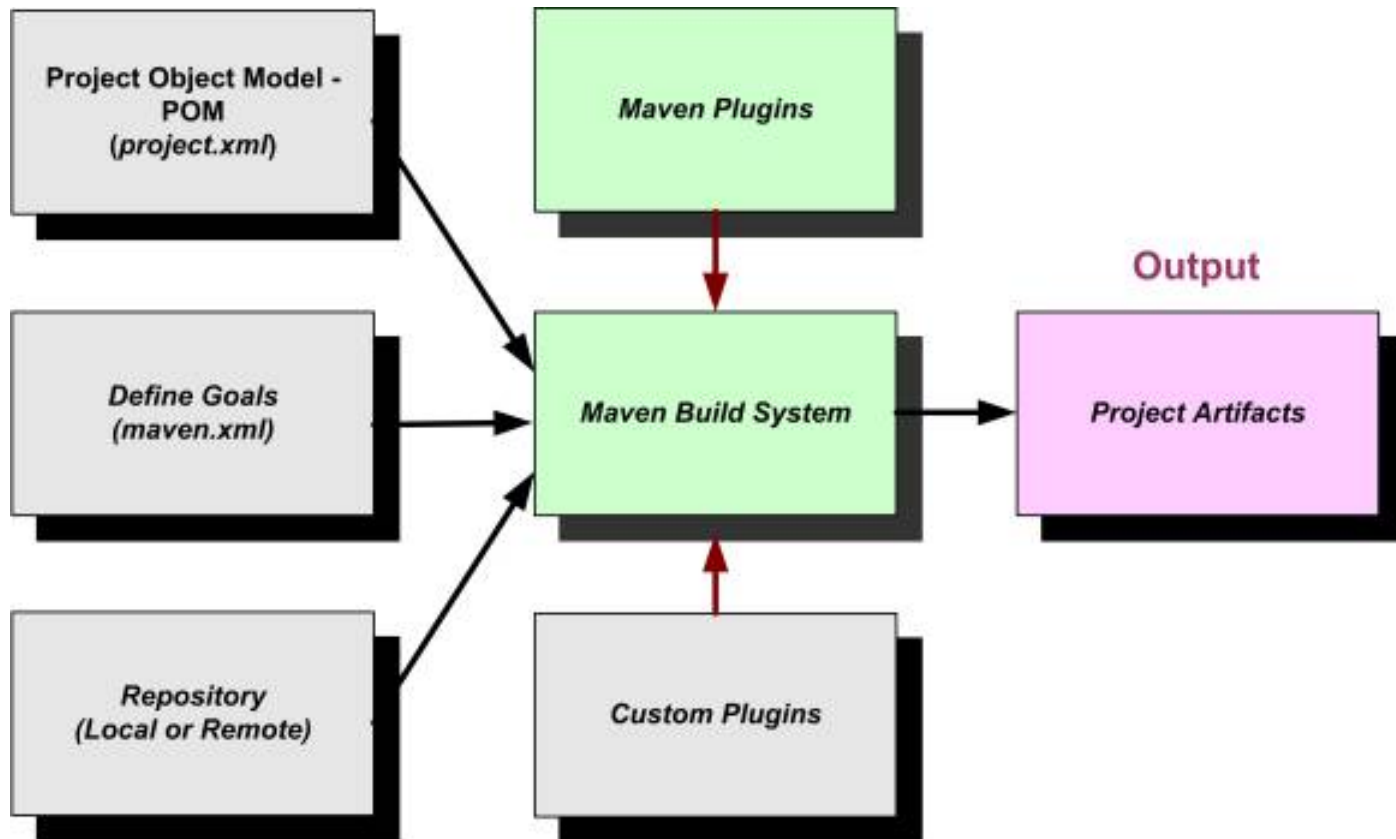
What is Maven?

- Maven is a
 - high-level
 - intelligent project management
 - build and development tool from the Apache project.
- Ant
 - Write the build script
 - Run the targets
- Maven
 - Describe the projects and configure plugins
 - Run existing plugins

Maven Basics

- The basic concept of Maven is **project (project.xml)**
- **Repository** holds the artifacts on which the project depends
- The unit of work in Maven is a **goal** (goal in Maven terms is similar to target in Ant). Maven provides prebuilt goals that will automatically:
 - Compile your codes
 - Create jar, war, ear, ejb files
 - Run unit tests
 - Run source code metrics and analysis
 - Create a report detailing violations of your team's coding standards.
 - Create a report of most recent CVS commits.
 - Create a report of CVS activity, pivoted by file and developer
 - Create HTML cross-referenced source code, and more.
- A **plugin** is a logical collection of goals written using Jelly – an XML based scripting language

Maven Build Elements



Project Object Model

- Project Object Model (POM) is described in the file *project.xml*
- *project.xml* (so-called Project Descriptor) is divided into four parts:
 - Project Management Section
 - Project Dependency Section
 - Project Build Section
 - Project Reports Section

project.xml

```
<?xml version="1.0"?>
<project>
  <pomVersion>3</pomVersion>
  <groupId>Sample-Maven-Project</groupId>
  <artifactId>sample-project</artifactId>
  <currentVersion>1.1</currentVersion>
  <name>Sample Maven Project</name>

  <!-- Project Management section goes here -->

  <!-- Project Dependency section goes here -->

  <!-- Project Build section goes here -->

  <!-- Project Reports section goes here -->

</project>
```

project.xml Basic Items

- **artifactId**: a unique identifier that is comprised of lowercase letters [a-z] where hyphens can be used:

```
<project>
  <groupId>bar</groupId>
  <artifactId>foo</artifactId>
  ...
</project>
```

- A project should have a group identifier. This group identifier is the basis for the name

```
<project>
  <groupId>bar</groupId>
  <artifactId>foo</artifactId>
  ...
</project>
```

- All artifacts published by a project should be based on the project's unique identifier and placed in a directory based on the project's group identifier. So for the above entry, assuming an artifact type of `jar`, we might have something that looks like the following:

```
repository
|
+-- bar
   |-- distribution
   |-- jar
       |-- foo-1.0.jar
       |-- foo-2.0.jar
```

Project Management Section in 'project.xml'

```
<organization>
  <name>IFS - TU Wien</name>
  <url>http://www.ifs.tuwien.ac.at</url>
  <logo>http:// www.ifs.tuwien.ac.at/logo.jpg</logo>
</organization>

<inceptionYear>2004</inceptionYear>
<package>sample.*</package>
<logo>http://www.ifs.tuwien.ac.at/project-logo.jpg</logo>
<description>Project description goes here</description>
<shortDescription>Short Description</shortDescription>
<url>http://www.ifs.tuwien.ac.at</url>
<issueTrackingUrl>http://...</issueTrackingUrl>
<siteAddress>http://...</siteAddress>
<siteDirectory>/etc/staging</siteDirectory>
<distributionDirectory>/etc/builds</distributionDirectory>
```

Project Management Section (2)

```
<repository>
  <connection>cvs:extssh:anon@nemesis.ifs.tuwien.ac.at:/proj
  </connection>
  <url>http://scm.ifs.tuwien.ac.at</url>
</repository>
<mailingLists>
  <mailingList>
    <name>Dev List</name>
    <subscribe>subscribe-dev@foobar.com</subscribe>
    <unsubscribe>unsubscribe-dev@foobar.com</unsubscribe>
  </mailingList>
  ...
</mailingLists>
<developers>
  <developer>
    <name>Hanh</name>
    <id>hhhanh</id>
    <email>hhhanh@ifs.tuwien.ac.at</email>
  </developer>
  ...
</developers>
```



Project Documentation

About SemanticLIFE-
Hello World
Downloads

▼ Project Info

Mailing Lists
Project Team
Dependencies

► Project Reports
Development Process



General Project Information

This document provides an overview of the various documents and links that are part of this project's general information. All of this content is automatically generated by [Maven](#) on behalf of the project.

Overview

Document	Description
Mailing Lists	This document provides subscription and archive information for this project's mailing lists.
Project Team	This document provides information on the members of this project. These are the individuals who have contributed to the project in one form or another.
Dependencies	This document lists the projects dependencies and provides information on each dependency.

Project Build Section in 'project.xml'

```
<build>
  <nagEmailAddress>hhhanh@if.tuwien.ac.at</nagEmailAddress>
  <sourceDirectory>${basedir}/src/java</sourceDirectory>
  <unitTestSourceDirectory>${basedir}/test/java
</unitTestSourceDirectory>
  <unitTest>
    <includes>
      <include>**/*Test.java</include>
    </includes>
  </unitTest>
  <resources>
    <resource>
      <directory>${basedir}/src/conf</directory>
      <includes>
        <include>*.properties</include>
      </includes>
    </resource>
  </resources>
</build>
```

Project Reports Section in 'project.xml'

<reports>

```
<report>maven-changes-plugin</report>  
<report>maven-jdepend-plugin</report>  
<report>maven-checkstyle-plugin</report>  
<report>maven-pmd-plugin</report>  
<report>maven-junit-report-plugin</report>  
<report>maven-clover-plugin</report>  
<report>maven-changelog-plugin</report>  
<report>maven-file-activity-plugin</report>  
<report>maven-developer-activity-plugin</report>  
<report>maven-file-activity-plugin</report>  
<report>maven-license-plugin</report>  
<report>maven-linkcheck-plugin</report>  
<report>maven-jxr-plugin</report>
```

</reports>

Project Documentation

About SemanticLIFE-

Hello World

Downloads

► Project Info

▼ **Project Reports**

Change Log

Checkstyle

Developer Activity

File Activity

Metrics

Source Xref 

Project License

Link Check Report

PMD Report

Task List


JavaDocs 

JavaDoc Report



Development Process 



Maven Generated Reports

This document provides an overview of the various reports that are automatically generated by [Maven](#) . Each report is briefly described below.

Overview

Document	Description
Change Log	Report on the source control changelog.
Checkstyle	Report on coding style conventions.
Developer Activity	Report on the amount of developer activity.
File Activity	Report on file activity.
Metrics	Report on source code metrics.
Source Xref 	A set of browsable cross-referenced sources.
Project License	Displays the primary license for the project.
Link Check Report	Report on the validity of all links in the documentation.
PMD Report	Verification of coding rules.
Task List	Report on tasks specified in the source code.
JavaDocs 	JavaDoc API documentation.
JavaDoc Report	Report on the generation of JavaDoc.

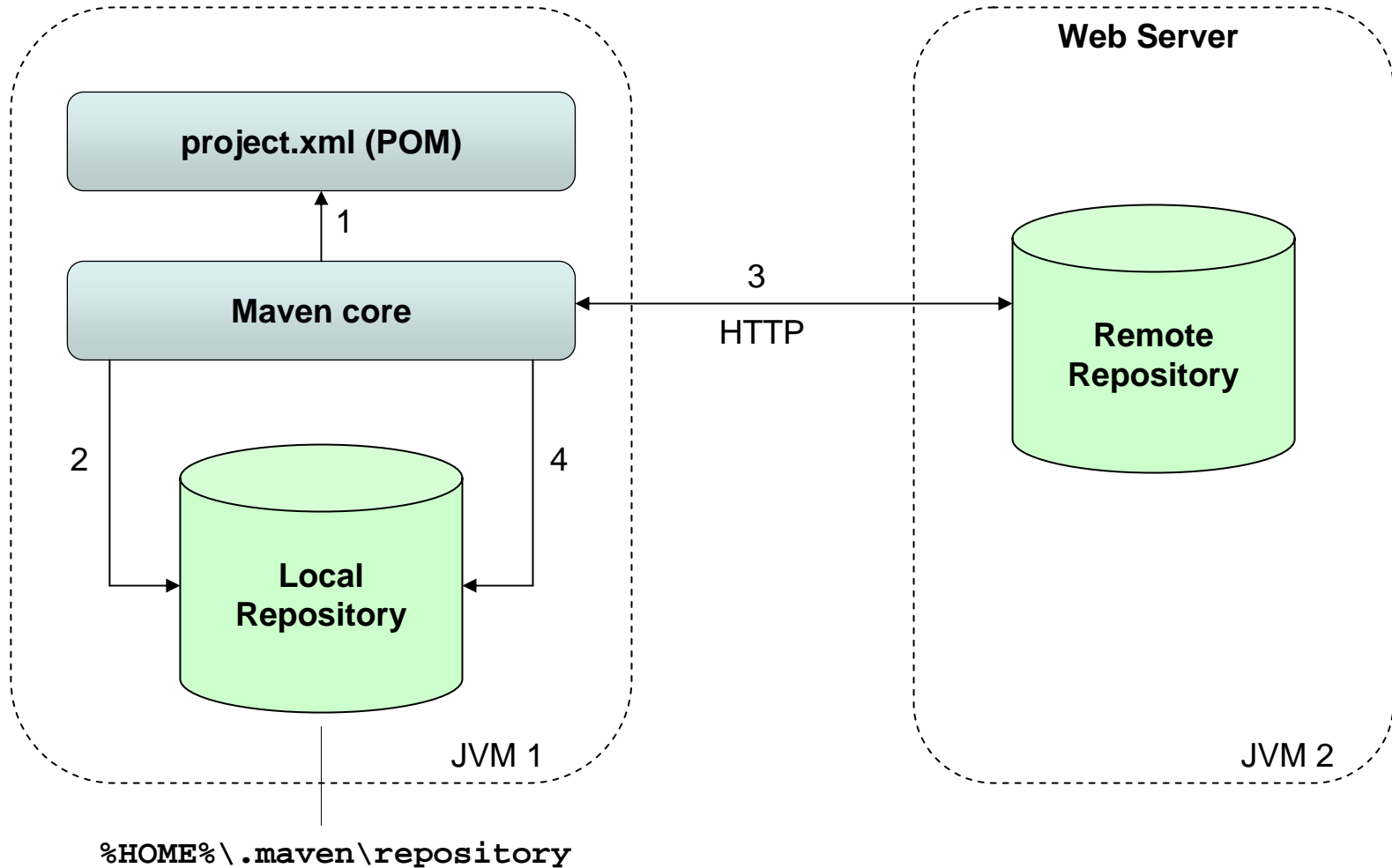
Project Dependency Section in 'project.xml'

```
<dependencies>
  <dependency>
    <groupId>BeanUtils</groupId>
    <artifactId>commons-beanutils</artifactId>
    <version>1.5</version>
  </dependency>
  <dependency>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>1.0.3</version>
  </dependency>
  <dependency>
    <groupId>castor</groupId>
    <artifactId>castor</artifactId>
    <version>0.9.4.3</version>
  </dependency>
</dependencies>
```

Project with Multiple Artifacts

```
<dependencies>
  <!-- A -->
  <dependency>
    <groupId>ant</groupId>
    <artifactId>ant</artifactId>
    <version>1.4.1</version> </dependency>
  <!-- B -->
  <dependency>
    <groupId>ant</groupId>
    <artifactId>ant-optional</artifactId>
    <version>1.4.1</version> </dependency>
  <!-- C -->
  <dependency>
    <groupId>ant</groupId>
    <artifactId>poorly-named</artifactId>
    <version>1.4.1</version>
  </dependency>
</dependencies>
```

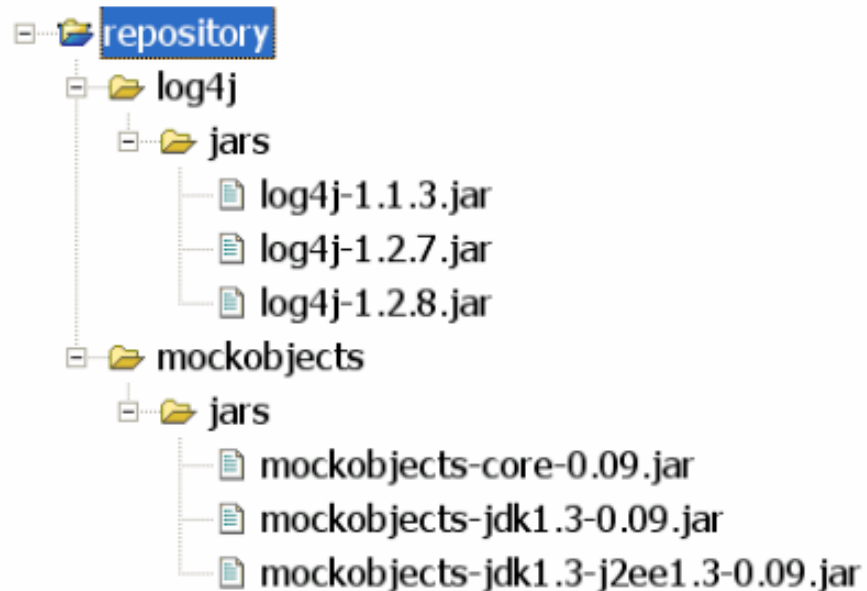
Maven Artifact Repositories (1/2)



Maven Artifact Repositories (2/2)

```
<project>
  [...]
  <dependencies>
    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.8</version>
      <type>jar</type>
    </dependency>
    [...]
  </dependencies>
  [...]
</project>
```

`%MAVEN_REPO%/`
`<groupId>/<type>s/<artifactId>-<version>.<type>`



Contents

- What is Maven and How it works
- **Installing and Using Maven**
- A Simple Example
- Continuous Integration with Maven
- Customizing Maven

Installing Maven (1/2)

- Download Maven from <http://maven.apache.org/start/download.html>
- Set `JAVA_HOME` environment variable to a JDK directory (i.e. the directory contains the `bin`, `demo`, `jre` and `lib` directories)
- Define `MAVEN_HOME` environment variable: is the directory where you unpacked the Maven install archive
- Create your **local repository**: running following commands:
 - UNIX:
`$MAVEN_HOME/bin/install_repo.sh $HOME/.maven/repository`
 - Windows:
`%MAVEN_HOME%\bin\install_repo.bat %HOME%/.maven/repository`
- Note: `%HOME%` = `%HOMEDRIVE%%HOMEPATH%`
(C:\Documents and Settings\)

Installing Maven (2/2)

- Verify: In the top-level directory of MAVEN_HOME, you should see the following structure:

```
.
| --bin
| --lib
| --plugins
```

- To confirm that you can start Maven, run:

```
maven -v
```

```

  ____  _
 / ___|| | | | Apache
| |___| | | | / ___| | | | v. 1.0
| |___| | | | / ___| | | | ~ intelligent projects ~
| |___| | | | / ___| | | |


```

Using Maven

- Maven command syntax

```
maven <plugin name>[:<goal name>]
```

- Some commands of Maven

- maven -g
- maven java:compile
- maven jar:install
- maven site:generate
- maven site:deploy

- Start a new project

```
maven -Dpackage=com.mycompany.app genapp
```

New Project

- When the command finishes executing you should have a complete project tree that looks something like the following:

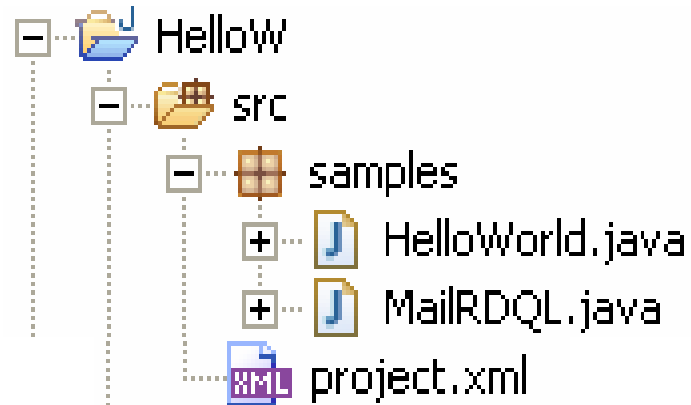
```
.
├-- project.properties
├-- project.xml
└-- src
    ├── conf
    │   └-- app.properties
    ├── java
    │   ├── com
    │   │   ├── mycompany
    │   │   │   └-- app
    │   │   │       └-- App.java
    └-- test
        ├── com
        │   ├── mycompany
        │   │   └-- app
        │   │       ├── AbstractTestCase.java
        │   │       ├── AppTest.java
        │   │       └-- NaughtyTest.java
```

Contents

- What is Maven and How it works
- Installing and Using Maven
- **A Simple Example**
- Continuous Integration with Maven
- Customizing Maven

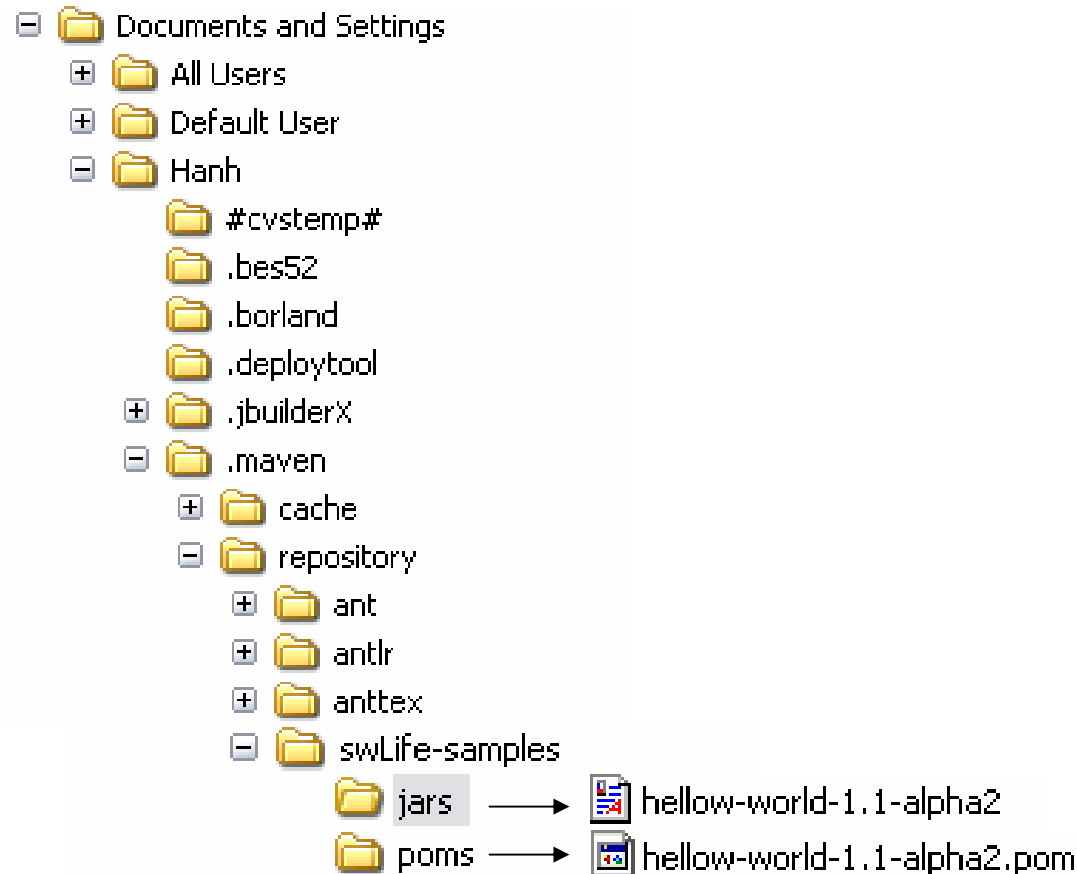
“Hello World!” Example

- Create a simple project having structure as follow



- See [project.xml](#)

Example (4)



Maven repository for “swLife-sample”

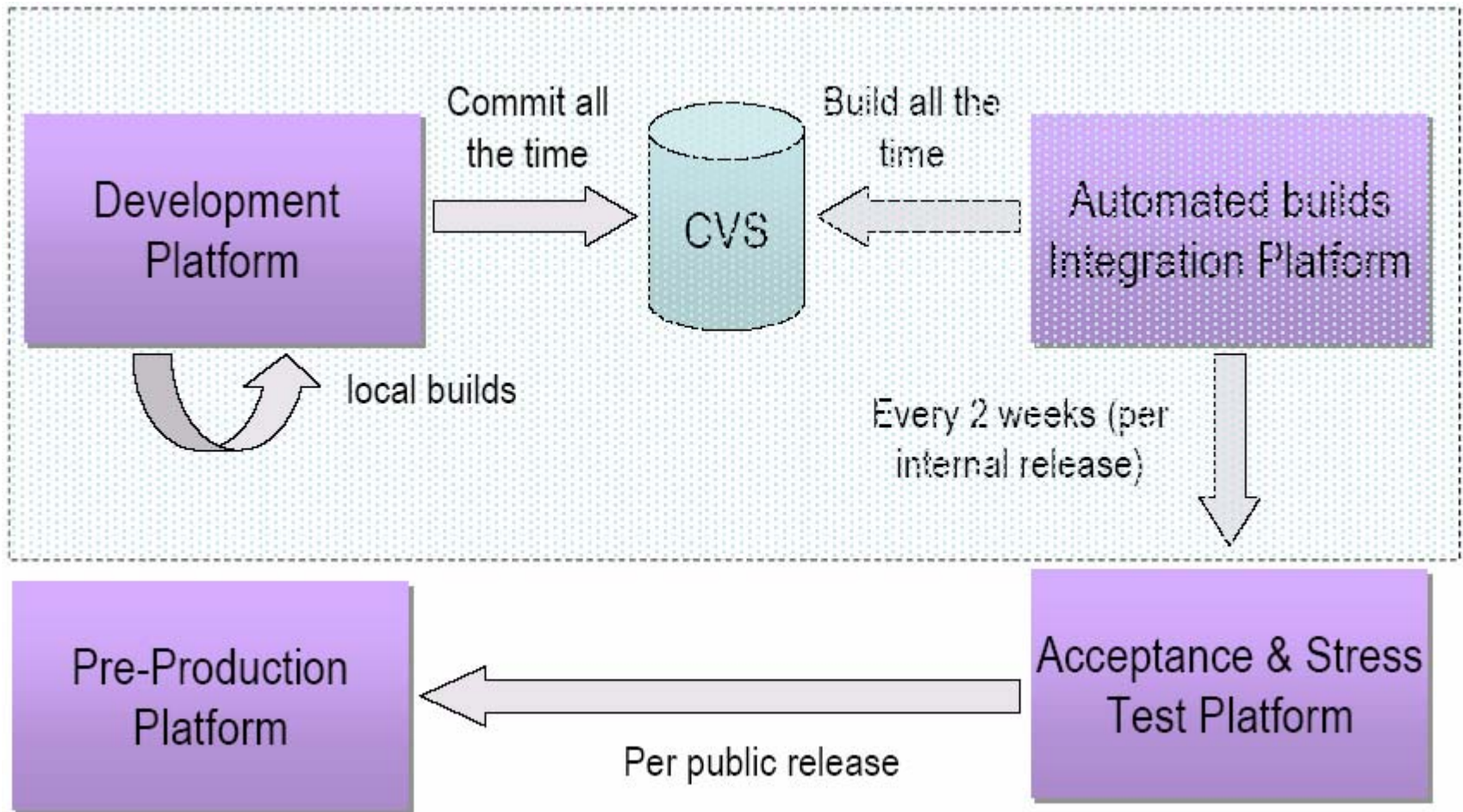
Example (5)

- Generate the lists of documents for project website: `maven site:generate`
 - Project info
 - Developer list
 - Mailing lists
 - Dependencies
 - Unit tests reports
 - Source code metrics,
 - Change logs
 - JavaDocs
 - Cross-referenced source code
- Deploy the website
 - `maven [-Dmaven.user=${username}] site:deploy`
- See at: <http://www.ifs.tuwien.ac.at/~hhhanh/semanticLIFE>

Contents

- What is Maven and How it works
- Installing and Using Maven
- A Simple Example
- **Continuous Integration with Maven**
- Customizing Maven

Continuous Integration



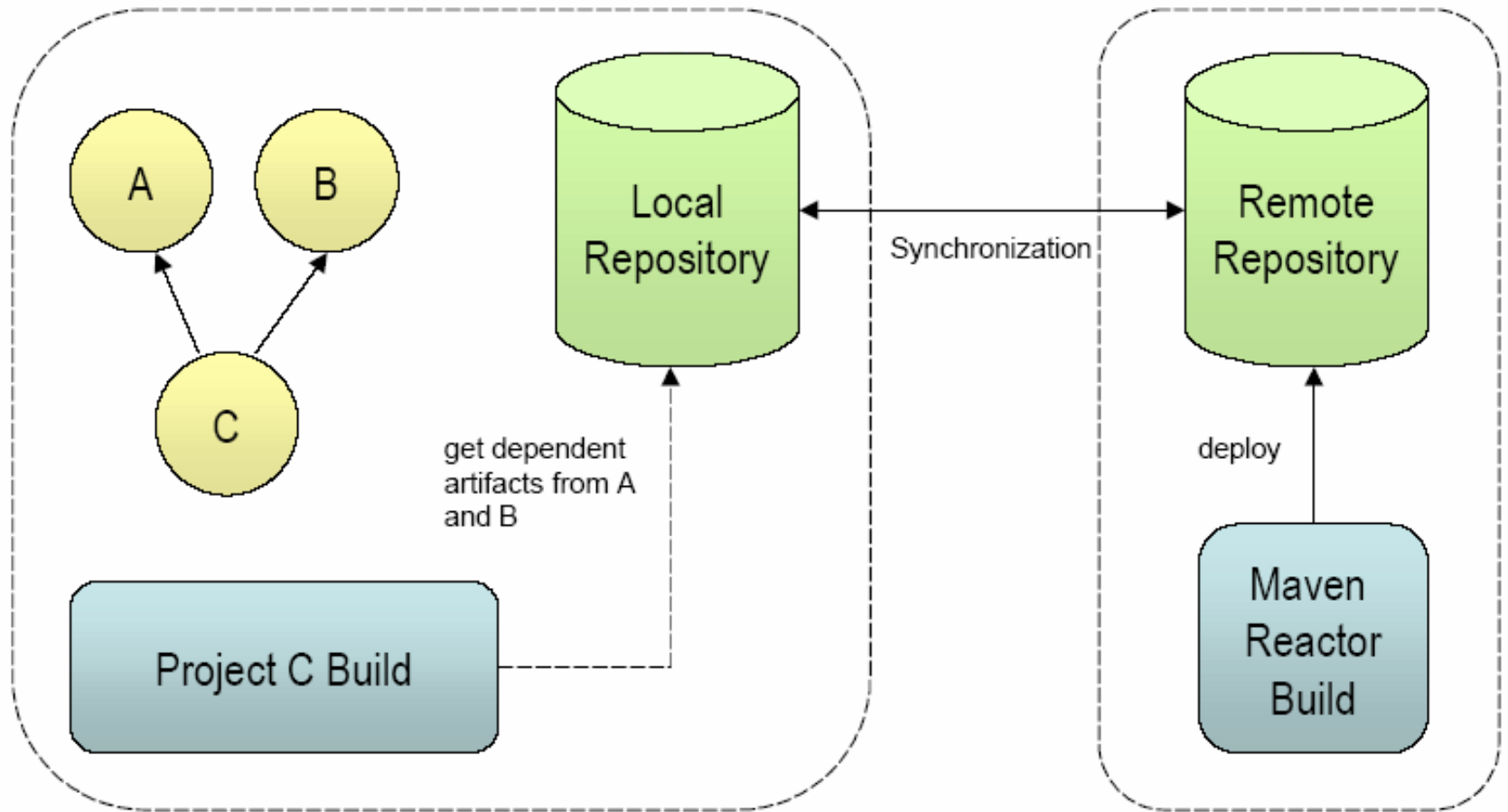
Continuous Integration

- Maven Reactor

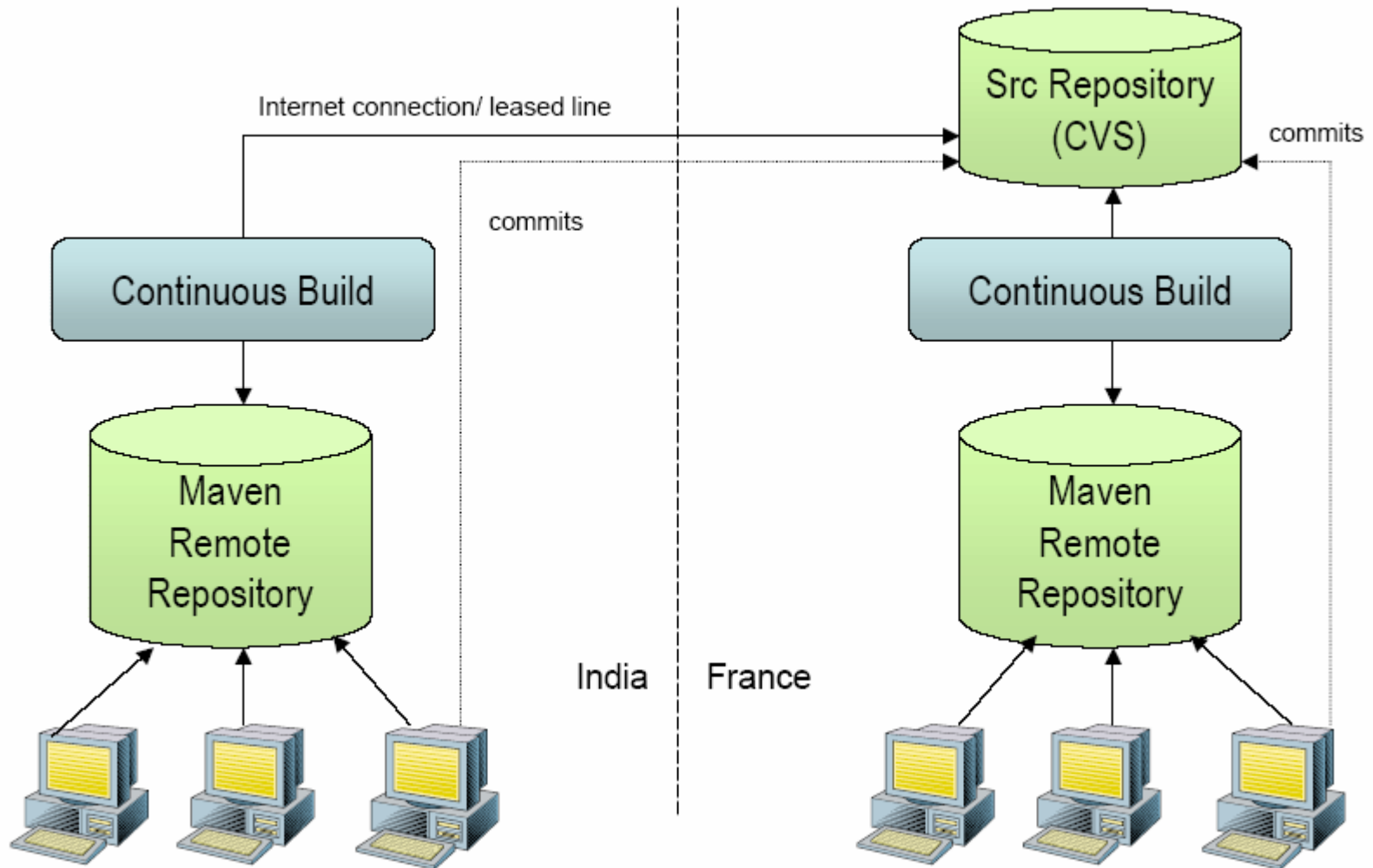
maven.xml

```
<goal name="everest:dist-all">
  <maven:reactor
    basedir="${basedir}"
    includes="**/${pattern}/**/project.xml"
    excludes="project.xml"
    goals="everest:dist"
    banner="Building"
    ignoreFailures="false"/>
</goal>
```

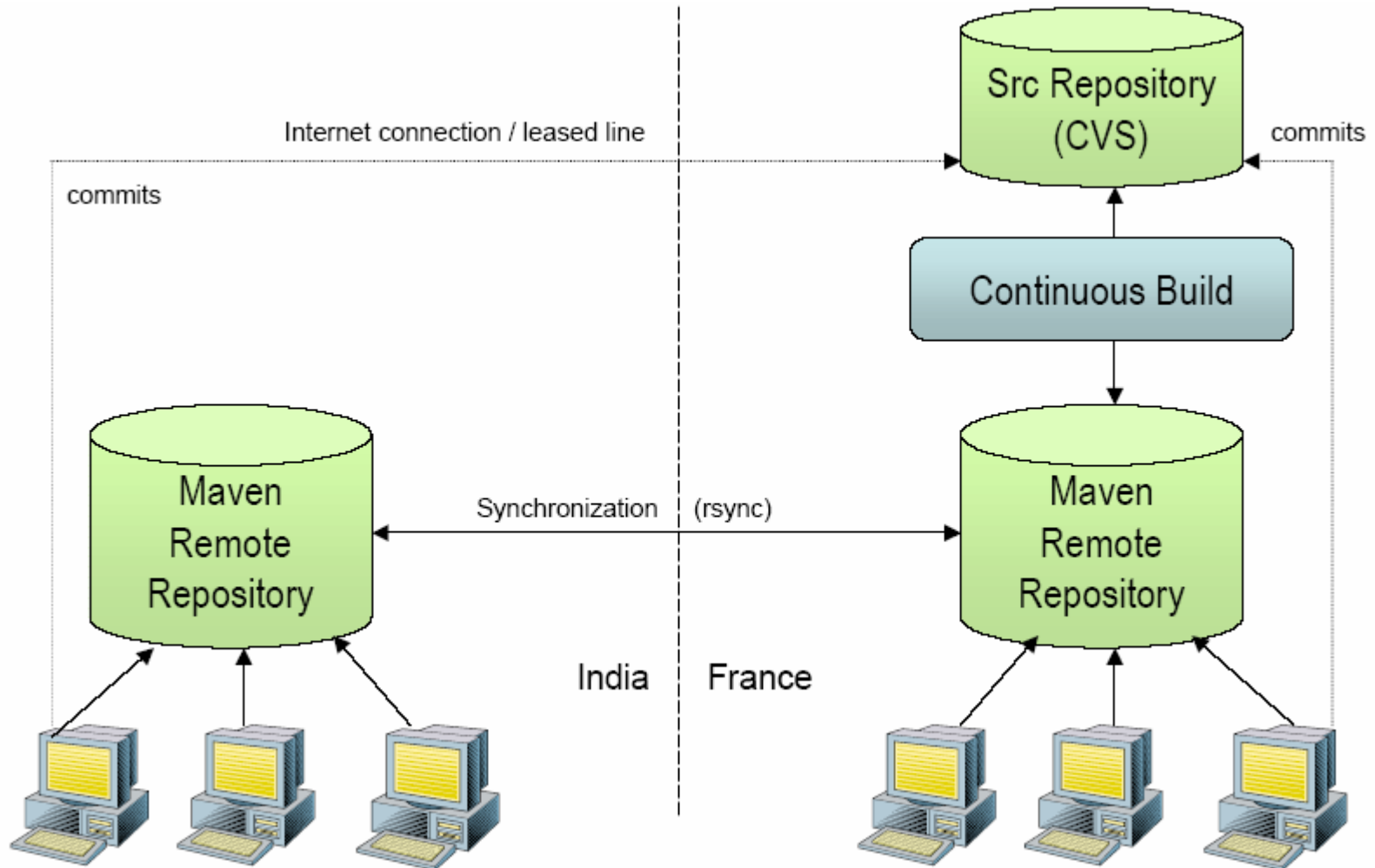
Setting up Maven in a Corporate environment



Multi-location Setup (1/2)



Multi-location Setup (2/2)



Contents

- What is Maven and How it works
- Installing and Using Maven
- A Simple Example
- Continuous Integration with Maven
- **Customizing Maven**

Customizing Maven

- First level of customization
 - project.properties/build.properties
 - For plugin configuration
- Second level of customization
 - create a maven.xml file
 - Containing new goals
 - Intercepting existing goals with <preGoal/> and <postGoal/>
 - factorize maven.xml by using a common inherited project
- Third level of customization
 - create plugins to share common build logic

maven.xml

- In this file, you can customize goals, define pre and pos goals

```
<project default="myGoal" xmlns:m="jelly:maven">
  <preGoal name="java:compile">
    <attainGoal name="xdoclet:webdoclet" />
  </preGoal>
  <goal name="myGoal">
    <attainGoal name="war:install" />
  </goal>
  <postGoal name="war:war">
    <mkdir dir="\${test.result.dir}" />
    <echo>Creating directories</echo>
  </postGoal>
</project>
```

Writing a Maven Plugins

plugin.jelly

```
<?xml version="1.0"?>

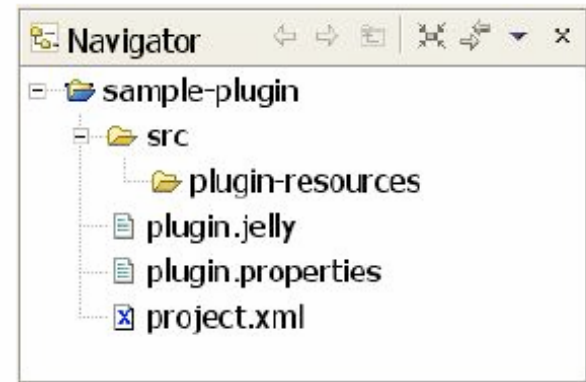
<project
  xmlns:j="jelly:core"
  xmlns:ant="jelly:ant">

  <goal name="hello"
    description="Send nice message">

    <ant:echo>Hello world!</ant:echo>

  </goal>

</project>
```



To build and deploy: "maven plugin:install"

Maven Tips

- Do not create too many subprojects
 - It is so easy to create a project with Maven...
 - We had 1500+ projects, brought back to 150+
 - A project = a public interface
 - The more you have the less stable you are
- Build every few hours
- Need strong commitment from Management
 - To tell that successful builds is all-important
- Use Snapshot jars
- Define external jars in top level inherited project
 - So that all projects use the same version
 - Better: use common project.properties
 - But not supported yet by Maven (soon)
- Share project specific build logic through
 - Top level maven.xml
 - Project-specific Plugins

Maven Pros & Cons

- Pros

- Quick to set up
- Best development practice enforcer
- Benefit from numerous plugins
- Nice features
 - Dependencies handling
 - Easy to set up a Continuous integration process

- Cons

- Not mature yet
 - Little documentation (and not always up to date)
 - Lots of little details missing (but progressing quickly)
 - Needs someone to monitor/participate to the Mailing List

That's all. Thank you!