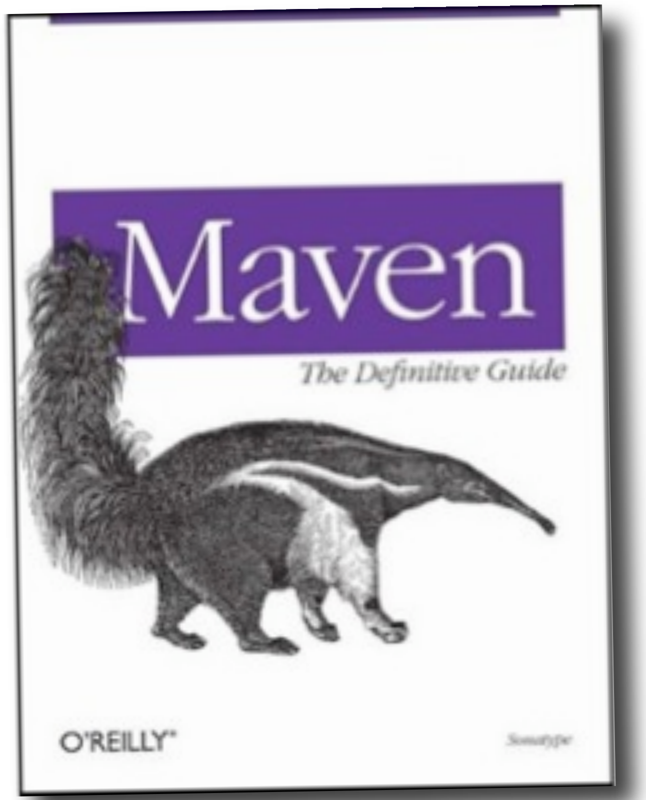
# Mastering Maven

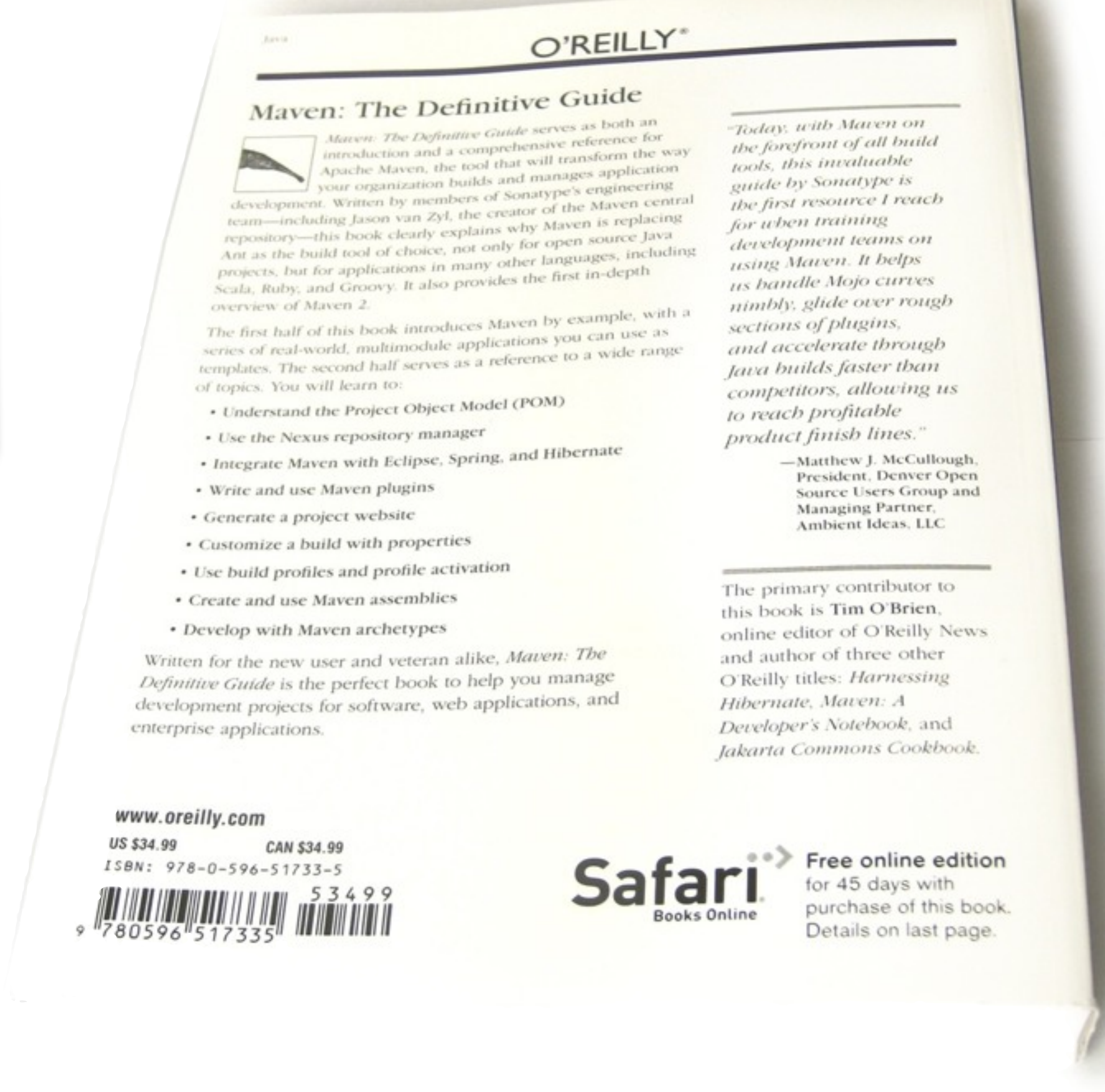
Increase your MoJo-tivity

by Matthew McCullough of Ambient Ideas, LLC

Hello
my name is
Matthew

AMbiENT [ idEAS ]





Java

O'REILLY

## Maven: The Definitive Guide

*Maven: The Definitive Guide* serves as both an introduction and a comprehensive reference for Apache Maven, the tool that will transform the way your organization builds and manages application development. Written by members of Sonatype's engineering team—including Jason van Zyl, the creator of the Maven central repository—this book clearly explains why Maven is replacing Ant as the build tool of choice, not only for open source Java projects, but for applications in many other languages, including Scala, Ruby, and Groovy. It also provides the first in-depth overview of Maven 2.

The first half of this book introduces Maven by example, with a series of real-world, multimodule applications you can use as templates. The second half serves as a reference to a wide range of topics. You will learn to:

- Understand the Project Object Model (POM)
- Use the Nexus repository manager
- Integrate Maven with Eclipse, Spring, and Hibernate
- Write and use Maven plugins
- Generate a project website
- Customize a build with properties
- Use build profiles and profile activation
- Create and use Maven assemblies
- Develop with Maven archetypes

Written for the new user and veteran alike, *Maven: The Definitive Guide* is the perfect book to help you manage development projects for software, web applications, and enterprise applications.

*"Today, with Maven on the forefront of all build tools, this invaluable guide by Sonatype is the first resource I reach for when training development teams on using Maven. It helps us handle Mojo curves nimbly, glide over rough sections of plugins, and accelerate through Java builds faster than competitors, allowing us to reach profitable product finish lines."*

—**Matthew J. McCullough, President, Denver Open Source Users Group and Managing Partner, Ambient Ideas, LLC**

The primary contributor to this book is **Tim O'Brien**, online editor of O'Reilly News and author of three other O'Reilly titles: *Harnessing Hibernate*, *Maven: A Developer's Notebook*, and *Jakarta Commons Cookbook*.

**www.oreilly.com**

US $34.99 CAN $34.99

ISBN: 978-0-596-51733-5

53499

9 780596 517335

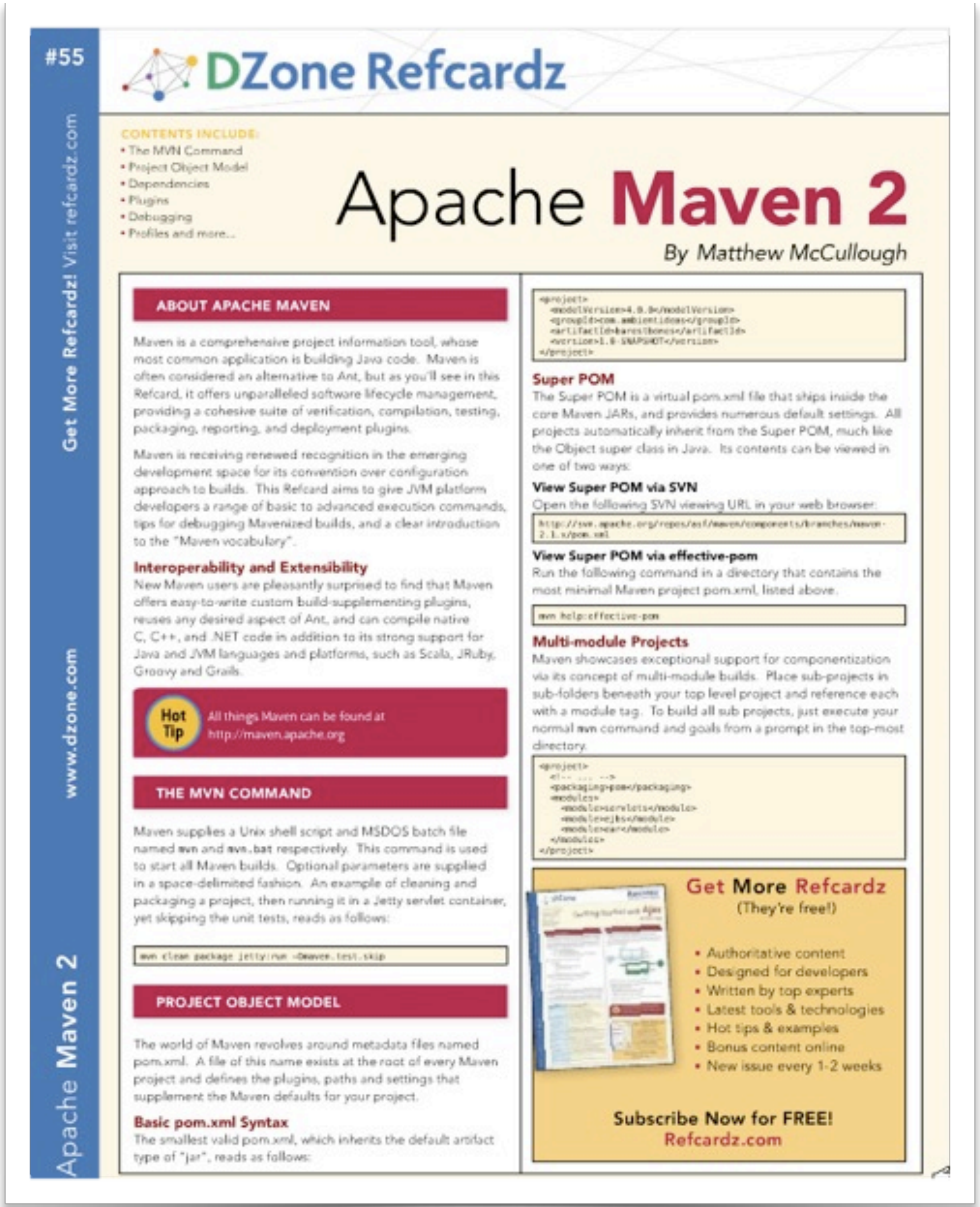
Safari Books Online

**Free online edition** for 45 days with purchase of this book. Details on last page.

NFJS

No Fluff Just Stuff.com

dzone



#55

DZone Refcardz

Get More Refcardz! Visit refcardz.com

www.dzone.com

Apache **Maven 2**

**CONTENTS INCLUDE:**

# Apache Maven 2

*By Matthew McCullough*

## ABOUT APACHE MAVEN

Maven is a comprehensive project information tool, whose most common application is building Java code. Maven is often considered an alternative to Ant, but as you'll see in this Refcard, it offers unparalleled software lifecycle management, providing a cohesive suite of verification, compilation, testing, packaging, reporting, and deployment plugins.

Maven is receiving renewed recognition in the emerging development space for its convention over configuration approach to builds. This Refcard aims to give JVM platform developers a range of basic to advanced execution commands, tips for debugging Mavenized builds, and a clear introduction to the "Maven vocabulary".

### Interoperability and Extensibility

New Maven users are pleasantly surprised to find that Maven offers easy-to-write custom build-supplementing plugins, reuses any desired aspect of Ant, and can compile native C, C++, and .NET code in addition to its strong support for Java and JVM languages and platforms, such as Scala, JRuby, Groovy and Grails.

**Hot Tip** All things Maven can be found at http://maven.apache.org

## THE MVN COMMAND

Maven supplies a Unix shell script and MSDOS batch file named **mvn** and **mvn.bat** respectively. This command is used to start all Maven builds. Optional parameters are supplied in a space-delimited fashion. An example of cleaning and packaging a project, then running it in a Jetty servlet container, yet skipping the unit tests, reads as follows:

```
mvn clean package jetty:run -Dmaven.test.skip
```

## PROJECT OBJECT MODEL

The world of Maven revolves around metadata files named pom.xml. A file of this name exists at the root of every Maven project and defines the plugins, paths and settings that supplement the Maven defaults for your project.

### Basic pom.xml Syntax

The smallest valid pom.xml, which inherits the default artifact type of "jar", reads as follows:

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.ambientideas</groupId>
  <artifactId>barestbones</artifactId>
  <version>1.0-SNAPSHOT</version>
</project>
```

### Super POM

The Super POM is a virtual pom.xml file that ships inside the core Maven JARs, and provides numerous default settings. All projects automatically inherit from the Super POM, much like the Object super class in Java. Its contents can be viewed in one of two ways:

**View Super POM via SVN**

Open the following SVN viewing URL in your web browser:

```
http://svn.apache.org/repos/asf/maven/components/branches/maven-
2.1.x/pom.xml
```

**View Super POM via effective-pom**

Run the following command in a directory that contains the most minimal Maven project pom.xml, listed above.

```
mvn help:effective-pom
```

### Multi-module Projects

Maven showcases exceptional support for componentization via its concept of multi-module builds. Place sub-projects in sub-folders beneath your top level project and reference each with a module tag. To build all sub projects, just execute your normal **mvn** command and goals from a prompt in the top-most directory.

```
<project>
  <!-- ... -->
  <packaging>pom</packaging>
  <modules>
    <module>servlets</module>
    <module>ejbs</module>
    <module>ear</module>
  </modules>
</project>
```

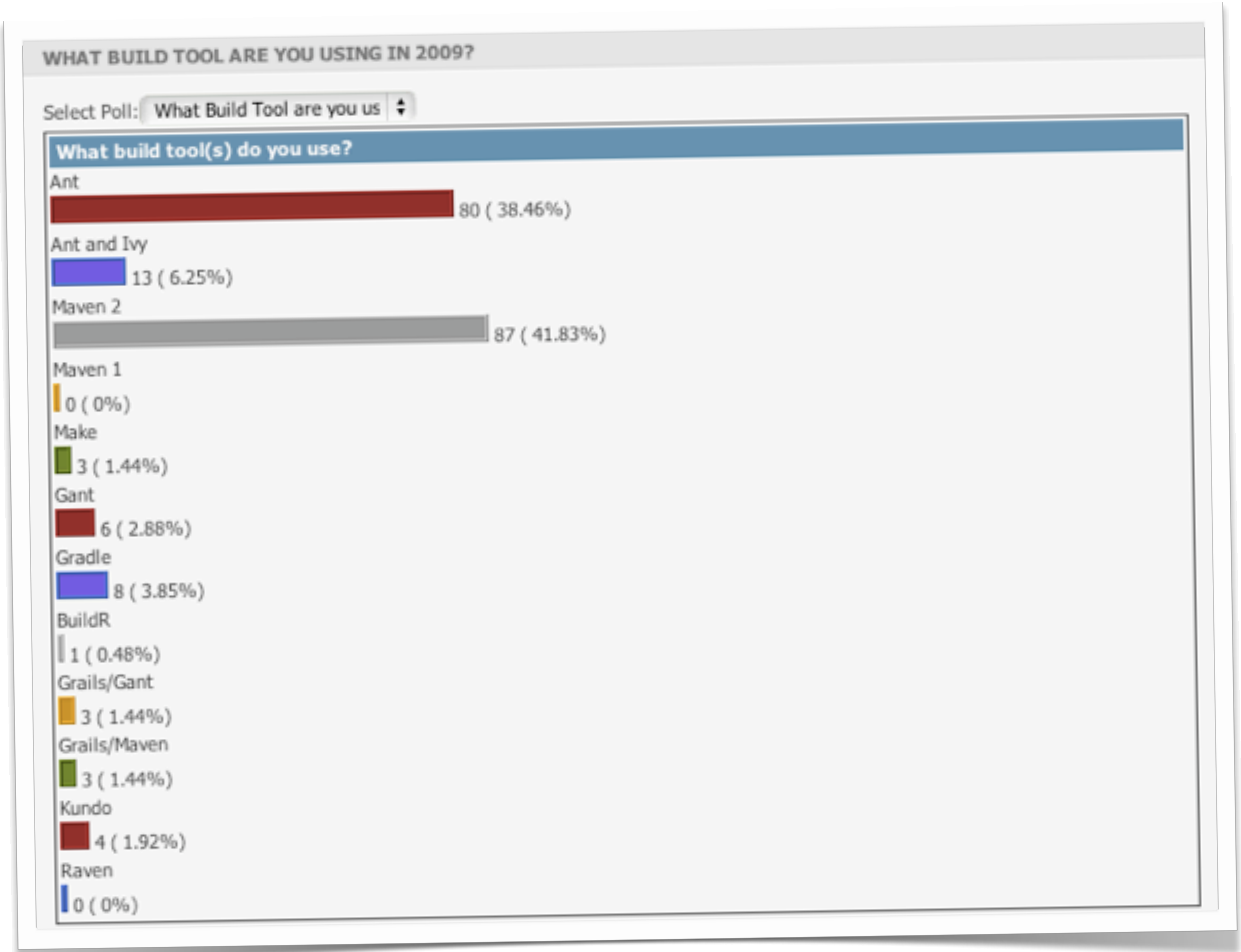
Get More Refcardz
(They're free!)

- Authoritative content
- Designed for developers
- Written by top experts
- Latest tools & technologies
- Hot tips & examples
- Bonus content online
- New issue every 1-2 weeks

Subscribe Now for FREE!
Refcardz.com

http://refcardz.dzone.com/refcardz/apache-maven-2

## Maven use on the upswing

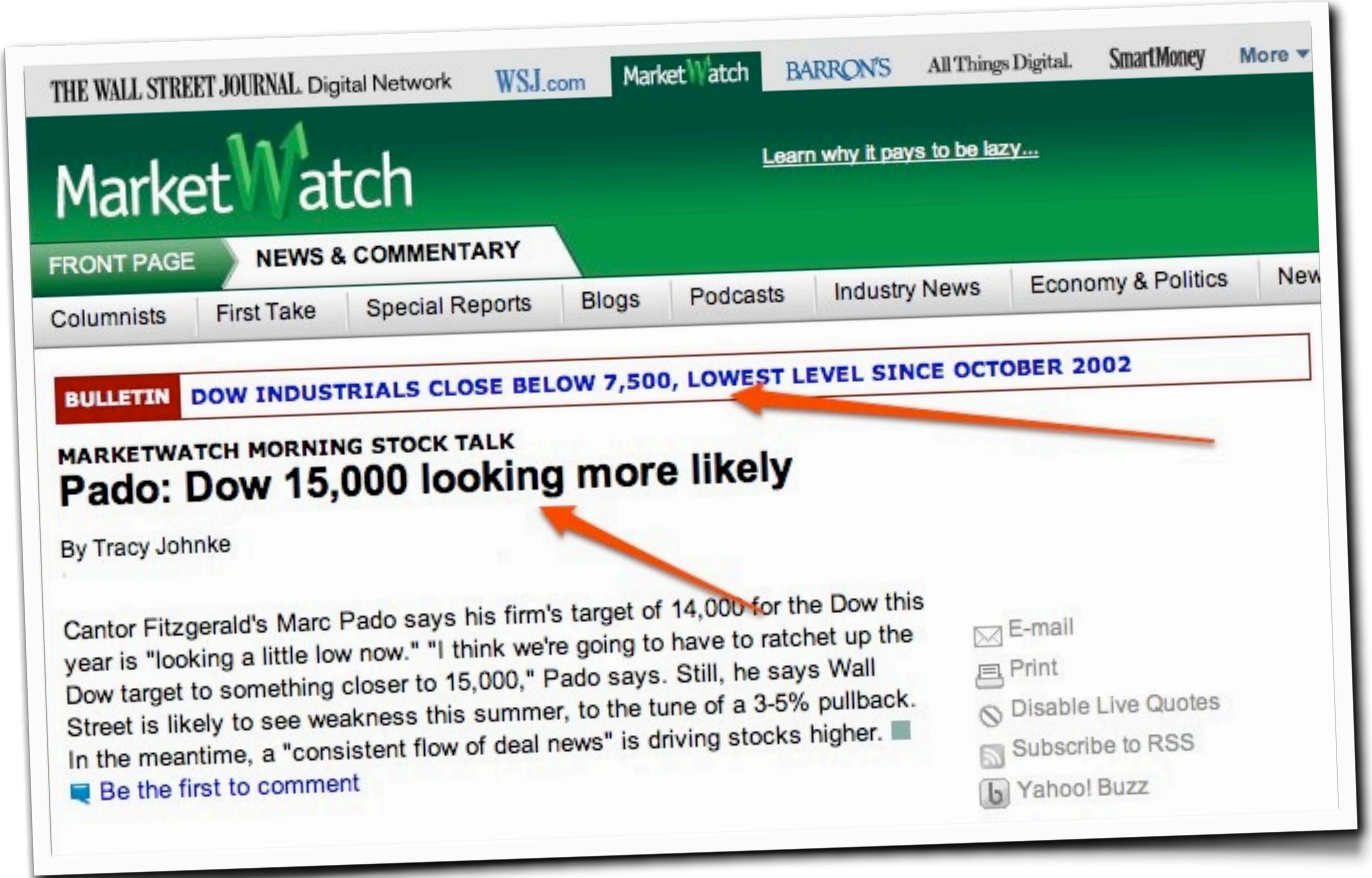


WHAT BUILD TOOL ARE YOU USING IN 2009?

Select Poll: What Build Tool are you us

What build tool(s) do you use?

| | |
|---|---|
| Ant | 80 ( 38.46%) |
| Ant and Ivy | 13 ( 6.25%) |
| Maven 2 | 87 ( 41.83%) |
| Maven 1 | 0 ( 0%) |
| Make | 3 ( 1.44%) |
| Gant | 6 ( 2.88%) |
| Gradle | 8 ( 3.85%) |
| BuildR | 1 ( 0.48%) |
| Grails/Gant | 3 ( 1.44%) |
| Grails/Maven | 3 ( 1.44%) |
| Kundo | 4 ( 1.92%) |
| Raven | 0 ( 0%) |



BULLETIN DOW INDUSTRIALS CLOSE BELOW 7,500, LOWEST LEVEL SINCE OCTOBER 2002

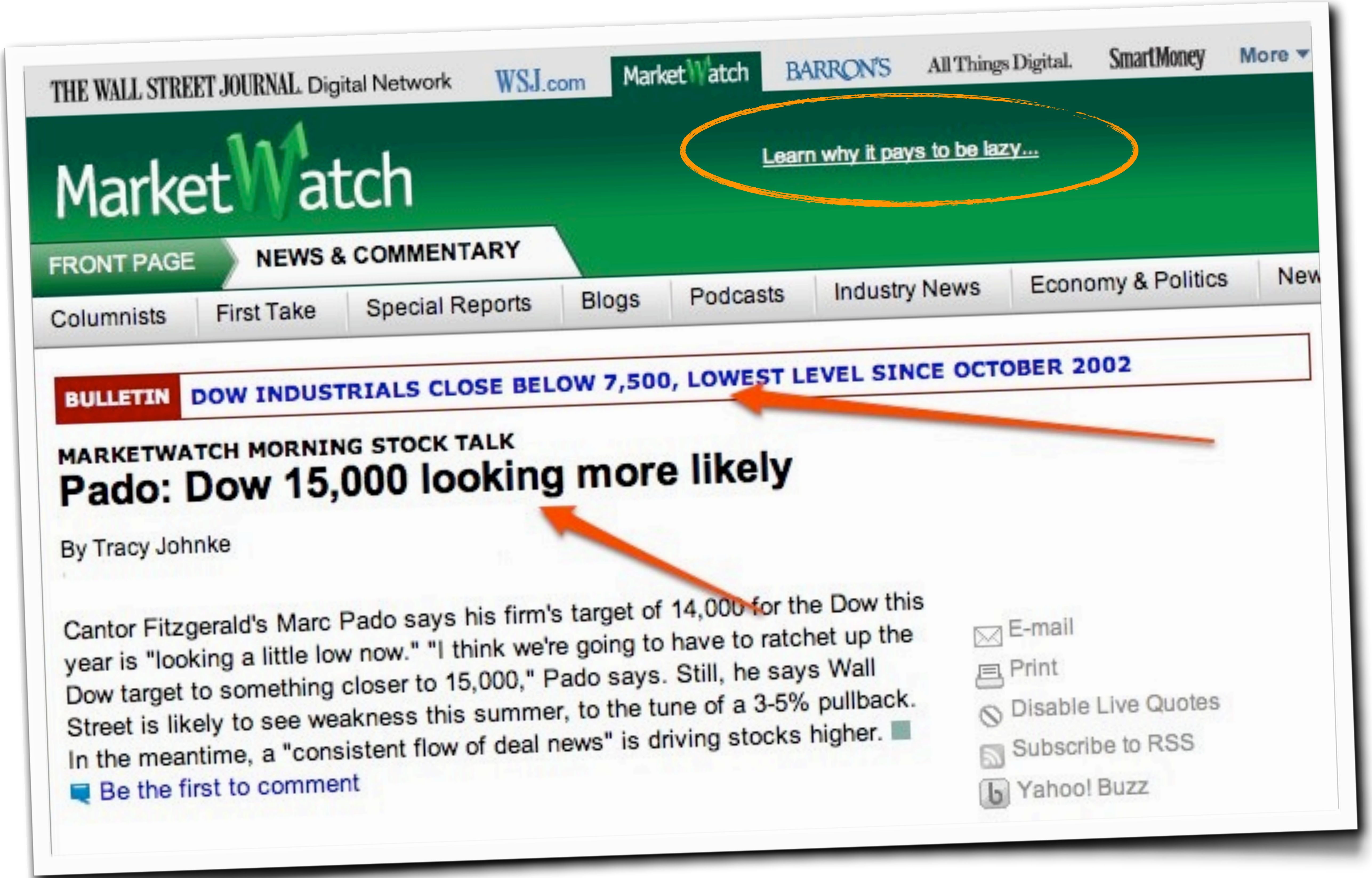
MARKETWATCH MORNING STOCK TALK

**Pado: Dow 15,000 looking more likely**

By Tracy Johnke

Cantor Fitzgerald's Marc Pado says his firm's target of 14,000 for the Dow this year is "looking a little low now." "I think we're going to have to ratchet up the Dow target to something closer to 15,000," Pado says. Still, he says Wall Street is likely to see weakness this summer, to the tune of a 3-5% pullback. In the meantime, a "consistent flow of deal news" is driving stocks higher.

Be the first to comment



DOW INDUSTRIALS CLOSE BELOW 7,500, LOWEST LEVEL SINCE OCTOBER 2002

MARKETWATCH MORNING STOCK TALK

**Pado: Dow 15,000 looking more likely**

By Tracy Johnke

Cantor Fitzgerald's Marc Pado says his firm's target of 14,000 for the Dow this year is "looking a little low now." "I think we're going to have to ratchet up the Dow target to something closer to 15,000," Pado says. Still, he says Wall Street is likely to see weakness this summer, to the tune of a 3-5% pullback. In the meantime, a "consistent flow of deal news" is driving stocks higher.

Be the first to comment

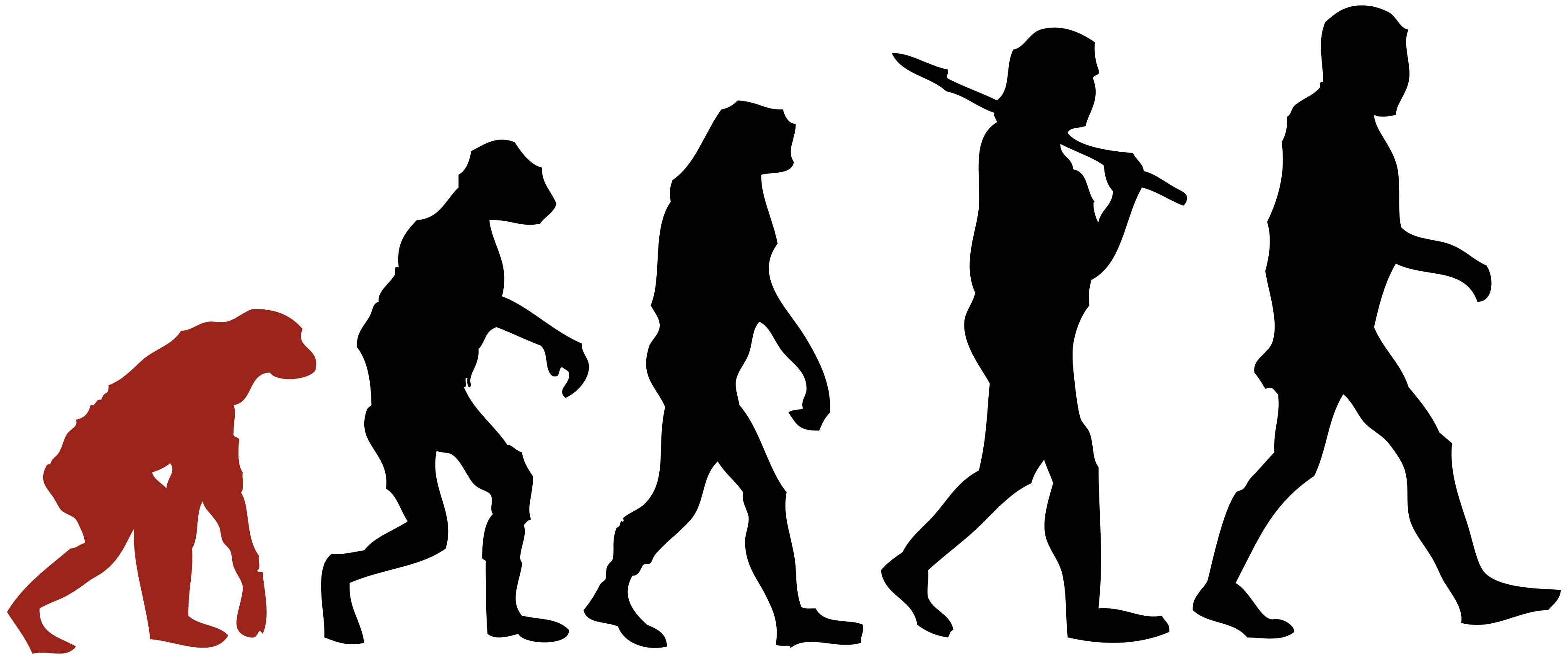
Convention

over

Configuration

Built by:

maven



## Default Goal

## Default Goal

- Often not set.
- Saves typing.
- Communicates author's intended goal.
- Only one goal or phase allowed.

mvn install

```
<project>
  <groupId>com.ambientideas</groupId>
  <artifactId>sample-defaultgoal</artifactId>
  [...]

  <build>
    <defaultGoal>install</defaultGoal>
  </build>

  [...]
</project>
```

mvn install

~~mvn install~~

## mvn

# Super Pom

# Super Pom

- Pseudo-invisible.

# Super Pom

- Pseudo-invisible.
- All projects inherit it.

## Super Pom

- Pseudo-invisible.
- All projects inherit it.
- Specifies file location defaults.

## Super Pom

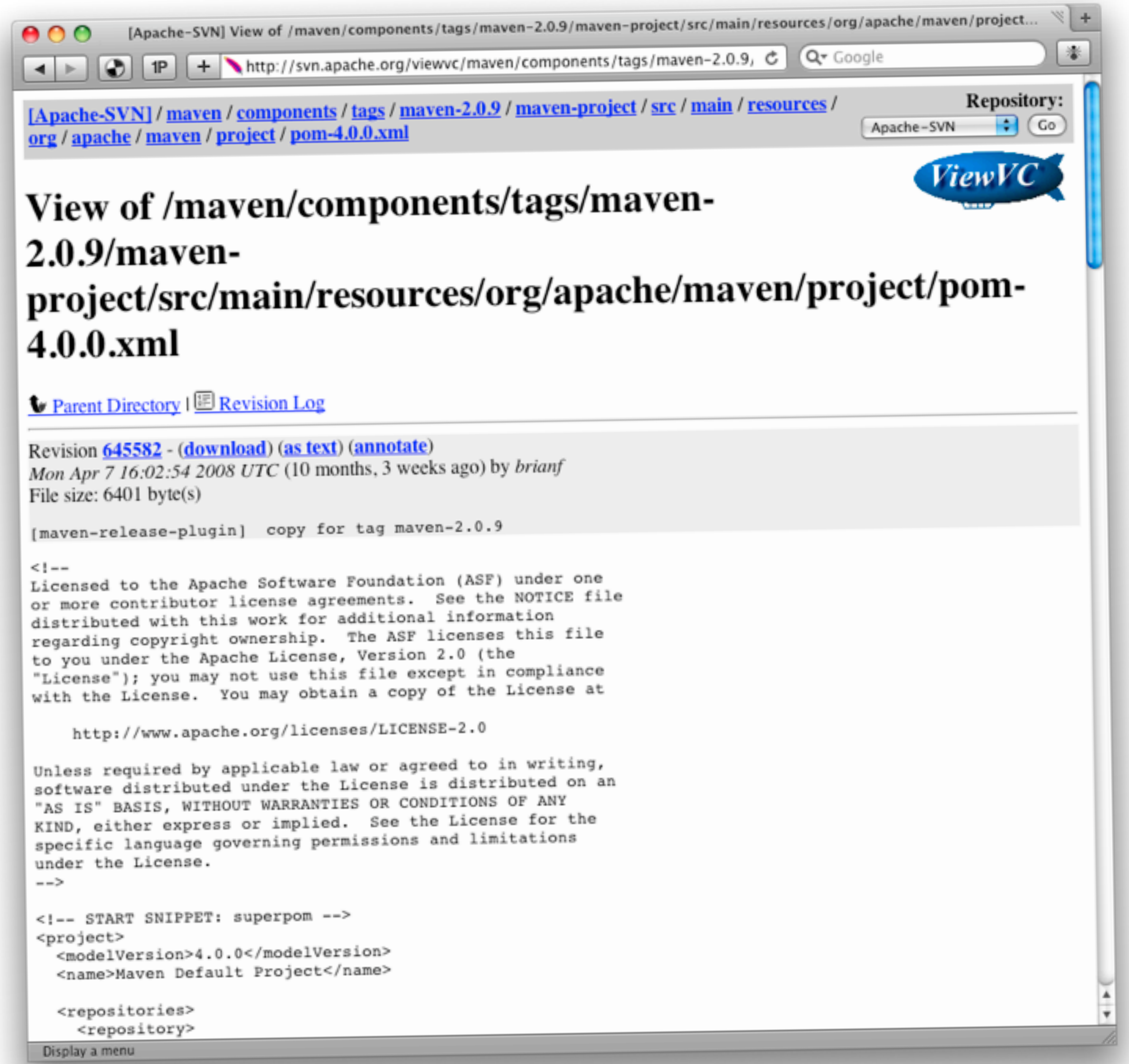
- Pseudo-invisible.
- All projects inherit it.
- Specifies file location defaults.
- Locks version of common plugins.
  - post mvn 2.0.8.
  - Increases build stability.

```
<!-- START SNIPPET: superpom -->
<project>
  <modelVersion>4.0.0</modelVersion>
  <name>Maven Default Project</name>

  <build>
    <directory>target</directory>
    <outputDirectory>target/classes</outputDirectory>
    <finalName>${project.artifactId}-${project.version}</finalName>
    [...]
    <pluginManagement>
       <plugins>
         <plugin>
           <artifactId>maven-clean-plugin</artifactId>
           <version>2.2</version>
         </plugin>
         <plugin>
           <artifactId>maven-compiler-plugin</artifactId>
           <version>2.0.2</version>
         </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

## Super Pom

- Familiarize yourself with it.
- View it in SVN.


[Apache-SVN] View of /maven/components/tags/maven-2.0.9/maven-project/src/main/resources/org/apache/maven/project...
http://svn.apache.org/viewvc/maven/components/tags/maven-2.0.9,
Google
[Apache-SVN] / maven / components / tags / maven-2.0.9 / maven-project / src / main / resources / org / apache / maven / project / pom-4.0.0.xml
Repository: Apache-SVN Go
ViewVC
Parent Directory | Revision Log
Display a menu

## View of /maven/components/tags/maven-2.0.9/maven-project/src/main/resources/org/apache/maven/project/pom-4.0.0.xml

Revision 645582 - (download) (as text) (annotate)
*Mon Apr 7 16:02:54 2008 UTC* (10 months, 3 weeks ago) by *brianf*
File size: 6401 byte(s)

```
[maven-release-plugin]  copy for tag maven-2.0.9
```

```
<!--
Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements.  See the NOTICE file
distributed with this work for additional information
regarding copyright ownership.  The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License.  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied.  See the License for the
specific language governing permissions and limitations
under the License.
-->

<!-- START SNIPPET: superpom -->
<project>
  <modelVersion>4.0.0</modelVersion>
  <name>Maven Default Project</name>

  <repositories>
    <repository>
```

## Saving time with ArcheTypes

# Project via Archetype

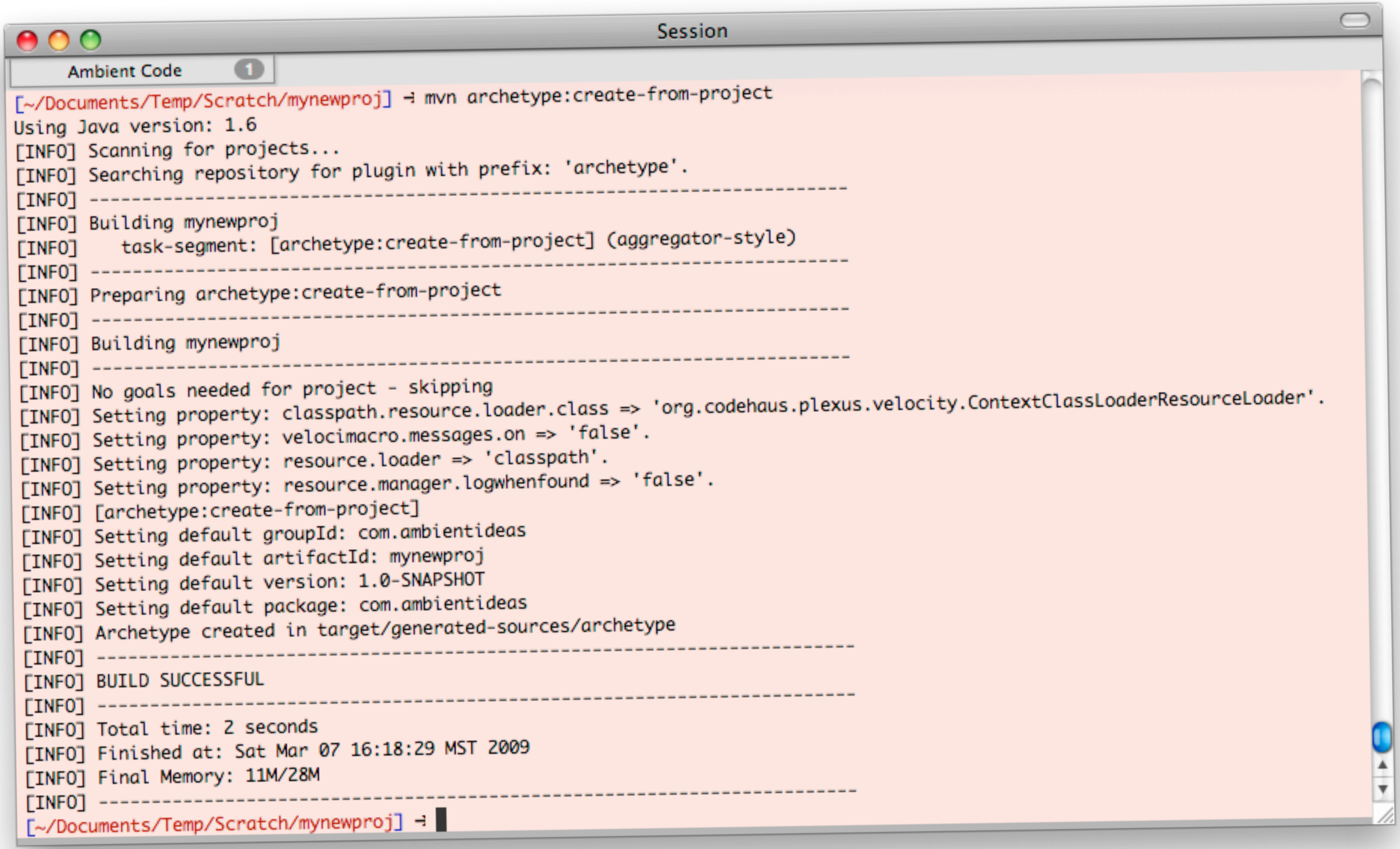
- Project templates on steroids.
    - Seed unit tests.
    - Standardize directory structure.
    - Corporate licenses, OSS licenses.
    - Bundle corporate READMEs.
- Doesn't mutate like a copy-n-paste template.
- Replaceables for classnames, company info.

```
$ mvn archetype:generate
```

## Archetype via Project

- Uses existing project as seed.
- Turns it into a maven archetype.
- Can publish to a repo for others to use.

```
$ mvn archetype:create-from-project
```



```
[~/Documents/Temp/Scratch/mynewproj] ⊣ mvn archetype:create-from-project
Using Java version: 1.6
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'archetype'.
[INFO] ------------------------------------------------------------------------
[INFO] Building mynewproj
[INFO]    task-segment: [archetype:create-from-project] (aggregator-style)
[INFO] ------------------------------------------------------------------------
[INFO] Preparing archetype:create-from-project
[INFO] ------------------------------------------------------------------------
[INFO] Building mynewproj
[INFO] ------------------------------------------------------------------------
[INFO] No goals needed for project - skipping
[INFO] Setting property: classpath.resource.loader.class => 'org.codehaus.plexus.velocity.ContextClassLoaderResourceLoader'.
[INFO] Setting property: velocimacro.messages.on => 'false'.
[INFO] Setting property: resource.loader => 'classpath'.
[INFO] Setting property: resource.manager.logwhenfound => 'false'.
[INFO] [archetype:create-from-project]
[INFO] Setting default groupId: com.ambientideas
[INFO] Setting default artifactId: mynewproj
[INFO] Setting default version: 1.0-SNAPSHOT
[INFO] Setting default package: com.ambientideas
[INFO] Archetype created in target/generated-sources/archetype
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2 seconds
[INFO] Finished at: Sat Mar 07 16:18:29 MST 2009
[INFO] Final Memory: 11M/28M
[INFO] ------------------------------------------------------------------------
[~/Documents/Temp/Scratch/mynewproj] ⊣
```

## Archetype via Archetype

- “Template for templates”

## Archetype via Archetype

- "Template for templates"
- Can be selected from the archetype list.

## Archetype via Archetype

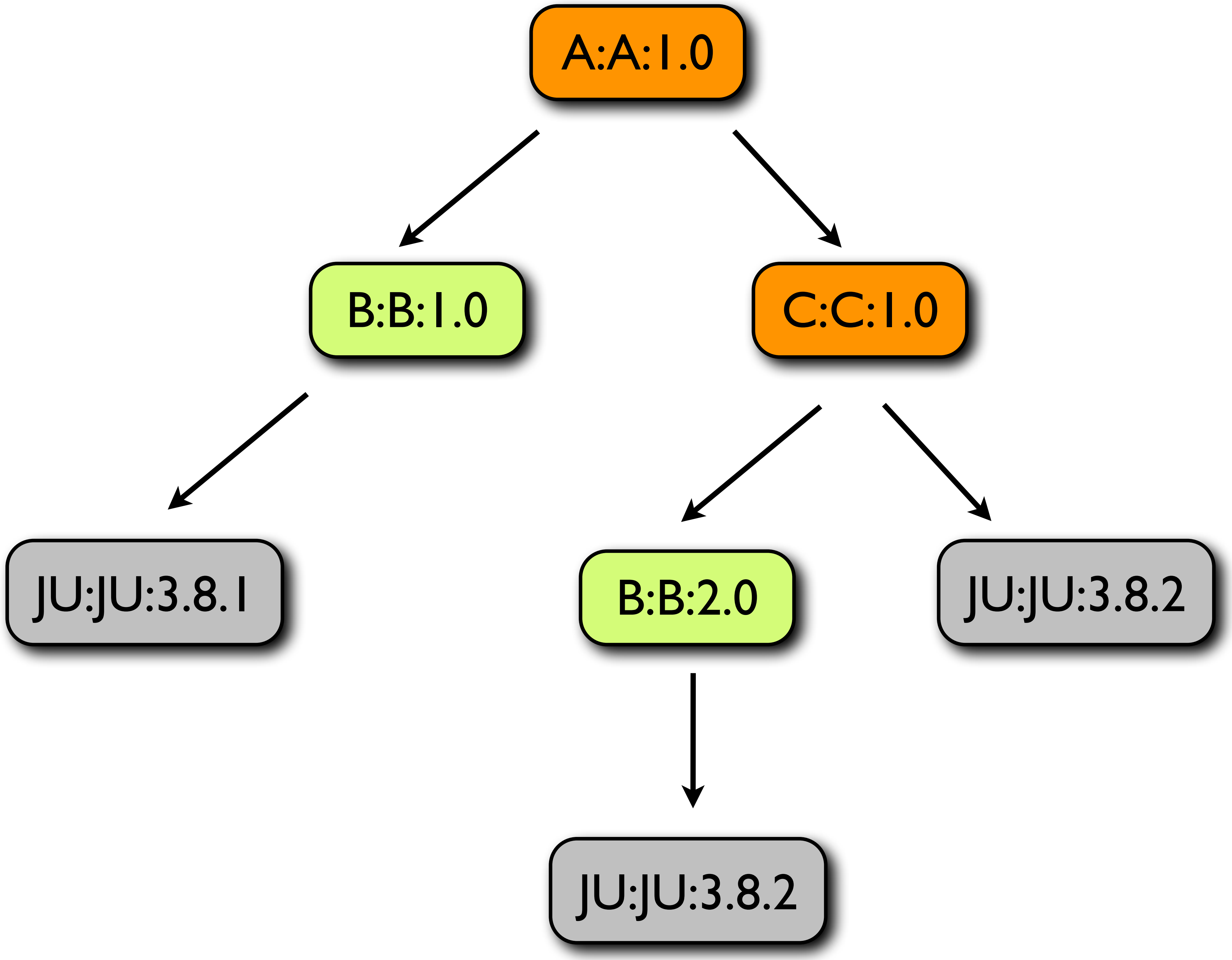
- "Template for templates"
- Can be selected from the archetype list.

```
mvn archetype:generate
```

```
$ mvn archetype:generate
    -DarchetypeGroupId="org.apache.maven.archetypes"
    -DarchetypeArtifactId=maven-archetype-archetype
    -DarchetypeVersion=1.0
    -DgroupId=com.ambientideas
    -DartifactId=mysamplearchetype
```

# Dependency Management

## Dirty Tree



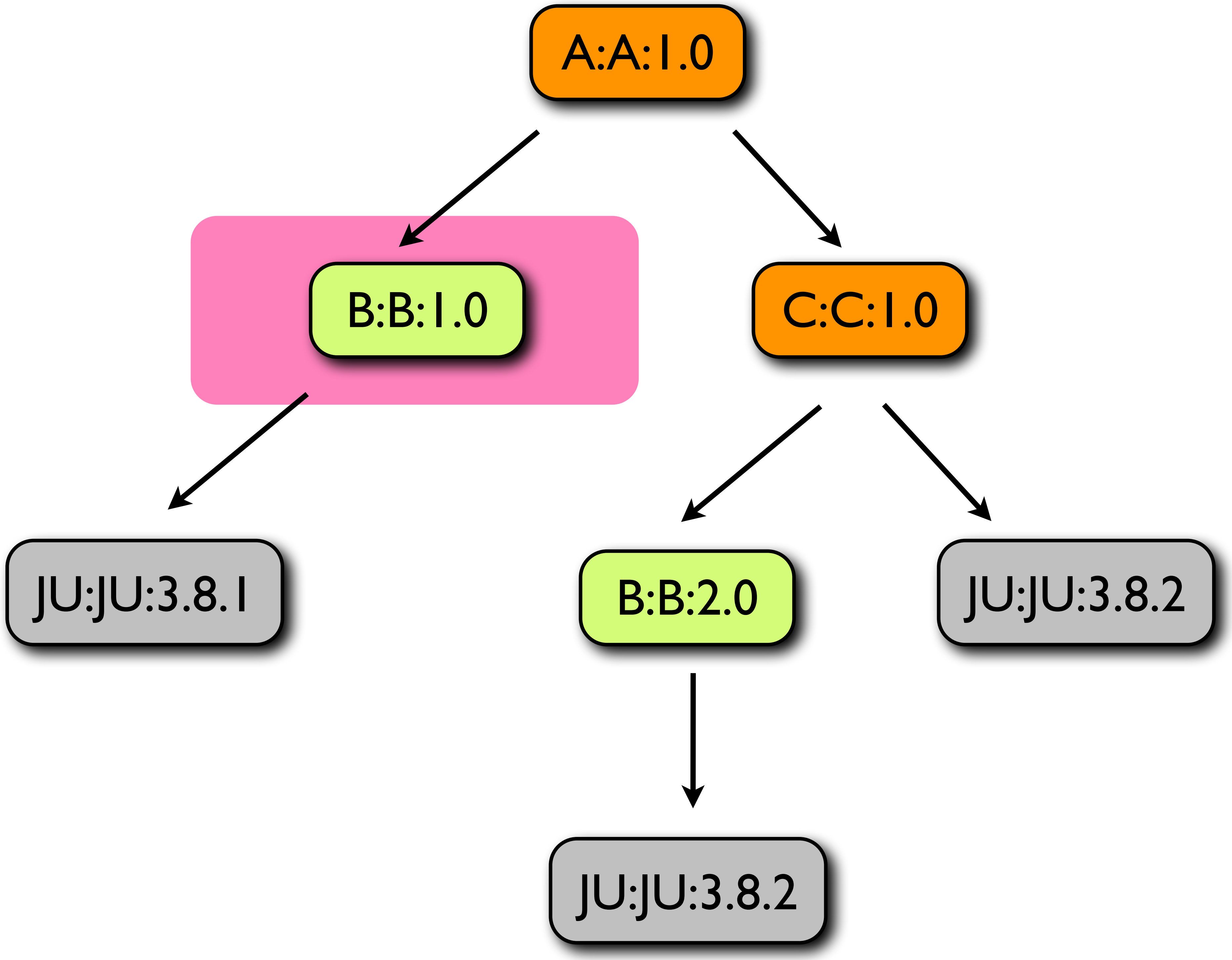
## How does **Maven** resolve **versions**?

## Nearest

## Dirty Tree

$$\begin{tikzcd}
 & & \text{A:A:1.0} \arrow[dl] \arrow[dr] & & \\
 & \text{B:B:1.0} \arrow[dl] & & \text{C:C:1.0} \arrow[dl] \arrow[dr] & \\
\text{JU:JU:3.8.1} & & \text{B:B:2.0} \arrow[d] & & \text{JU:JU:3.8.2} \\
 & & \text{JU:JU:3.8.2} & &
\end{tikzcd}$$

## Dirty Tree

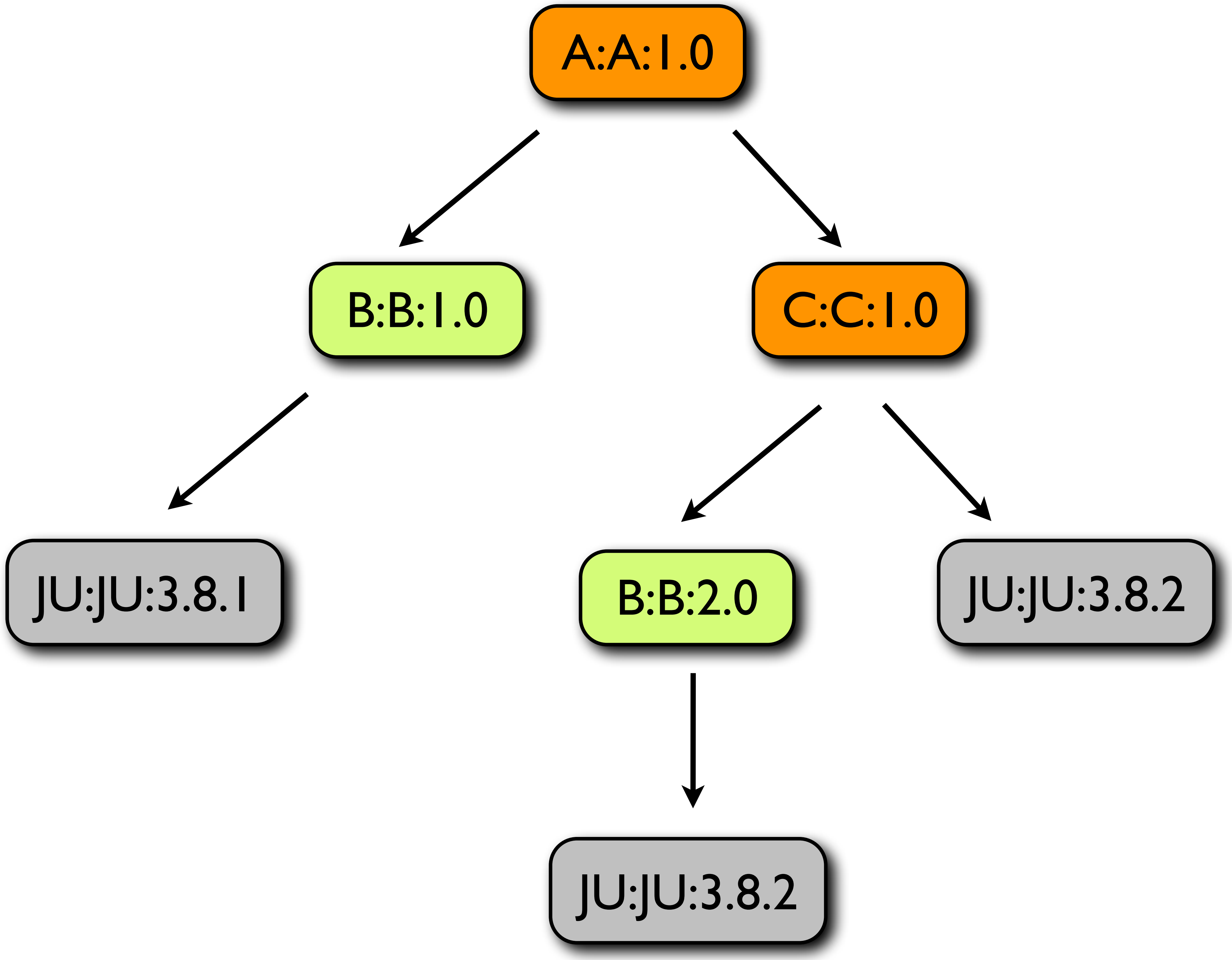


Then **highest** version?

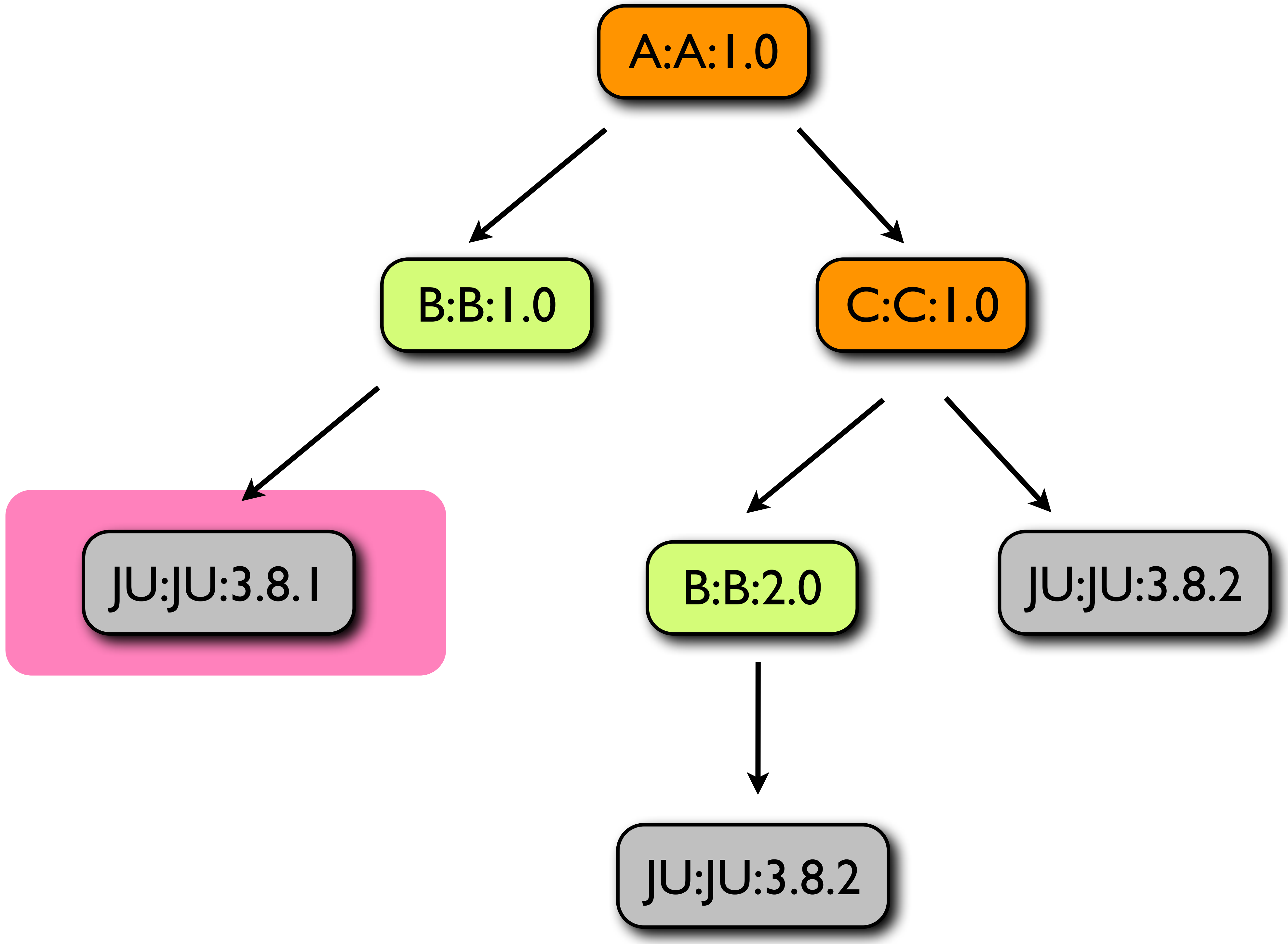
~~Then **highest** version?~~

Then sequential **first** at same distance

## Dirty Tree



## Dirty Tree



## First sequential at same level

```
[INFO] [dependency:tree]
[INFO] root.project:ear:ear:1.0
[INFO] +- root.project:ejbs:ejb:1.0:compile
[INFO] |  \- junit:junit:jar:4.0:compile
[INFO] +- root.project.servlets:servlet:war:1.0:compile
[INFO] +- root.project:primary-source:jar:1.0:compile
[INFO] \- root.project.projects:logging:jar:1.0:compile
```

## First sequential at same level

4.0 is resolved

```
[INFO] [dependency:tree]
[INFO] root.project:ear:ear:1.0
[INFO] +- root.project:ejbs:ejb:1.0:compile
[INFO] |  \- junit:junit:jar:4.0:compile
[INFO] +- root.project.servlets:servlet:war:1.0:compile
[INFO] +- root.project:primary-source:jar:1.0:compile
[INFO] \- root.project.projects:logging:jar:1.0:compile
```

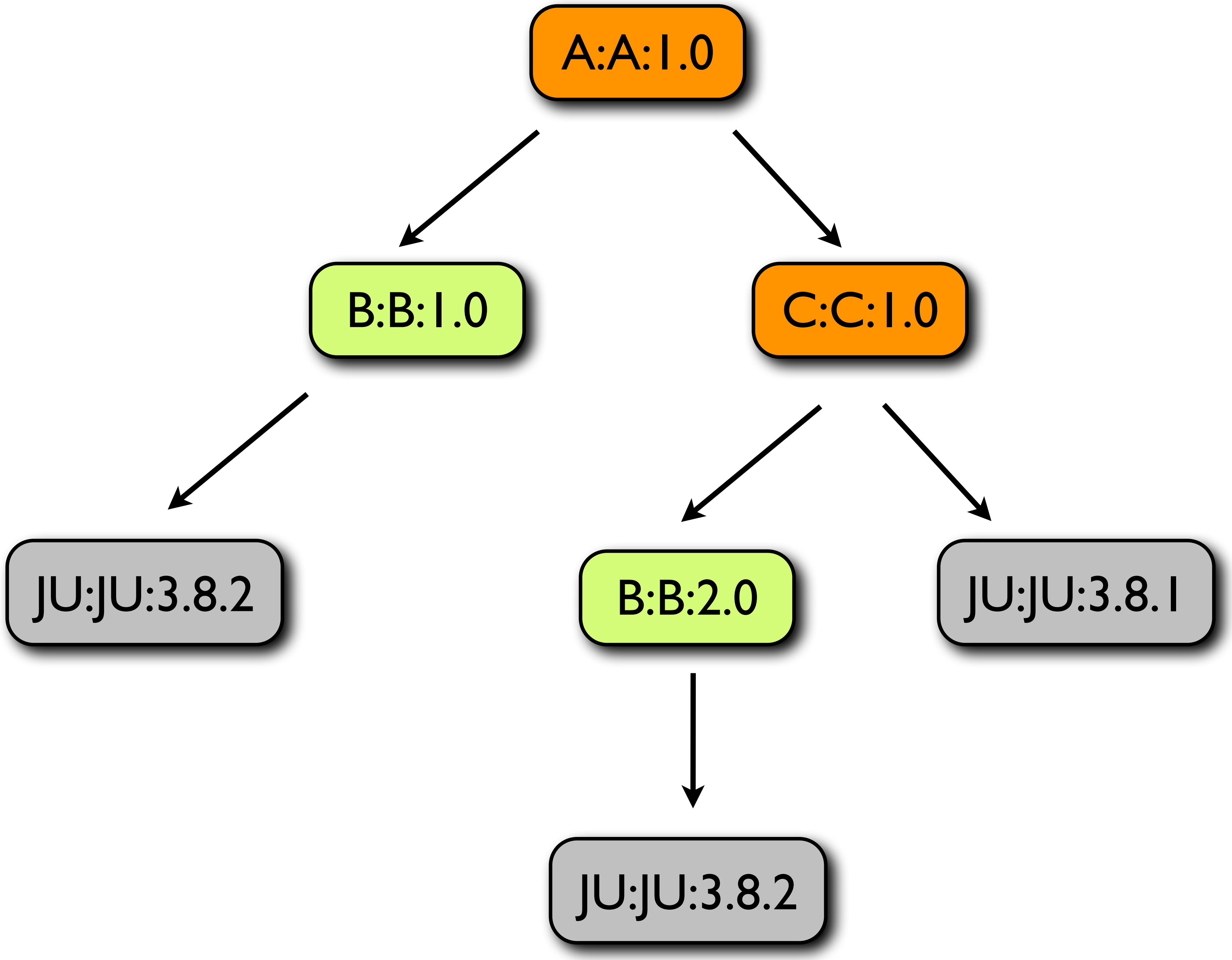
## First sequential at same level

4.0 is resolved

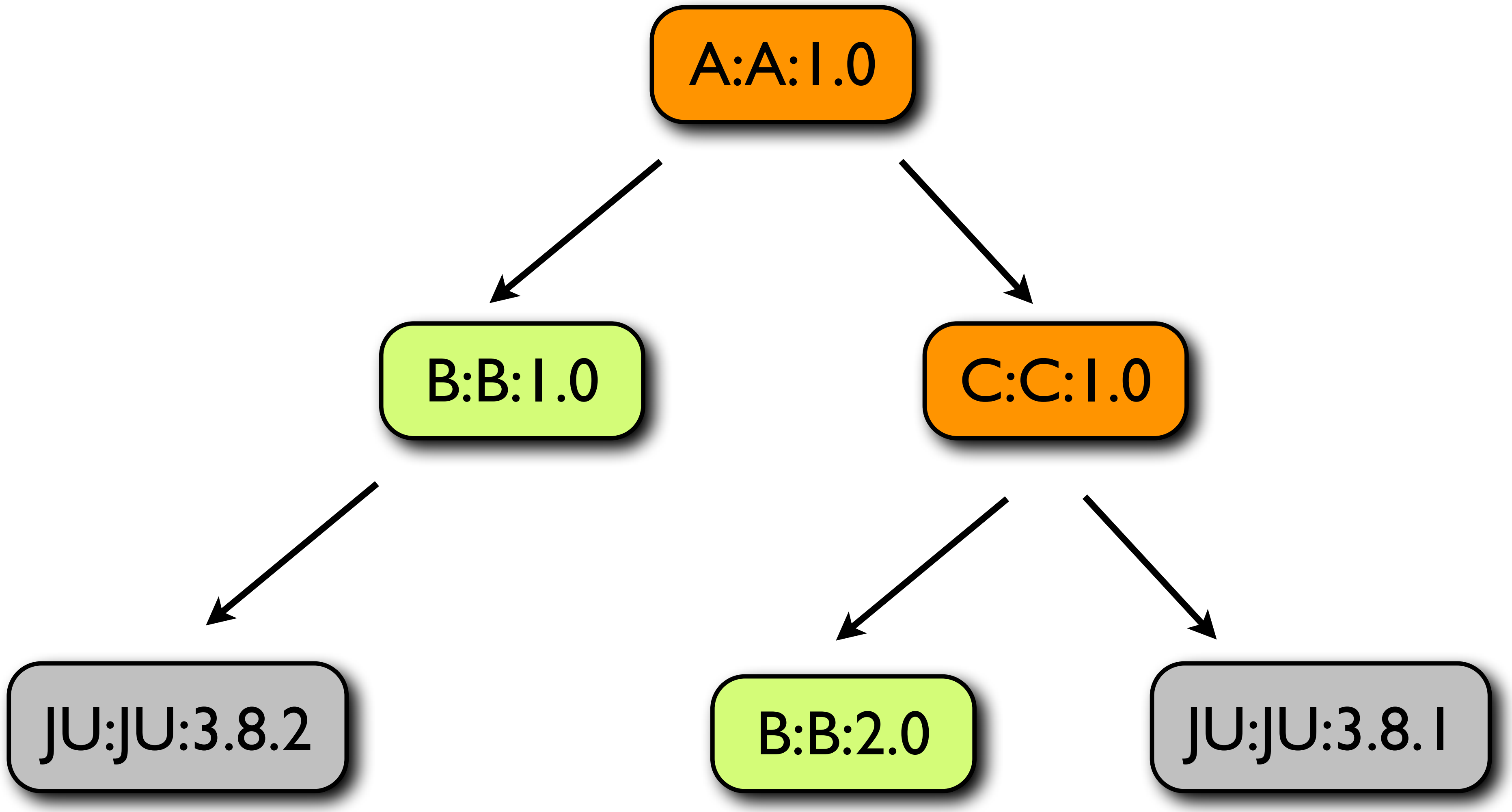
```
[INFO] [dependency:tree]
[INFO] root.project:ear:ear:1.0
[INFO] +- root.project:ejbs:ejb:1.0:compile
[INFO] |  \- junit:junit:jar:4.0:compile
[INFO] +- root.project.servlets:servlet:war:1.0:compile
[INFO] +- root.project:primary-source:jar:1.0:compile
[INFO] \- root.project.projects:logging:jar:1.0:compile
```

Even though I have 4.4 declared as a dependency of primary-source

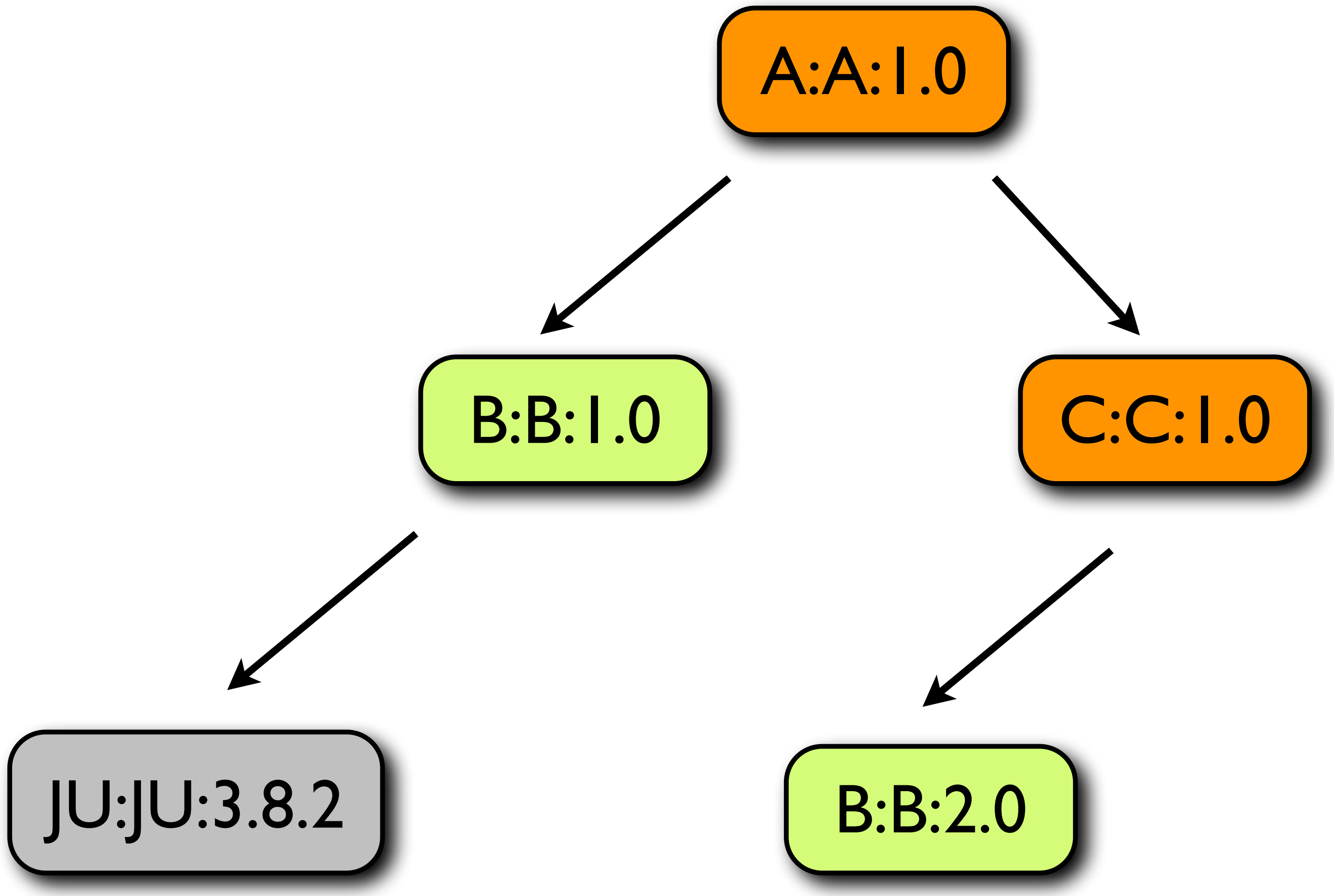
## Resolving Tree



## Resolving Tree



## Resolving Tree



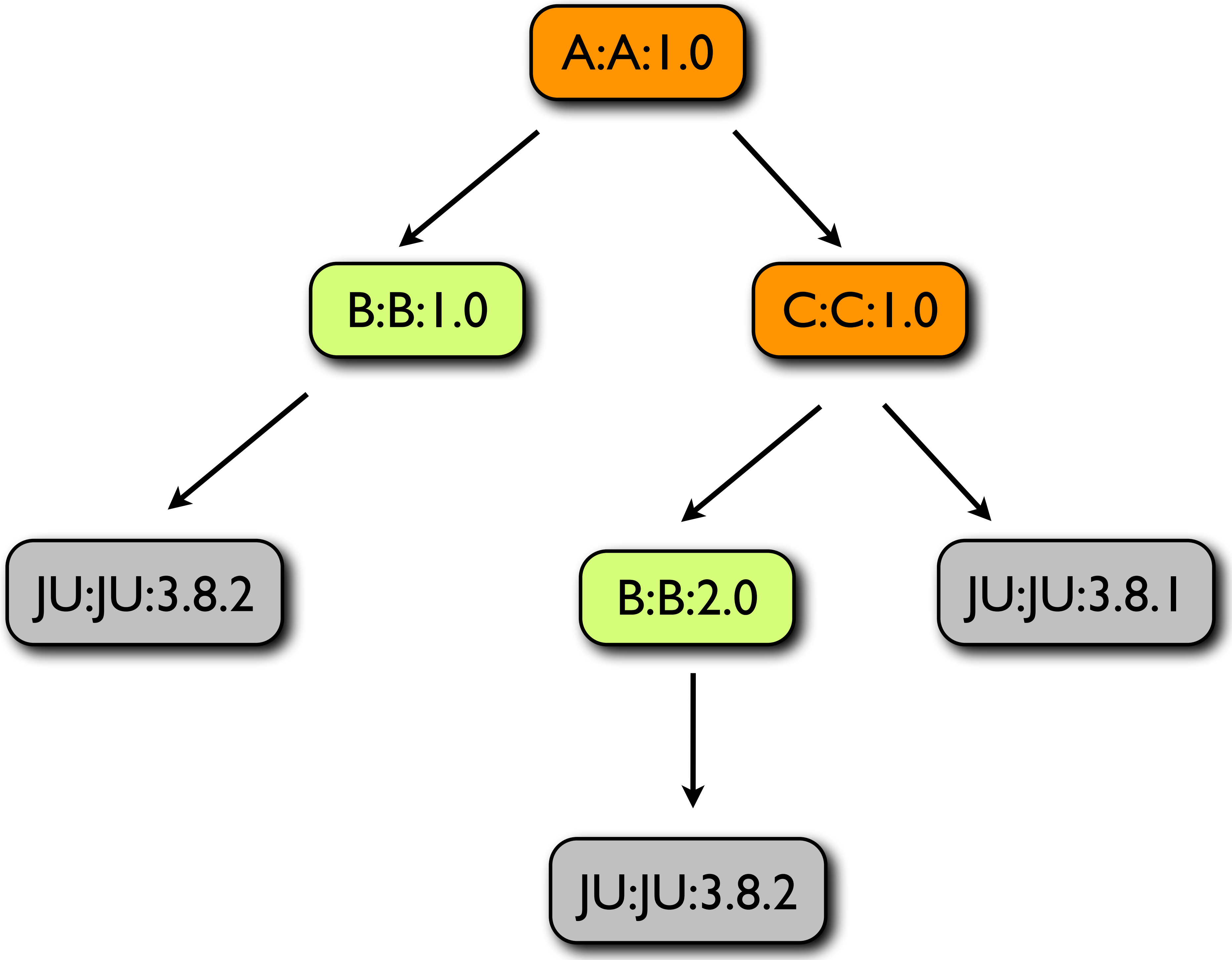
## Resolved Tree

$$\begin{tikzcd}
 & & \text{A:A:1.0} \arrow[dl] \arrow[dr] & \\
 & \text{B:B:1.0} \arrow[dl] & & \text{C:C:1.0} \\
\text{JU:JU:3.8.2} & & &
\end{tikzcd}$$

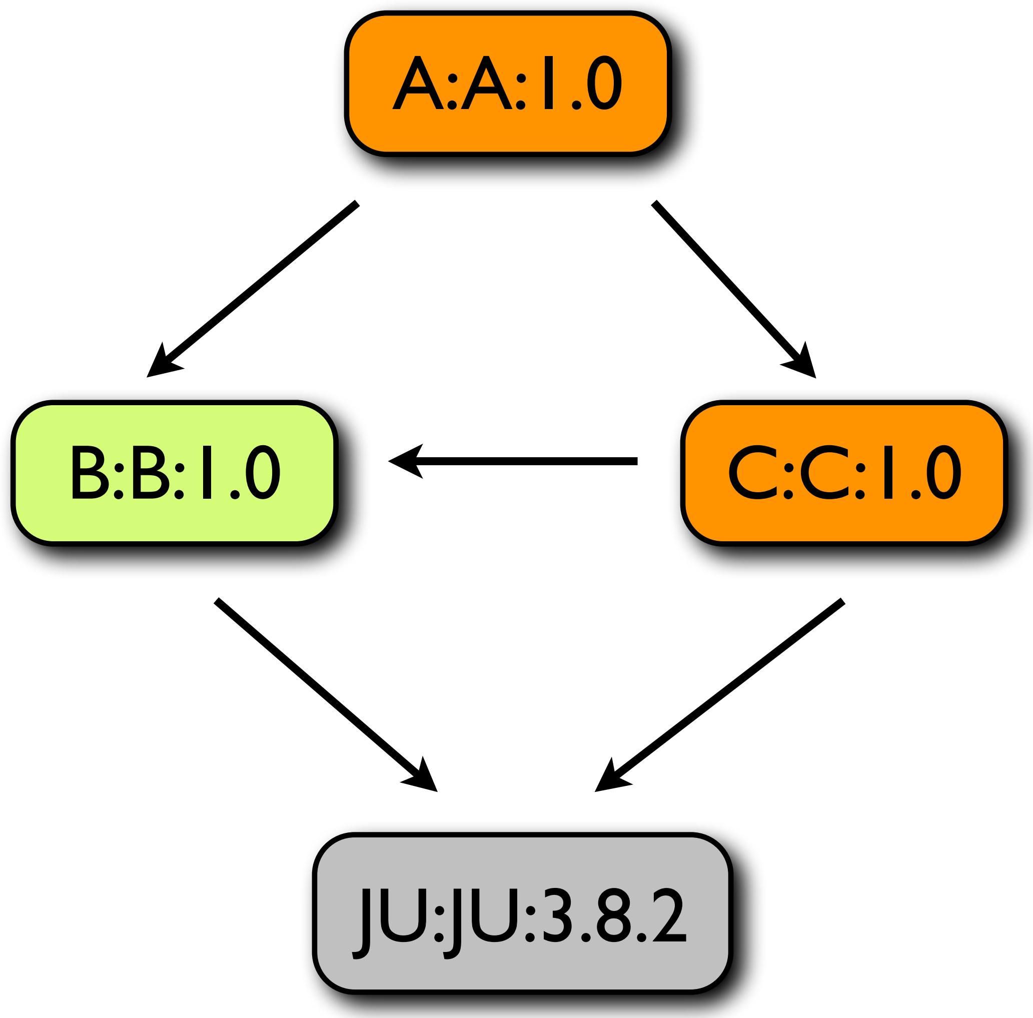
We've resolved the **version tree.**

How about the **dependency graph**?

## Dirty Graph



## Resolved Graph



## A **dice roll** for **complex** graphs**?**

## Declare **desired** versions

Advertise **authorized** versions

# <*MANAGEMENT>

- `<dependencyManagement>`
  - Inheritance for dependency versions.

# <*Management>

- `<dependencyManagement>`
  - Inheritance for dependency versions.
- `<pluginManagement>`
  - Inheritance for plugin versions and config.

<dependencyManagement> and <pluginManagement>

are essentially **identical**

## **Minimizing** declarations with **<*Management>**

In the **base** pom.xml...

```xml
<pluginManagement>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-antrun-plugin</artifactId>
      <version>1.1</version>
      <executions>
        <execution>
          <id>echodir</id>
          <goals>
            <goal>run</goal>
          </goals>
          <phase>install</phase>
          <configuration>
            <tasks>
              <echo>Build Dir: ${project.build.directory}</echo>
            </tasks>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</pluginManagement>
```

```
<pluginManagement>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-antrun-plugin</artifactId>
      <version>1.1</version>
      <executions>
        <execution>
          <id>echodir</id>
          <goals>
            <goal>run</goal>
          </goals>
          <phase>install</phase>
          <configuration>
            <tasks>
              <echo>Build Dir: ${project.build.directory}</echo>
            </tasks>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</pluginManagement>
```

```xml
<pluginManagement>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-antrun-plugin</artifactId>
      <version>1.1</version>
      <executions>
        <execution>
          <id>echodir</id>
          <goals>
            <goal>run</goal>
          </goals>
          <phase>install</phase>
          <configuration>
            <tasks>
              <echo>Build Dir: ${project.build.directory}</echo>
            </tasks>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</pluginManagement>
```

```xml
<pluginManagement>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-antrun-plugin</artifactId>
      <version>1.1</version>
      <executions>
        <execution>
          <id>echodir</id>
          <goals>
            <goal>run</goal>
          </goals>
          <phase>install</phase>
          <configuration>
            <tasks>
              <echo>Build Dir: ${project.build.directory}</echo>
            </tasks>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</pluginManagement>
```

## In the **child** pom.xml...

```xml
<plugins>
  <plugin>
    <artifactId>maven-antrun-plugin</artifactId>
  </plugin>
</plugins>
```

# Dependency Lists

- List view
  - `mvn dependency:resolve`
- Tree view
  - `mvn dependency:tree`
- Plugin list view
  - `mvn dependency:resolve-plugins`

```
$ mvn dependency:tree
[INFO] com.ambientideas:sample13-wicket:war:1.0-SNAPSHOT
[INFO] +- org.apache.wicket:wicket:jar:1.3.2:compile
[INFO] |  \- org.slf4j:slf4j-api:jar:1.4.2:compile
[INFO] +- org.apache.wicket:wicket-extensions:jar:1.3.2:compile
[INFO] +- commons-collections:commons-collections:jar:3.1:compile
[INFO] +- org.slf4j:slf4j-log4j12:jar:1.4.2:compile
[INFO] +- log4j:log4j:jar:1.2.14:compile
[INFO] +- junit:junit:jar:3.8.2:test
[INFO] +- org.mortbay.jetty:jetty:jar:6.1.4:provided
[INFO] |  \- org.mortbay.jetty:servlet-api-2.5:jar:6.1.4:provided
[INFO] +- org.mortbay.jetty:jetty-util:jar:6.1.4:provided
[INFO] \- org.mortbay.jetty:jetty-management:jar:6.1.4:provided
[INFO]    +- mx4j:mx4j:jar:3.0.1:provided
[INFO]    \- mx4j:mx4j-tools:jar:3.0.1:provided
```

## Dependency Analysis

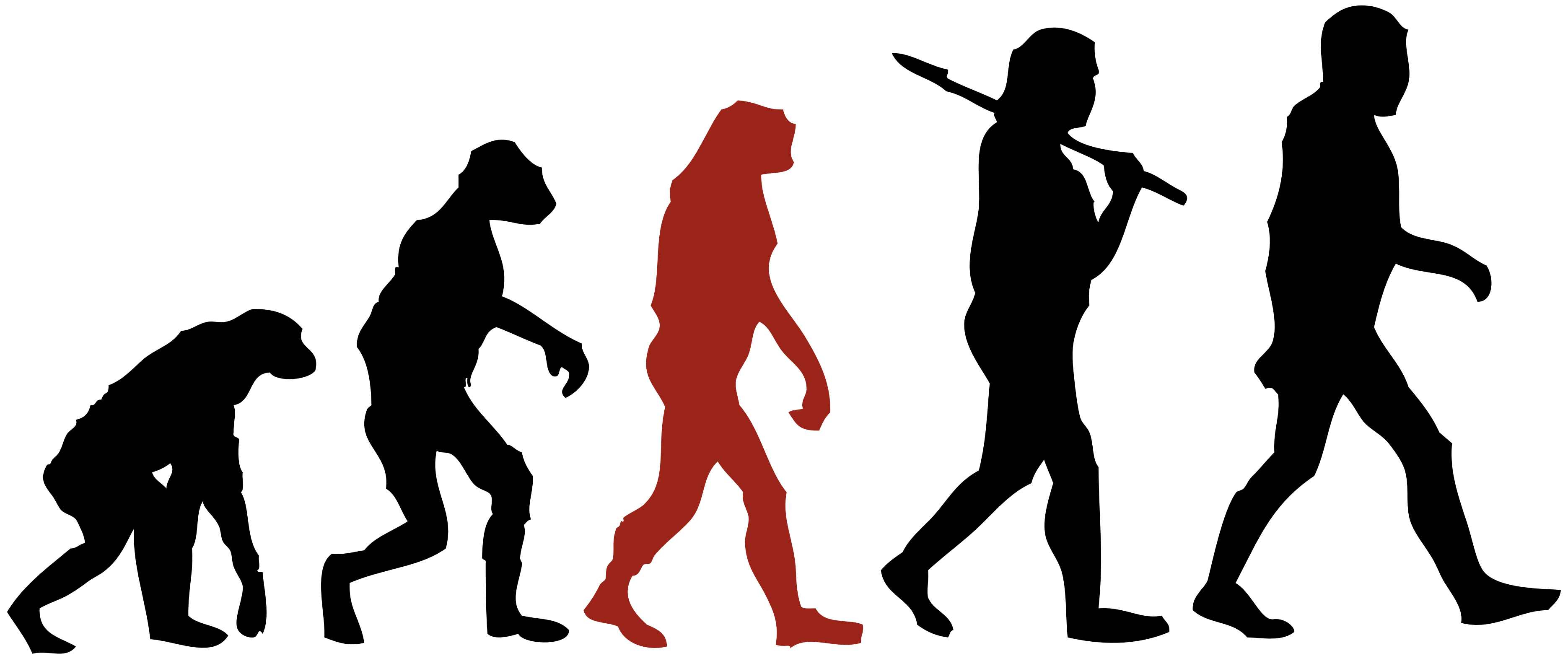
- Analyze to help prune unneeded.
  - `mvn dependency:analyze`

## Dependency Analysis

- See if overrides are colliding.
  - `mvn dependency:analyze-dep-mgt`

```
$ mvn dependency:analyze

[WARNING] Unused declared dependencies found:
[WARNING]  org.slf4j:slf4j-log4j12:jar:1.4.2:compile
[WARNING]  log4j:log4j:jar:1.2.14:compile
[WARNING]  org.mortbay.jetty:jetty-management:jar:6.1.4:provided
[WARNING]  org.apache.wicket:wicket-extensions:jar:1.3.2:compile
[WARNING]  commons-collections:commons-collections:jar:3.1:compile
```



## DeBugging

Purity: 99.98%
100 ml
TOXIC
Pesticide
(concentrate)
Toxic by inhalation, in contact with skin and if swallowed.
Do not release into the environment!

## Maven Debug Flags

- Output full error stacktraces.
  - `mvn <anygoal> -e`
- Output debug level operational info.
  - `mvn <anygoal> -X`
  - Always use when submitting questions or bug reports.

## $ mvn <anygoal> -X

```
[DEBUG] Configuring mojo 'org.apache.maven.plugins:maven-dependency-plugin:2.0:analyze' -->
[DEBUG]   (f) baseDir = /Users/mccm06/Documents/Teach/Courses/Mastering-Maven-1Hour/examples/maven-training.git/sample13-wicket-withdependencies
[DEBUG]   (f) failOnWarning = false
[DEBUG]   (f) project = MavenProject: com.ambientideas:sample13-wicket-withunneededdependencies:1.0-SNAPSHOT @ /Users/mccm06/Documents/Teach/Courses/Mastering-Maven-1Hour/examples/maven-training.git/sample13-wicket-withunneededdependencies/pom.xml
[DEBUG]   (f) scriptableFlag = $$%%%
[DEBUG]   (f) scriptableOutput = false
[DEBUG]   (f) verbose = false
```

# Debug Maven

- Waits on socket for debugger to connect.
  - `mvnDebug <anygoal>`
- Useful for debugging plugins.

- Can be used for unit test debugging.
  - `mvnDebug test -DforkMode=none`

## Debug Unit Tests

- Waits on socket for debugger to connect.

```
mvn test -Dmaven.surefire.debug
```

# Groovy Support

# Groovy Support

- Groovy application code compilation.
  - `mvn archetype:generate ...gmaven-archetype-basic`
- Groovy maven plugin authoring.
  - `mvn archetype:generate ...gmaven-archetype-mojo`
- Joint compiler in both cases.

# Grails Support

- Grails now fully supports Maven...
  - Archetype
  - Grails goals: create-controller, run-app, etc.

```
mvn org.apache.maven.plugins:maven-archetype-plugin:2.0-
alpha-4:generate
 -DarchetypeGroupId=org.grails \
 -DarchetypeArtifactId=grails-maven-archetype \
 -DarchetypeVersion=1.0-SNAPSHOT \
 -DarchetypeRepository=http://snapshots.repository.codehaus.org \
 -DgroupId=com.ambientideas -DartifactId=sample-grails
```

# command line Help!

# Maven Help Plugin

```
mvn help:describe -Dplugin=<anygoal>
```

- Lists and describes plugin goals.

# Maven Help Plugin

```
mvn help:system
```

- Outputs environment variables and system properties.

# Maven Help Plugin

```
mvn help:active-profiles
```

- See what profiles are being triggered by environment, files, params.

## Maven Help Plugin

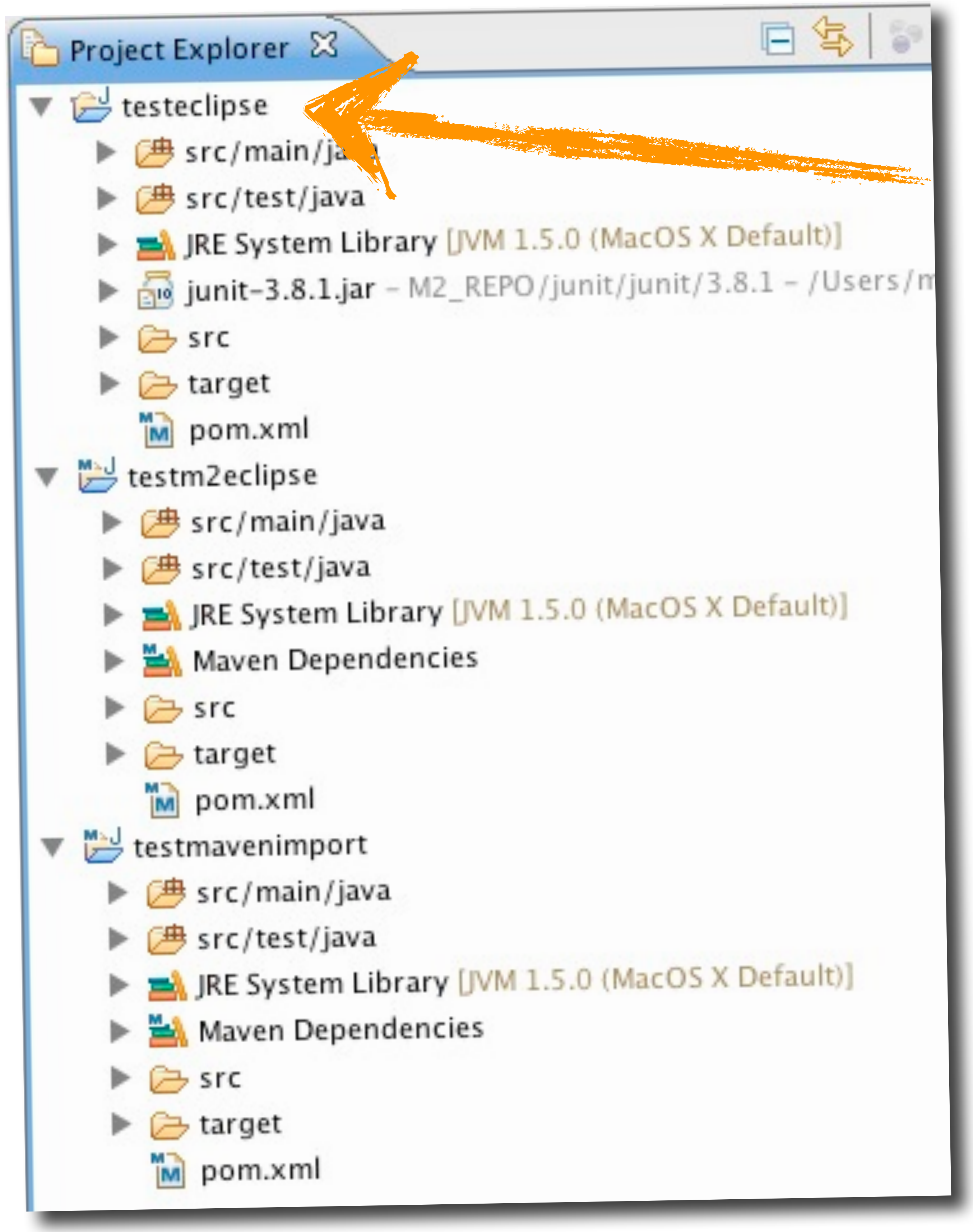
```
mvn help:effective-pom
```

- Outputs the resultant pom.
- Includes inherited sections.

M2Eclipse

# m2eclipse Plugin

- Advanced Eclipse GUI for Maven
  - Update site: http://m2eclipse.sonatype.org/update-dev/
- Eclipse 3.2-3.4 compatibility
  - 3.5 support arrived May 09
- Features include:
  - Exclusion/Inclusion via clicks
  - Searching for artifacts
  - Dependency diagrams
  - Click-to-run Maven goals



```
Project Explorer
▼ testeclipse
    ▶ src/main/ja
    ▶ src/test/java
    ▶ JRE System Library [JVM 1.5.0 (MacOS X Default)]
    ▶ junit-3.8.1.jar - M2_REPO/junit/junit/3.8.1 - /Users/m
    ▶ src
    ▶ target
      pom.xml
▼ testm2eclipse
    ▶ src/main/java
    ▶ src/test/java
    ▶ JRE System Library [JVM 1.5.0 (MacOS X Default)]
    ▶ Maven Dependencies
    ▶ src
    ▶ target
      pom.xml
▼ testmavenimport
    ▶ src/main/java
    ▶ src/test/java
    ▶ JRE System Library [JVM 1.5.0 (MacOS X Default)]
    ▶ Maven Dependencies
    ▶ src
    ▶ target
      pom.xml
```

```
mvn eclipse:eclipse
```

1

File > Import > Existing Projects into Workspace

Set up Eclipse Classpath Variable M2_REPO