maven

Build System

Making Projects Make Sense

Maven Special High Intensity Training

Zen Questions

- **Why are we here?**
- What is a project?
- What is Maven?
- What is good?
- What is the sound of one hand clapping?

Zen Questions

- Why are we here?
- **What is a project?**
- What is Maven?
- What is good?
- What is the sound of one hand clapping?

A software project is

- 1 or more files with some ultimate purpose
- In Maven this means
  - A POM giving a file structure purpose
  - The ultimate purpose being an artifact (JAR, WAR, POM, etc.)

POM

- Project Object Model
- Objectifies a project
- Simplifies a build infrastructure

A build infrastructure is

- A collection of tools, process and conventions for managing projects
  - Management
    - Version control
    - Code Compilation/Generation Tools
    - Continuous Integration
  - Constraint
    - Best practices
    - Pre-defined project structure

Zen Questions

- Why are we here?
- What is a project?
- **What is Maven?**
- What is good?
- What is the sound of one hand clapping?

Maven is

- Build and Dependency tool
  - Conceptually elegant (avoids Ant's pitfalls):
    - Declarative. Build complexity doesn't correlate to configuration size
    - Network portability
  - Integrated Dependency Management (unlike Ivy):
    - Object-oriented
- Site and Document management tool
- Project organization and Plugin framework
- Thriving Community
  - Active developer and user community
  - 9+ Gigs of projects in Maven public repository

Maven's Objectives

- Create Standards and Best Practices for healthy and flexible build infrastructures
  - Why only standard APIs and best practices when developing applications? Maven pushes this practice down to the build infrastructure
  - Nourish stable build infrastructures that persevere under high degrees of flux: build with Plexus IoC for increased flexibility
- Vastly simplify project relationships
  - Via inheritance and dependency management
- Increase project portability
  - With Maven-managed local and remote repositories

Maven Framework

- Based on a community of industry experience
- Creates a standard infrastructure for
  - Code/Configuration Generation
  - Provisioning
  - Building
  - Testing
  - Releasing
  - Issue/Task Management
  - Artifact/Asset Management
  - Documentation
  - Continuous Integration
  - IDE Integration

What Maven really is

- Model for software projects
- Patterns for software development and development infrastructures
- Ultimately the basis for a new form of team collaboration

Zen Questions

- Why are we here?
- What is a project?
- What is Maven?
- **What is good?**
- What is the sound of one hand clapping?

Maven was built assuming these are good

- Intentional infrastructure
  - You must invest in your infrastructure to yield returns
  - It must be carefully planned, infrastructures don't just happen
- Long-term sustainability
  - The key is keeping people involved
  - sustainability by virtue versus sustainability by force
- Healthy growth
  - Some constraints are necessary
  - Style is a constraint that reduces erratic behaviors
- Promotion of community
  - Create opportunities for people to interact

Maven Achieves These By

- Model-Driven Development
  - Project Object Model (POM)
  - Standard build lifecycles
- Convention over Configuration
  - Standard directory layout
  - One primary artifact per build
  - Naming conventions
- Encapsulate and Reuse build logic
  - Maven is a plugin execution framework
  - All build logic is encapsulated in plugins
- Coherent Organization of dependencies

A consistent model and patterns makes

- Automation easier
  - Releasing
  - Continuous Integration
  - IDE Workspace Materialization
- Tooling easier
  - Dependency metadata
  - Plugin metadata
  - Standard lifecycle
- Real dependency analysis possible

Zen Questions

- Why are we here?
- What is a project?
- What is Maven?
- What is good?
- **What is the sound of one hand clapping?**

The sound of one ~~hand~~ build ~~clapping~~ crashing

```
[INFO] ------------------------------------------------------------------------
[INFO] Building JBoss AOP MC Integration
[INFO]    task-segment: [install]
[INFO] ------------------------------------------------------------------------
…
[INFO] [resources:resources]
[INFO] Error for project: JBoss AOP MC Integration (during install)
[INFO] ------------------------------------------------------------------------
[INFO] Failed to resolve artifact.

Missing:
----------
1) org.jboss.micro:kernel:jar:2.0.0-beta-SNAPSHOT                <= missing dependency

  Try downloading the file manually from the project website.

  Then, install it using the command:
      mvn install:install-file -DgroupId=org.jboss.micro -DartifactId=kernel \  <= how to install it
          -Dversion=2.0.0-beta-SNAPSHOT -Dpackaging=jar -Dfile=/path/to/file

  Path to dependency:
        1) org.jboss.micro:aop-mc-int:jar:2.0.0-beta-SNAPSHOT
        2) org.jboss.micro:kernel:jar:2.0.0-beta-SNAPSHOT
                                                                 <= who needs it

----------
1 required artifact is missing.

for artifact:
  org.jboss.micro:aop-mc-int:jar:2.0.0-beta-SNAPSHOT

from the specified remote repositories:
  central (http://repo1.maven.org/maven2),                       <= where Maven looked
  jboss (http://repository.jboss.com/maven2)

[INFO] ------------------------------------------------------------------------
[INFO] For more information, run Maven with the -e switch
[INFO] ------------------------------------------------------------------------
[INFO] BUILD ERRORS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8 seconds
[INFO] Finished at: Mon Mar 12 00:26:53 CDT 2007
[INFO] Final Memory: 7M/13M
[INFO] ------------------------------------------------------------------------
```

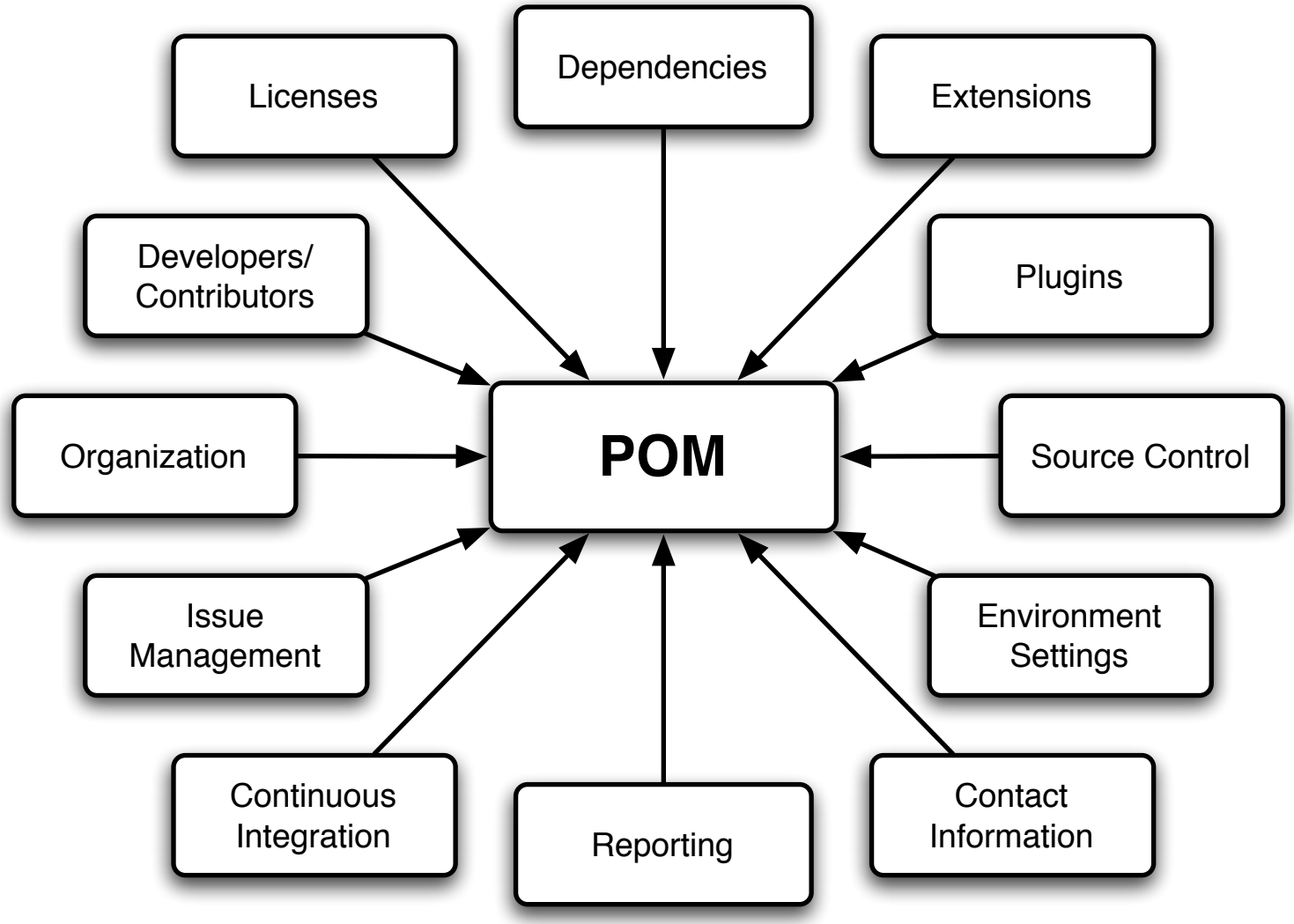
Zen Questions

- Why are we here?
- What is a project?
- What is Maven?
- What is good?
- What is the sound of one hand clapping?
- **So.... what is a project?**

The POM

- Project Object Model
- Declarative definition
  - Size of the POM not necessarily correlated to task amount
- Objectifies a collection of project files as a single unit with a single artifact output
- Defines how projects are related to each other (even transitively)
- Contains physical *and* conceptual project information
  - Build lifecycle hints, Plugin configurations
  - Developers involved
- Convention over configuration
  - Our goal is to pre-configure for 95% of use-cases

The POM

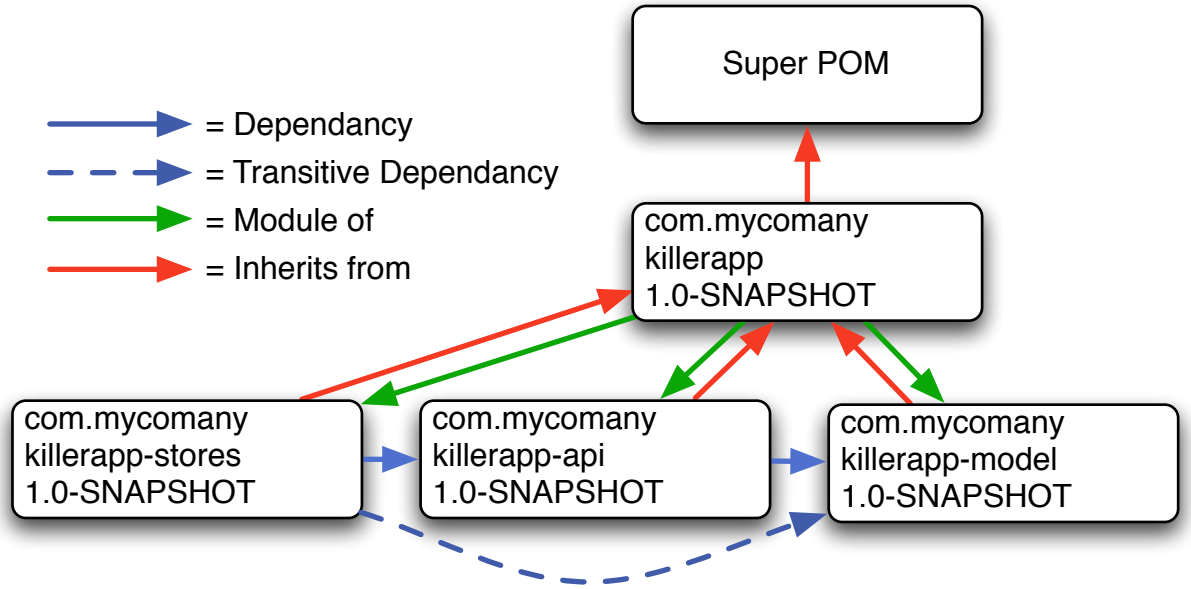

Licenses
Dependencies
Extensions
Developers/ Contributors
Plugins
Organization
POM
Source Control
Issue Management
Environment Settings
Continuous Integration
Reporting
Contact Information

The POM: Sample

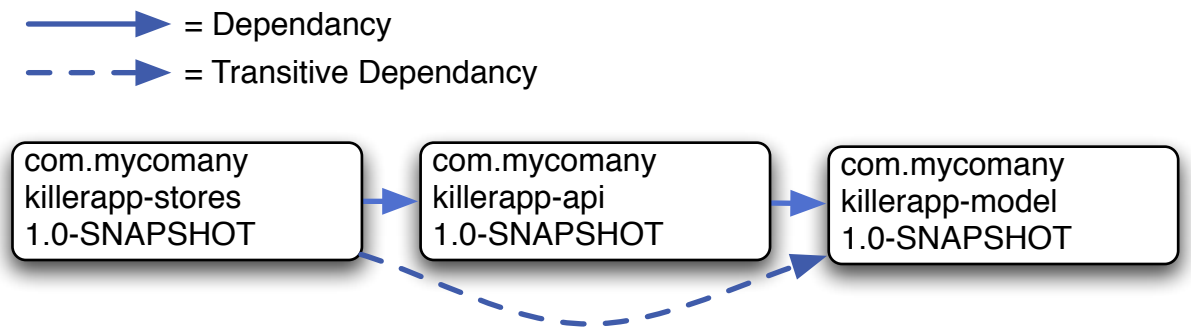
```
<project>
  <groupId>com.company-x</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-beta-2</version>
  <packaging>jar</packaging>
  <name>My JAR Application</name>
  <dependencies>
    <dependency>
      <groupId>org.opensource-y</groupId>
      <artifactId>parent-app-y</artifactId>
      <version>3.2.1</version>
    <dependency>
  </dependencies>
</project>
```

Just set it and forget it!

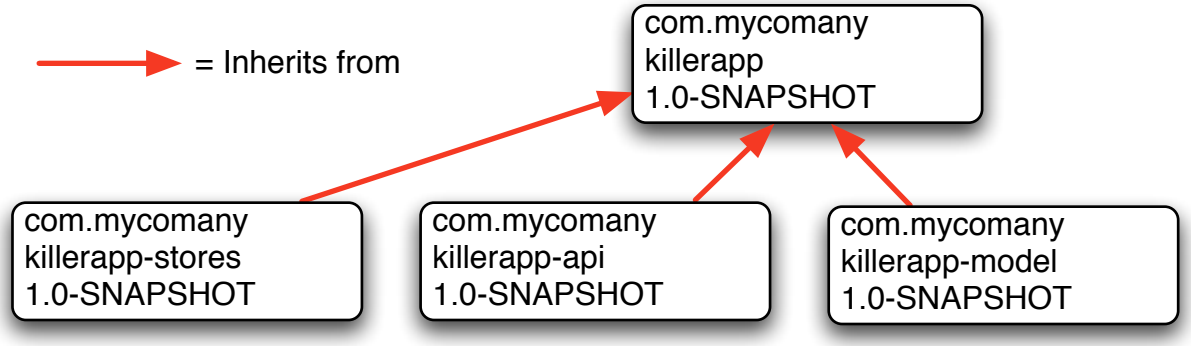
Project Relationships



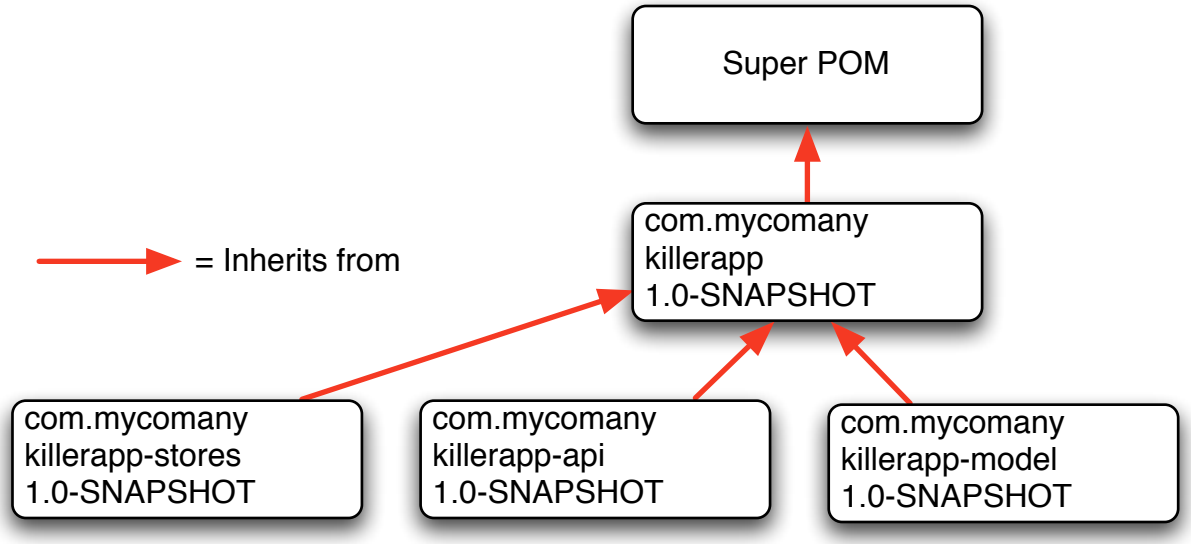
Dependencies



Inheritance



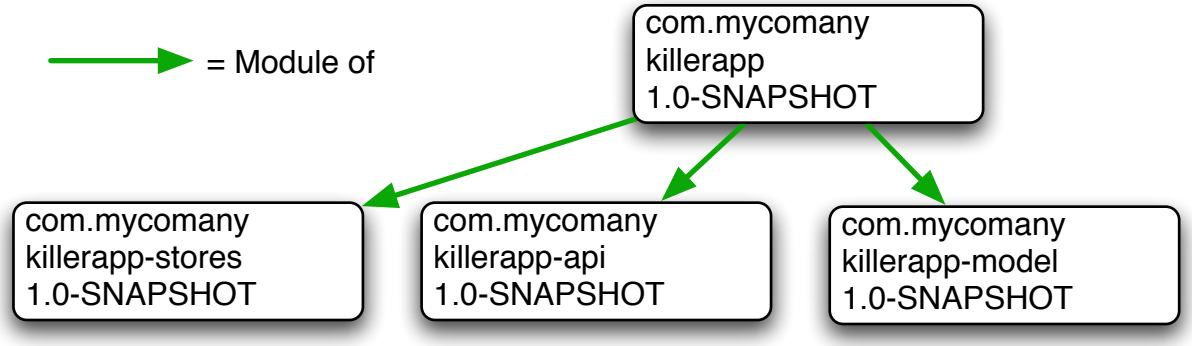
Inheritance: Super POM



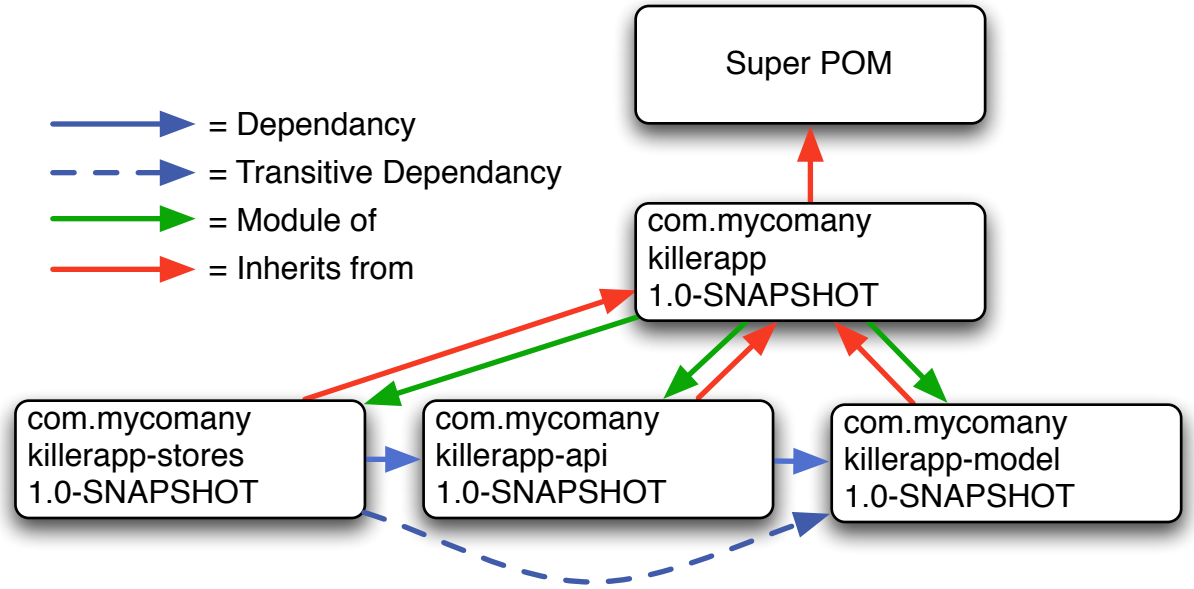
Super POM

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <name>Maven Default Project</name>

  <repositories>
    <repository>
      <id>central</id>
      <name>Maven Repository</name>
      <layout>default</layout>
      <url>http:// repo1.maven.org/maven2</url>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
    </repository>
  </repositories>

  <pluginRepositories>
    <pluginRepository>
      <id>central</id>
      <name>Maven Plugin Repository</name>
      <url>http:// repo1.maven.org/maven2</url>
      <layout>default</layout>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
      <releases>
        <updatePolicy>never</updatePolicy>
      </releases>
    </pluginRepository>
  </pluginRepositories>

  <build>
    <directory>target</directory>
    <outputDirectory>target/classes</outputDirectory>
    <finalName>${artifactId}-${version}</finalName>
    <testOutputDirectory>target/test-classes</testOutputDirectory>
    <sourceDirectory>src/main/java</sourceDirectory>
    <scriptSourceDirectory>src/main/scripts</scriptSourceDirectory>
    <testSourceDirectory>src/test/java</testSourceDirectory>
    <resources>
      <resource>
        <directory>src/main/resources</directory>
      </resource>
    </resources>
    <testResources>
      <testResource>
        <directory>src/test/resources</directory>
      </testResource>
    </testResources>
  </build>

  <reporting>
    <outputDirectory>target/site</outputDirectory>
  </reporting>

  <profiles>
    <profile>
      <id>release-profile</id>
      <activation>
        <property>
          <name>performRelease</name>
          <value>true</value>
        </property>
      </activation>

      <build>
        <plugins>
          <plugin>
            <inherited>true</inherited>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-source-plugin</artifactId>
            <executions>
              <execution>
                <id>attach-sources</id>
                <goals>
                  <goal>jar</goal>
                </goals>
              </execution>
            </executions>
          </plugin>

          <plugin>
            <inherited>true</inherited>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-javadoc-plugin</artifactId>
            <executions>
              <execution>
                <id>attach-javadocs</id>
                <goals>
                  <goal>jar</goal>
                </goals>
              </execution>
            </executions>
          </plugin>

          <plugin>
            <inherited>true</inherited>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-deploy-plugin</artifactId>
            <configuration>
              <updateReleaseInfo>true</updateReleaseInfo>
            </configuration>
          </plugin>
        </plugins>
      </build>
    </profile>
  </profiles>
</project>
```

Multi-modules



Project Relationships



The POM Details: Artifacts

- Artifacts are the output of plugin goal actions on project files
  - These are JARs, WARs, etc… but can just as easily be a directory of class files
  - Ultimately your build should shoot for 1 artifact
- Best practice
  - If your project creates more than one artifact, break it up!

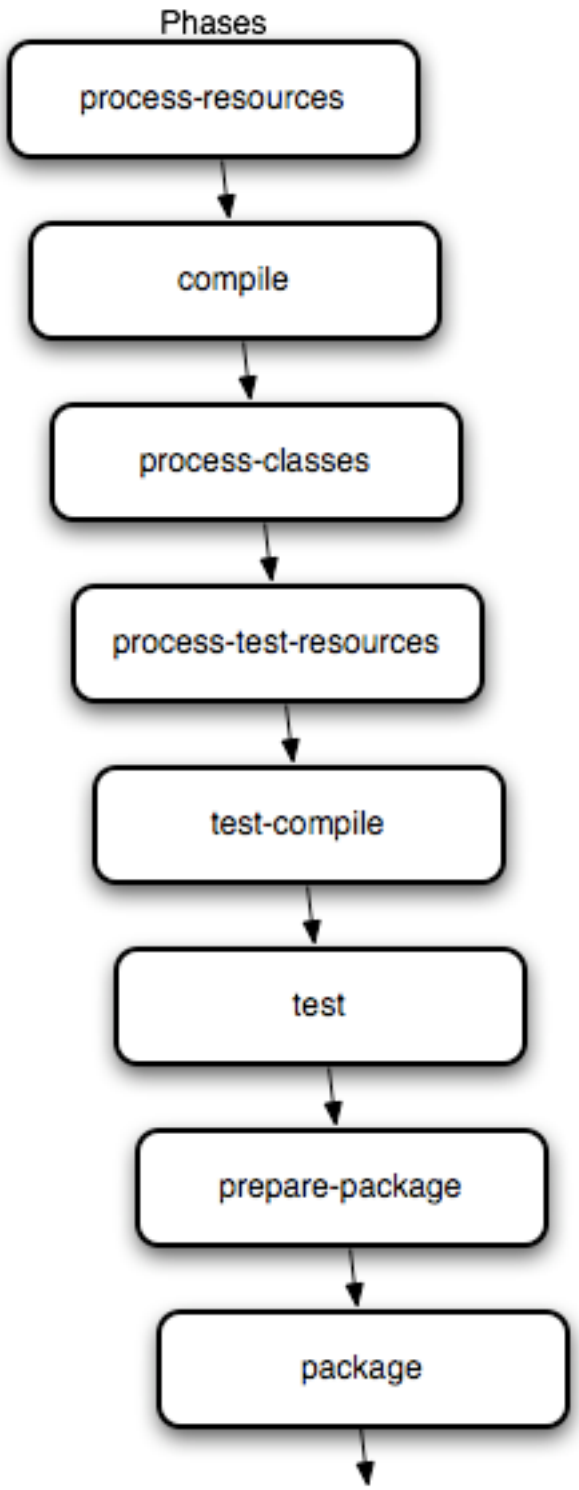
Plugins Perform Actions

- Maven is built as a plugin execution framework
- The unit of work in a plugin is a *goal*
  - Actions are declared and self-contained
- Plugins are applied to a Maven project
  - Projects and artifacts correspond to **nouns**;
    Plugins goals correspond to **verbs**.
    - **Compile** the project's **Java files**
    - **Package** the **classes**
    - **Run** the **unit tests**
- Plugins are concerned with the *how*, so project management can concern itself with the *what*.

Action Details: Goal Execution

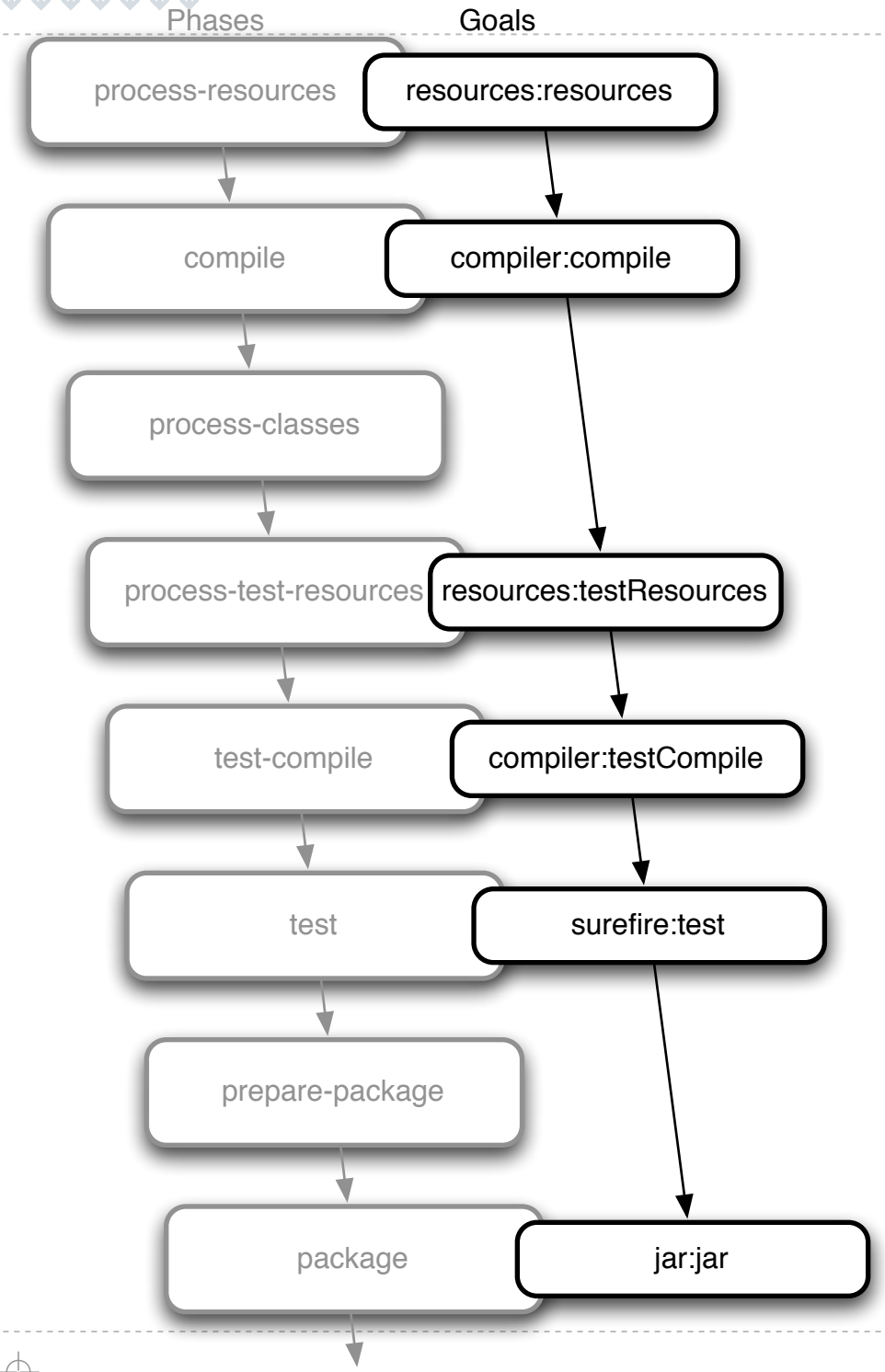
```
mvn compiler:compile
```

plugin goal



Action Details: The Build Lifecycle

- A sequence of goal executions
- Pre-defined by POM's packaging type
- Phases are configurable
- Can create custom types – ergo, custom lifecycle configurations
- Goals are bound to phases


Phases
Goals
process-resources
resources:resources
compile
compiler:compile
process-classes
process-test-resources
resources:testResources
test-compile
compiler:testCompile
test
surefire:test
prepare-package
package
jar:jar

Action Details: Running Phases

- Running a phase will execute all phases up to and including the one specified for the correct packaging type's build lifecycle definition

Action Details: Running Phases

- Running

```
mvn test
```

no colon (only individual goals have colons)

phase

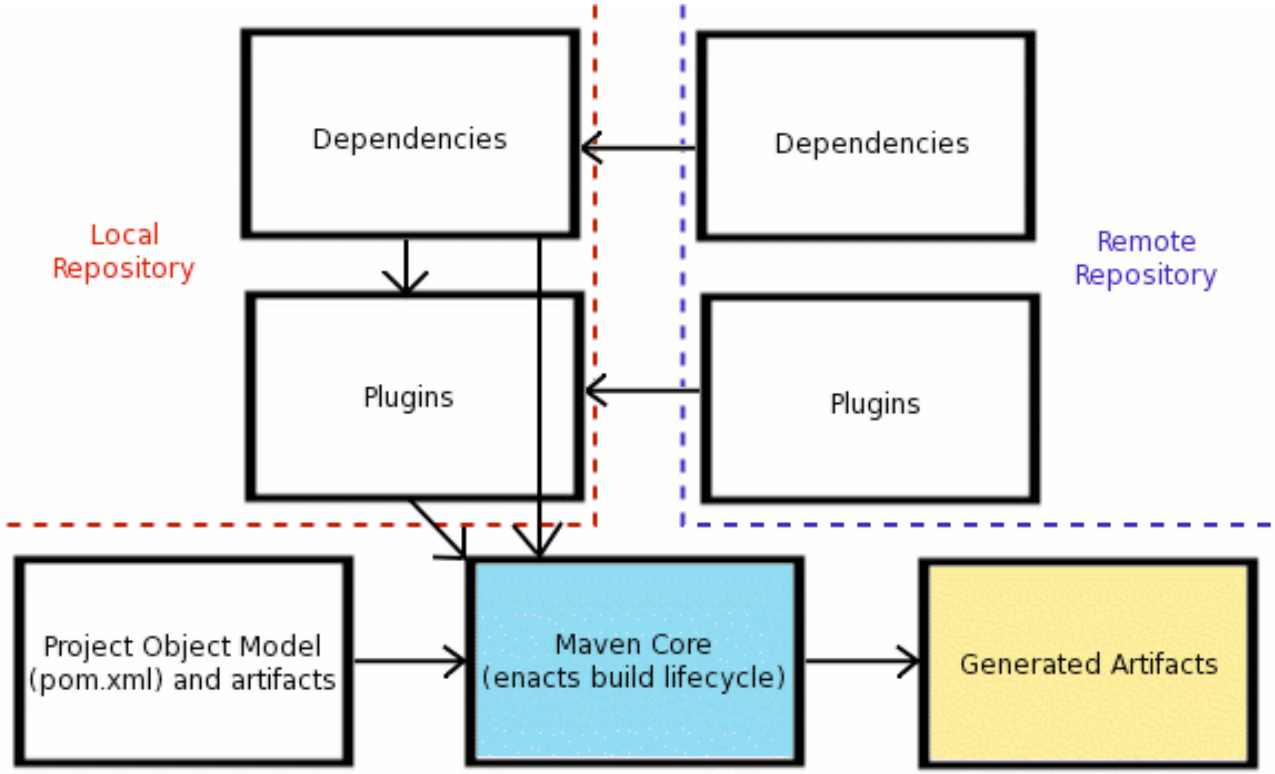
- Actually runs:

Phases:	Goals:
`process-resources`	`resources:resources`
`compile`	`compiler:compile`
`process-test-resources`	`resources:testResources`
`test-compile`	`compiler:testCompile`
`test`	`surefire:test`

The Repository: It's not Magic

- Maven organizes and makes available dependencies and project tools for you – just ask
- Say what you need; not where/how to get it
  - Dependencies in Maven are requested in a declarative fashion
- Artifacts and Repositories
  - Remote repository is for the portability of dependencies
  - Local repository is a developer's personal cache of downloaded or installed dependencies

From POM to Artifact (using Repository)



Questions?

More Reading

- Web
  - http://maven.apache.org/guides/
  - http://maven.apache.org/articles.html
- Examples
  - http://s3.amazonaws.com/maven2/index.html
- Books
  - Better Builds with Maven (http://devzuz.com)
  - Maven: The Definitive Guide (http://sonatype.com/book)
  - Java Power Tools (John Smart – Dec. 2007)
- Me
  - http://blog.propellors.net