Dive Into Maven

Growing your own POMs

Miquel Martin

contact@miquelmartin.org

Maven?

- It’s a Software Build tool
- Like Makefile, Ant, Jam

- Differences:
  - It knows what you want to do
  - Only asks you to intervene if you deviate from “the standard”

Maven Golden rule

In Maven, what you don’t see, is a default

Diving in: your base project

```
mvn archetype:generate
-DarchetypeGroupId=org.apache.maven.archetypes
-DarchetypeArtifactId=maven-archetype-quickstart
```

- Creates your base project structure
- Ignore the pom.xml file, we'll make our own

Our first POM!

The Basic POM (Project Object Model)

```
<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/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>plants</groupId>
  <artifactId>leaf</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <dependencies>
     <dependency>
       <groupId>junit</groupId>
       <artifactId>junit</artifactId>
       <version>3.8.1</version>
       <scope>test</scope>
     </dependency>
  </dependencies>
</project>
```

Maven coordinates

This POM alone can…

- Compile your code
- Run tests
- Deploy to your repository
- Generate documentation

- Try:
  - `mvn package`
  - `mvn site`
  - `mvn clean`

So, what's happening really?

- 3 build lifecycles
  - clean: erase everything in target
  - default: where the interesting things happen
  - site: build an info site

For further info: http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

Build cycle details

- It’s divided into phases (abridged version):
  - validate – project sanity check
  - compile - compile the source
  - test – run tests (e.g. junit)
  - package – put it into a jar, war, or similar
  - verify – sanity on the package + quality
  - install – copy it to the local maven repository
  - deploy – copy to remote repository

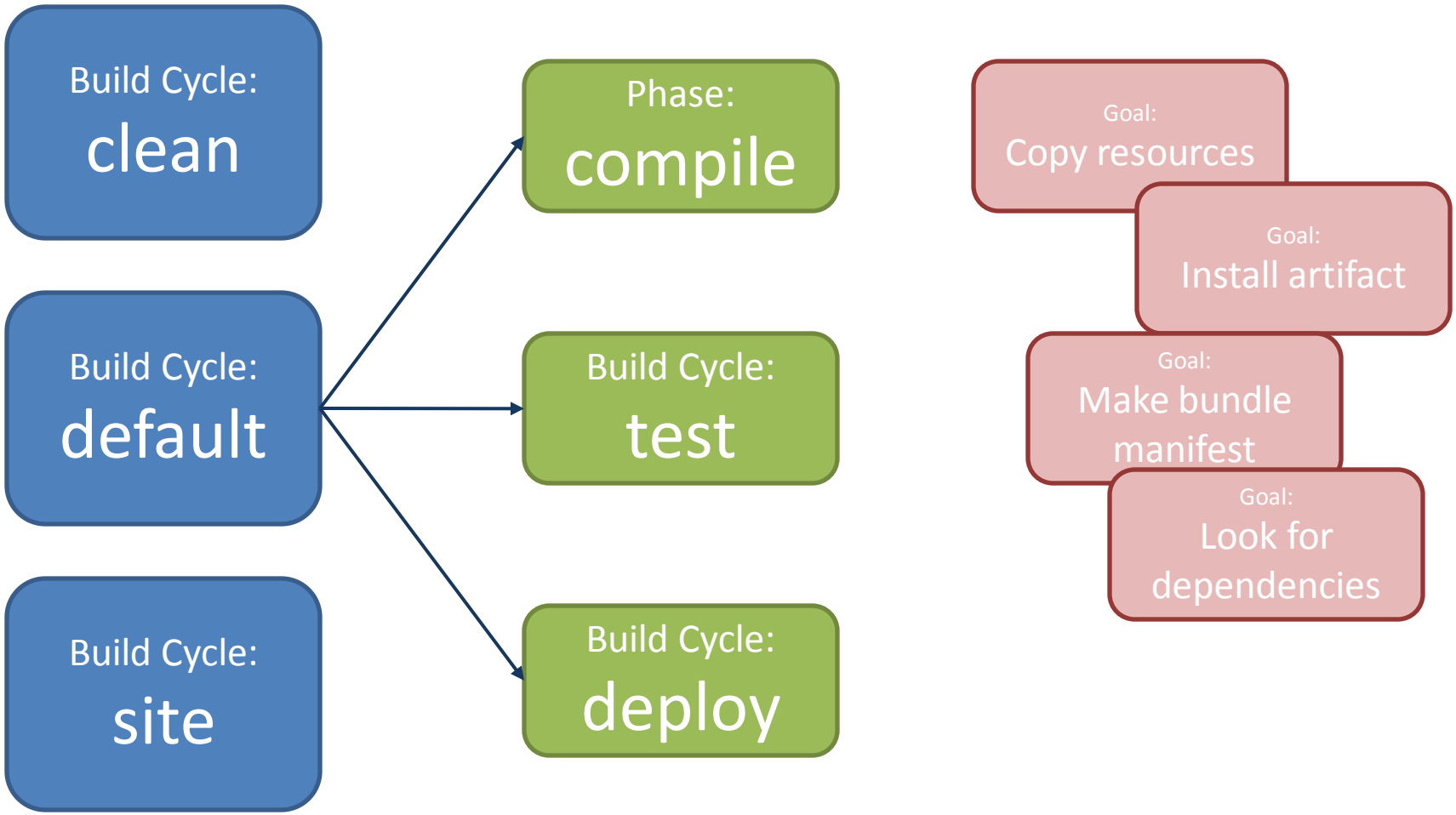
Phases

- Phases are run in order. Calling package calls everything before
- Phases just empty drawers, nothing happens in them

Goals

- For things to happen, we need to put actions into the phases. These actions are “goals”
  - E.g.: create a MANIFEST.MF file, compress a jar, process declarative service annotations, etc…

Recap


Build Cycle:
clean
Build Cycle:
default
Build Cycle:
site
Phase:
compile
Build Cycle:
test
Build Cycle:
deploy
Goal:
Copy resources
Goal:
Install artifact
Goal:
Make bundle manifest
Goal:
Look for dependencies

Binding goals to Phases

- In maven, there are (of course) default bindings. They are called “packaging” types

- E.g.: jar, bundle, pom, war, ejb…

Example: bindings for Jar packaging

- process-resources: resources:resources
- compile: compiler:compile
- process-test-resources: resources:testResources
- test-compile: compiler:testCompile
- test: surefire:test
- package: jar:jar
- install: install:install
- deploy: deploy:deploy

Notes:

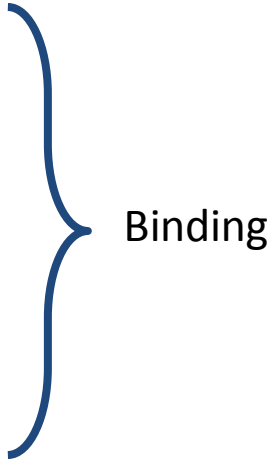
- The format is: phase: Plugin:goal
- There are more phases than we really mentioned

Now you know

- That’s why your little pom could do so much

Configuring plugins

```
<project…>
…
<build>
      <plugins>
        <plugin>
          <groupId>com.mycompany.example</groupId>
          <artifactId>display-maven-plugin</artifactId>
          <version>1.0</version>
          <executions>
            <execution>
              <phase>process-test-resources</phase>
              <goals>
                <goal>time</goal>
              </goals>
            </execution>
          </executions>
        </plugin>
      </plugins>
</build>
….
</project>
```



A more elaborate example

- Example for maven-bundle-plugin (see bundleExamples)

1 Pom, 2 Poms!

POM inheritance

Let’s distinguish

- A POM that compiles other POMs:
  parents pointing at their children

- Inherit POM properties:
  children point at parents

A POM that compiles other POMs: The Leaf and its gardener

- Using <modules>
- See bundleExamples

Inherit POM properties:

- The SuperPOM!
- Inheriting from not-so-cool POMs, using <Parent>. See parentExamples
- Dependency Management

A setup proposal

- parent: the default pom
- builder: a pom that includes all modules
- module1
- …
- moduleN

About repositories

- About repositories:
  - Local
  - Company Proxy
  - Remote
- Normal VS SNAPSHOT versions
  - E.g.: 1.0.0 VS 1.0.0-SNAPSHOT

For next time

- Assembly plugin: copy all bundles to a release folder
- Maven-dependency-plugin: copy dependencies of bundles to a release folder

Thanks! Questions?

Special thanks to Benjamin Hebgen, Jochen Bauknecht and Julien Poumailloux. Learning in the team totally beats stackoverflow!