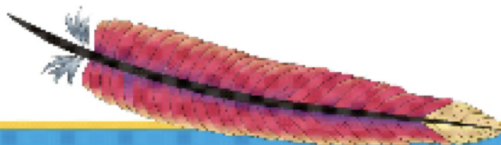


Apache Maven: Best Practices

Brett Porter - brett@apache.org

<http://www.devzuz.org/blogs/bporter>



Maven without the PAIN

- Sometimes unpleasant
- It's for your own good!
- Can avoid or alleviate the problems



Who am I?

maven

Apache Maven
developer since 2003



Co-author of
Better Builds with Maven

archiva

Started the Archiva
project



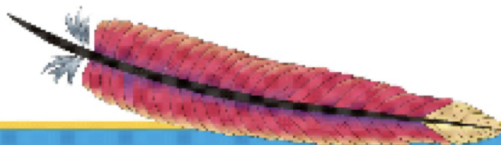
Co-founder of
DevZuz

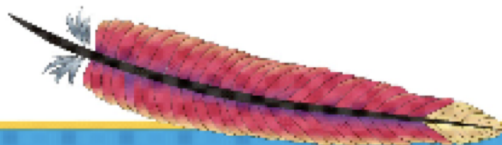


Getting Ready for Maven

Why are you using Maven?

- Consider this from the beginning
- Right tool for the job
- Ensure you reap the benefits





Planning Ahead

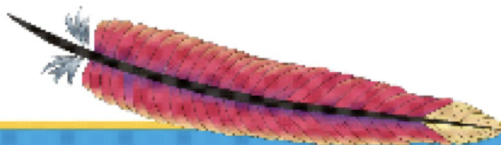
Preparing Your Environment

Development Environment

- Maven is best used in the bigger picture
- Plan for the infrastructure you will use

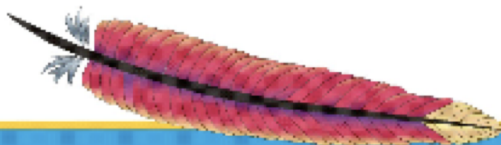
```
<scm>
  <connection>...
  <developerConnection>...
  <url>...
</scm>
```

```
<issueManagement>
  <url>...
  <system>...
</issueManagement>
```



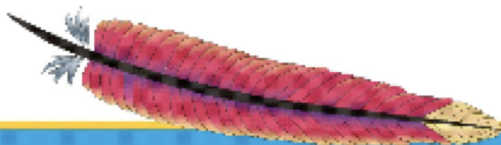
Automated Build Servers

- Maven is intended to be automated
- More than continuous integration



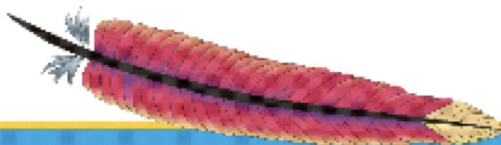
Repository Management

- Centralise storage of artifacts
 - Store the artifacts you build
 - Store third-party artifacts you consume



Repository Servers

- Can start with a simple storage
- Servers can help manage the artifacts



Using Repositories

```
<repository>
  <id>apache-snapshots</id>
  <name>Apache Snapshots Repository</name>
  <url>http://people.apache.org/repo/m2-snapshot-repository</url>
  <releases>
    <enabled>>false</enabled>
  </releases>
</repository>

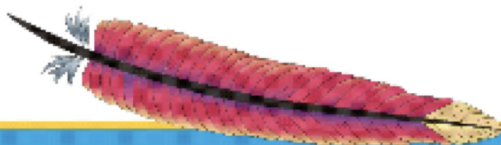
<repository>
  <id>java-net-m1</id>
  <name>Java.net Repository</name>
  <url>http://download.java.net/maven/1/</url>
  <layout>legacy</layout>
  <snapshots>
    <enabled>>false</enabled>
  </snapshots>
</repository>
```

Using Repositories

```
<repository>
  <id>apache-releases</id>
  <name>Apache Releases Repository</name>
  <url>http://people.apache.org/repo/m2-ibiblio-rsync-
  repository</url>
  <snapshots>codehaus-snapshots</snapshots>
  <enabled>false</enabled>
  </repository>
  <repository>
    <id>codehaus-snapshots</id>
    <name>Codehaus Snapshots Repository</name>
    <url>http://snapshots.repository.codehaus.org/</url>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
    <releases>
      <enabled>false</enabled>
    </releases>
  </repository>
  <repository>
    <id>codehaus-releases</id>
    <name>Codehaus Releases Repository</name>
    <url>http://repository.codehaus.org/</url>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
    <releases>
      <enabled>true</enabled>
    </releases>
  </repository>
  <repository>
    <id>java-net-m1</id>
    <name>Java Net Maven 1 Repository</name>
    <url>http://download.java.net/maven/1/</url>
    <layout>legacy</layout>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
    <releases>
      <enabled>true</enabled>
    </releases>
  </repository>
  <repository>
    <id>java-net-m2</id>
    <name>Java Net Maven 2 Repository</name>
    <url>http://download.java.net/maven/2/</url>
    <layout>default</layout>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
    <releases>
      <enabled>true</enabled>
    </releases>
  </repository>
</repositories>
```

Using Repositories

- Minimise the number of repositories
- Only in the POM if you redistribute
- Use repository manager to centralise



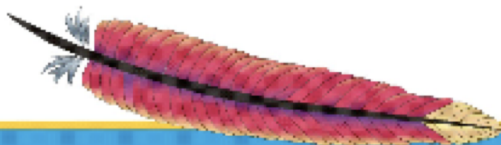
Locking Down

```
<mirrors>
  <mirror>
    <id>archiva-snapshots-apache</id>
    <url>http://reposer/archiva/repository/snapshots/</url>
    <mirrorOf>apache.snapshots</mirrorOf>
  </mirror>
  <mirror>
    <id>archiva-snapshots-codehaus</id>
    <url>http://reposer/archiva/repository/snapshots/</url>
    <mirrorOf>codehaus.snapshots</mirrorOf>
  </mirror>
  ...
  <mirror>
    <id>archiva-default</id>
    <url>http://reposer/archiva/repository/internal/</url>
    <mirrorOf>*</mirrorOf>
  </mirror>
</mirrors>
```



Settings and Installation

- Three levels of configuration
 - project (pom.xml)
 - user (~/.m2/settings.xml)
 - installation (<maven>/conf/settings.xml)

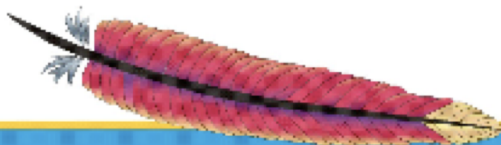


Keeping it Simple

Creating New Projects

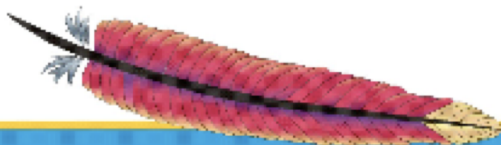
Archetypes

- Standard project layouts
- Include organisational POM
- Facilitates consistency



Writing the POM

- Write the build like you write code
- Utilise conventions
- Use multiple modules



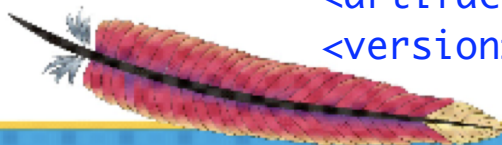
Inheritance

- Multi-module inheritance

```
<parent>
  <groupId>org.apache.maven</groupId>
  <artifactId>maven</artifactId>
  <version>2.1-SNAPSHOT</version>
</parent>
<artifactId>maven-core</artifactId>
```

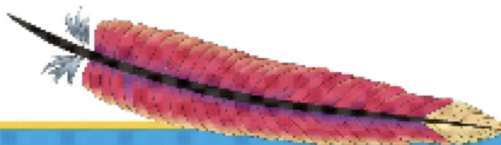
- Organisational POM hierarchy

```
<parent>
  <groupId>org.apache</groupId>
  <artifactId>apache</artifactId>
  <version>3</version>
</parent>
<groupId>org.apache.maven</groupId>
<artifactId>maven</artifactId>
<version>2.1-SNAPSHOT</version>
```



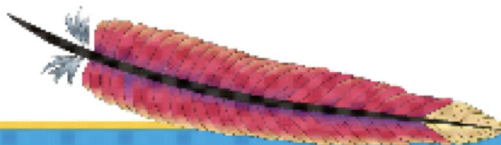
Dependencies

- Specify only what you need
- Specify scope
- Use dependencyManagement to:
 - coerce Maven to use a particular version
 - enforce consistency within a project



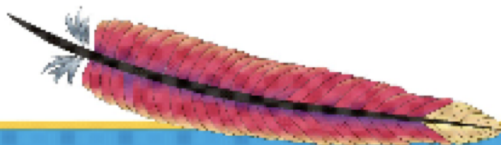
Build Pipeline

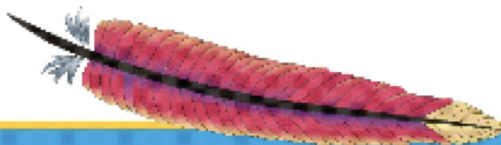
- Depends on your team
- Use profiles for controlling complexity
- Applies to testing as well
- Key is to keep the build fast



Scripting

- Maven is declarative by design
- Integrate scripting if necessary
- Consider writing plugins



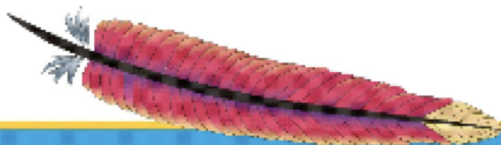


The D Word

Documenting as You Go

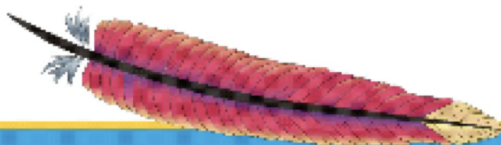
Documentation

- Document the build
- Developer documentation
 - separate from product documentation
 - keep it together with reports



Sites and Reports

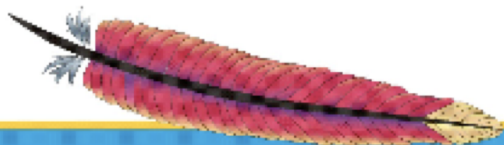
- A whole other topic of best practices!
- But apply the same principles
 - set up what you'll actually use
 - set up enforcement checks, not just reports





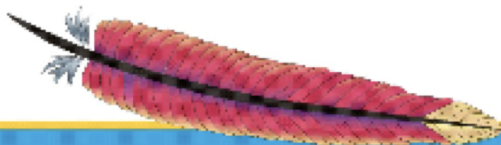
Portability

Playing with Others

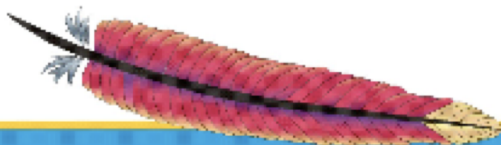


The Goal

- When a new developer builds the project
 - it works first go
 - ... and it keeps working

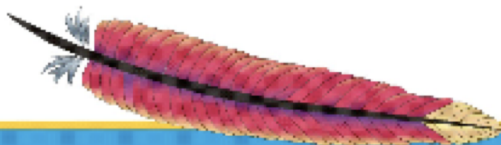


maven.test.skip
is evil



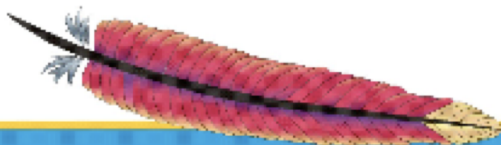
Hard Coding

- Don't hard code **paths**
- Don't hard code **databases**
- Don't hard code **properties**
- Don't do it in the **tests** either

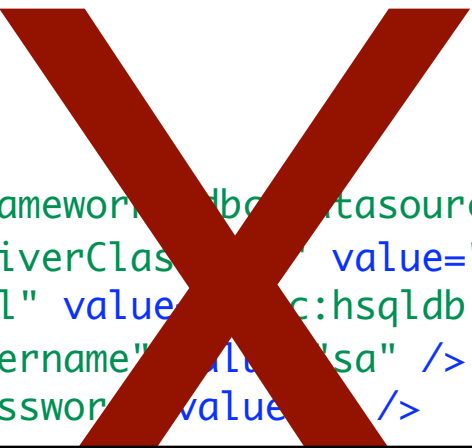


Profiles

- Very useful - but don't abuse them
- Document them all
- Avoid depending on the environment

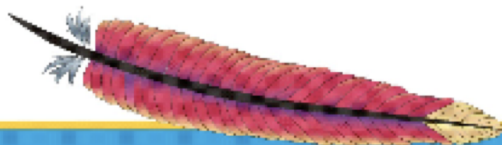


Portable Artifacts



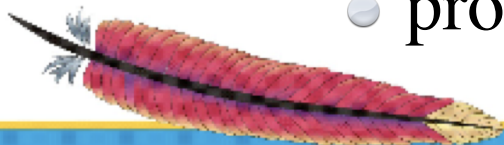
```
<bean
  id="dataSource"
  class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClass" value="org.hsqldb.jdbcDriver" />
    <property name="url" value="jdbc:hsqldb:database" />
    <property name="username" value="sa" />
    <property name="password" value="" />
  </bean>
```

```
value="org.hsqldb.jdbcDriver"
value="jdbc:hsqldb:database"
```



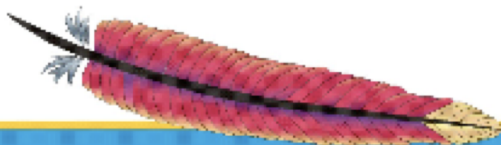
Portable Artifacts

- Artifacts in repository must be unique
- Either by classifier, or being portable
- Recommend externalising configuration
 - database
 - target environment
 - properties



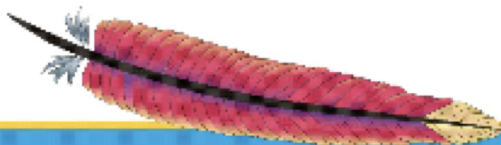
Resource Filtering

- Use with great care!
- Centralisation, not substitution
 - ✓ `google.analytics.code=UA-1234567-1`
 - X `database.username=admin`
- Useful for once-off alterations
- Consider externalising configuration



Shared Resources

- Don't duplicate resources across projects
- ZIP and put in the repository
- Use the dependency plugin to retrieve

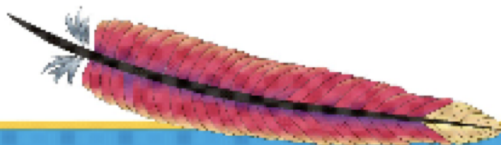


Reproducibility

Ensuring Reproducible Builds

Reproducibility

- Important for releases
- More important for source releases
- Build must be isolated from change
- Portability is a pre-requisite



The Enforcer

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-enforcer-plugin</artifactId>
  <version>1.0-alpha-4</version>
  <executions>
    <execution>
      <goals>
        <goal>enforce</goal>
      </goals>
      <configuration>
        <rules>
          <requirePluginVersions>
            <banLatest>true</banLatest>
            <banRelease>true</banRelease>
          </requirePluginVersions>
        </rules>
      </configuration>
    </execution>
  </executions>
</plugin>
```

...



The Enforcer

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-enforcer-plugin</artifactId>
  <version>1.0-alpha-4</version>
  <executions>
    <execution>
```

```
    <rules>
      <requirePluginVersions>
        <banLatest>true</banLatest>
        <banRelease>true</banRelease>
      </requirePluginVersions>
    </rules>
```

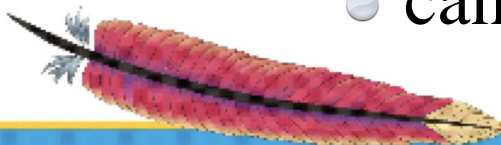
```
</configuration>
```

...



The Enforcer

- Help ensure build will be reproducible
- Based on rules
 - force specific plugin versions
 - ban snapshots
 - global exclusions
 - force Maven/Java/OS version
 - can write your own



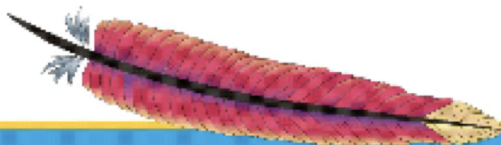


Releasing

Release Early, Often and Right

Releases

- Set the project version to a -SNAPSHOT
- Make them early and often
- Use Maven tools to automate



Questions?

Brett Porter - brett@apache.org

Better Builds with Maven, blog at

<http://www.devzuz.org/>

