Plugin migration to Maven3 dependencies

<table>
<thead>
<tr>
<th>Status</th>
<th>WIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Issue(s)</td>
<td></td>
</tr>
<tr>
<td>Sources</td>
<td></td>
</tr>
<tr>
<td>Developer(s)</td>
<td></td>
</tr>
</tbody>
</table>

The Apache Maven team announced a while ago to stop supporting Maven2. Now it is time for the plugins to follow. This pages describes the topics on what to do to make plugin Maven3+ compliant.

**Howto**

1. Set the Maven prerequisite and dependencies to 3.0, which might cause one or more of the following issues:
   a. Unknown dependencies in the pom.xml, e.g. the maven-project artifact doesn't exist anymore: its content is now part of maven-core
   b. CNFE. Some classes have been moved to maven-compat. You could add it temporarily to get it all working again, later you'll get the instructions how to get rid of it.
2. Set both the maven.compiler.source and maven.compiler.target properties to 1.7
3. If the project depends on maven-plugin-testing-harness, use version 2.1, since newer versions depend on newer version of Maven dependencies.
   Ensure to exclude the maven-2.x dependencies which don’t exist anymore, e.g. maven-project
4. Remove maven-compat (or give it the test-scope if it is required by the maven-plugin-testing-harness) Maven compat mainly contains business logic which has been moved to Aether. However, we need to support the implementations of both Sonatype (M3.0.x) and Eclipse (M3.1+). The shared component maven-artifact-transfer can select the proper implementation.
5. Transitive dependencies may still refer to Maven-2.x artifacts, even if they don’t exist for Maven 3.x any more: ensure these are excluded. You can enforce this by adding the requireSameVersions enforcer rule to your project.
6. Reporting plugins should include org.apache.maven.reporting:maven-reporting-api:3.0

There is a report generated daily with plugins prerequisites.

**Housekeeping**

While at it, some housekeeping can be done:

1. Rename package to org.apache.maven.plugins.xxx (matching the groupId, not conflicting with maven-core org.apache.maven.plugin package)
2. Resolve reflection usage, meant to call Maven3 specific code when available.
3. Remove deprecated parameters, methods, goals, classes
4. Remove unnecessary overloaded methods. Some methods have been extended with arguments to stay backwards compatible. Time to clean this up (if the number of arguments is still uncertain, use a Request/Result object containing the argument objects, so the method signature will stay the same).

**Plugins followup**

The following plugins already use successfully without maven-compat and with maven-artifact-transfer:

- maven-assembly-plugin (in trunk, with maven-compat in test scope)
- maven-deploy-plugin (https://svn.apache.org/repos/asf/maven/plugins/branches/m-deploy-p-3.0 )
- maven-install-plugin (https://svn.apache.org/repos/asf/maven/plugins/branches/m-install-p-3.0 )
- maven-shade-plugin (https://svn.apache.org/repos/asf/maven/plugins/branches/m-shade-p-3.0 )

Maven Shared Components needed to be upgraded:

- maven-shared-utils (3.0.0)
- maven-archiver (3.0.0)
- maven-filtering (3.0.0)
- maven-jarsigner (3.0.0)...(maven-shared-utils)
- maven-common-artifact-filters (maven-shared-utils)
- maven-mapping (3.0.0)
- maven-artifact-transfer (3.0.0) (maven-common-artifact-filters)
- maven-shared-io (3.0.0)
- file-management (3.0.0) (maven-shared-io)
- maven-repository-builder (maven-shared-utils maven-common-artifact-filters)

more of them?
Known plugins still to do (there are probably more):

- maven-surefire-plugin
- maven-ejb-plugin (uses maven-archiver \land maven-filtering \land maven-resources-plugin (uses maven-filtering)
- maven-war-plugin (maven-archiver 3.0.0 \land maven-filtering 3.0.0 \land maven-mapping 3.0.0)
- maven-ear-plugin (maven-archiver 3.0.0 \land maven-filtering 3.0.0)
- maven-jarsigner-plugin (maven-shared-utils 3.0.0 \land maven-jarsigner 3.0.0)
- maven-toolchains-plugin
- maven-assembly-plugin (maven-archiver \land maven-filtering \land maven-common-artifacts-filter \land maven-repository-builder \land maven-shared-utils \land file-management)
- maven-verifier-plugin (works so far so good. JUnit should be updated)

more of them?

The plugins which have been lifted to 3.0.0,

- maven-clean-plugin (has been released as 3.0.0).
- maven-ear-plugin (has been released as 3.0.0).
- maven-source-plugin (has been released as 3.0.0).

Stuff that may have changed

Testcases with this code:

```java
container = new DefaultPlexusContainer();
container.initialize(); // This statement can be removed for 3.0
container.start(); // This statement can be removed for 3.0
```

Intermittently (and while upgrading) you may find code like this:

```java
metadataSource = (ArtifactMetadataSource) lookup(ArtifactMetadataSource.ROLE);
```

While you are still working with maven-compat, you can fix this by switching to lookup by type;

```java
metadataSource = (ArtifactMetadataSource) lookup(ArtifactMetadataSource.class);
```

### Deprecation mappings

<table>
<thead>
<tr>
<th>Old</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.maven.project.MavenProjectBuilder</td>
<td>org.apache.maven.project.ProjectBuilder</td>
</tr>
<tr>
<td>Some adoptions necessary, some reflection to stay compatible with 2 Aether versions. See <a href="http://svn.apache.org/viewvc?view=revision&amp;revision=1685177">http://svn.apache.org/viewvc?view=revision&amp;revision=1685177</a> for sample</td>
<td></td>
</tr>
<tr>
<td>org.apache.maven.artifact.factory.ArtifactFactory</td>
<td>org.apache.maven.repository.RepositorySystem (however, there's only one implementation: LegacyRepositorySystem, which is again part of maven-compat AND reuses ArtifactFactory)</td>
</tr>
<tr>
<td>Depending on what you want to do next with the Artifact, you could use TransferUtils.toArtifactCoordinate() instead.</td>
<td></td>
</tr>
<tr>
<td>org.apache.maven.artifact.repository.ArtifactRepositoryFactory</td>
<td>org.apache.maven.artifact.repository.MavenArtifactRepository</td>
</tr>
</tbody>
</table>
Improved separation of dependencies and artifacts

It won't be possible any more to get the dependencies based on an artifact, since an artifact is basically a reference to a file with a coordinate. Instead you must use a dependency, plugin, extension or other model entity to resolve the dependencies. See http://maven.apache.org/shared-archives/maven-artifact-transfer-LATEST/comparison.html for a matrix.