Maven 2.1 Backward Compatibility Information

Plugin and Extension Loading

Maven's mechanisms for loading plugins and build extensions has been refactored. You can find more information in the Maven 2.1 Plugin and Extension Loading Design document.

Lifecycle

Aggregator Mojo Handling

Aggregator mojos bound to the lifecycle have been deprecated. This practice can produce some very strange results, and isn't really the right solution for many of the problems it attempts to solve. I'm hoping to include some better options for bracketing the normal build - both before, and after, explicitly - to make aggregator mojos obsolete, but for now they've been deprecated to avoid disrupting backward compatibility.

Also, aggregator mojos that **are** bound to the lifecycle will only be allowed to execute at most once during the build, to limit redundant execution. These mojos are meant to act on all projects in the reactor at once, and binding them to one pom.xml file is dangerous in that it can produce different build results depending on whether that pom.xml is included. This is further complicated if two modules in a reactor configure the same aggregator mojo...in which case, it may run multiple times...or, when the aggregator is configured in the parent pom, where it will run for each descendant module.

Plugin API

Shading of plexus-utils causing a ClassCastException on plugin.getConfiguration() (MNG-3012)

The fact that plexus-utils is hidden from plugins in the newer releases of Maven means that plugin.getConfiguration() from maven-model can cause a ClassCastException, if used from within a mojo. The plan to fix this is basically just to import Xpp3Dom from the shaded plexus-utils version in maven-core within the plugin's classrealm. This should allow us to share the same instance of that class (only, shouldn't really affect other p-u classes) and preserve backward compatibility for existing plugin releases.