Conflict Resolvers

Conflict Resolvers

Synopsis

A dependency version conflict occurs when building a Maven project whose dependency graph pulls in two different versions of the same dependency. Maven must decide which version of the conflicting dependency to use before proceeding with the build. The behaviour thus far has always been to select the version in the dependency graph that is 'nearest' to the project being built, where the distance is defined to be the number of transitive steps between two nodes in the graph. This aim of this proposal is to allow alternative conflict resolution techniques to be used when building projects with Maven.

Design

The existing but unused ConflictResolver interface will become the basis for the conflict resolver API. The following method will be added to handle conflict resolutions:

```
/**
 * Determines which of the specified versions of an artifact to use when there are
conflicting declarations.
 *
 * @param node1 the first artifact declaration
 * @param node2 the second artifact declaration
 * @return the artifact declaration to use: node1; node2; or null if this conflict
cannot be resolved
 */
ResolutionNode resolveConflict( ResolutionNode node1, ResolutionNode node2 );
```

The following `ConflictResolver` implementations will be provided:

Name	Strategy
nearest	Resolves conflicting artifacts by always selecting the nearest declaration. Nearest is defined as the declaration that has the least transitive steps away from the project being built.
farthest	Resolves conflicting artifacts by always selecting the farthest declaration. Farthest is defined as the declaration that has the most transitive steps away from the project being built.
newest	Resolves conflicting artifacts by always selecting the newest declaration. Newest is defined as the declaration whose version is greater according to `ArtifactVersion.compareTo`.
oldest	Resolves conflicting artifacts by always selecting the oldest declaration. Oldest is defined as the declaration whose version is less according to `ArtifactVersion.compareTo`.

ArtifactCollector will have an additional method added to accept a list of `ConflictResolvers`:

```
ArtifactResolutionResult collect( Set artifacts, Artifact originatingArtifact, Map
managedVersions,
                                  ArtifactRepository localRepository, List
remoteRepositories,
                                  ArtifactMetadataSource source, ArtifactFilter
filter, List listeners,
                                  List conflictResolvers )
    throws ArtifactResolutionException
```

Correspondingly, DefaultArtifactCollector will be updated to process the chain of `ConflictResolvers` to resolve dependency version conflicts. The chain will be processed until one `ConflictResolver` can resolve the conflict. In the event that no `ConflictResolvers` can resolve the conflict, an ArtifactResolutionException will be thrown to fail the build.

ArtifactResolver will also have a corresponding new method in order to delegate to the new `ArtifactCollector` method:

```
ArtifactResolutionResult resolveTransitively( Set artifacts,
                                              Artifact originatingArtifact,
                                              Map managedVersions,
                                              ArtifactRepository localRepository,
                                              List remoteRepositories,
                                              ArtifactMetadataSource source,
                                              ArtifactFilter filter,
                                              List listeners,
                                              List conflictResolvers )
    throws ArtifactResolutionException, ArtifactNotFoundException;
```

To ensure that the conflict resolution strategy is consistent throughout the build, all usages of `ArtifactCollector` and `ArtifactResolver` within Maven will be updated to use these new overloaded versions. The list of `ConflictResolvers` to use will be obtained from the `MavenProject` being built. In order to preserve the 2.0.x POM schema, a POM property will initially be used to configure the `ConflictResolver` chain. The POM property will be named `mavenConflictResolvers` and its value will be defined as a comma-separated list of conflict resolver names. For example:

```
<properties>
    <mavenConflictResolvers>newest,nearest</mavenConflictResolvers>
</properties>
```

For 2.1.x, it is likely that the POM schema will be modified to introduce a more appropriate syntax for configuring conflict resolvers. For example:

```
<dependencies>
    <conflictResolvers>
        <conflictResolver>newest</conflictResolver>
        <conflictResolver>nearest</conflictResolver>
    </conflictResolvers>
</dependencies>
```

For backwards compatibility, if a project does not declare any conflict resolvers then the 'nearest' conflict resolver will be used by default.