

Mercury Dependency Builder

Mercury **DependencyTreeBuilder/VirtualRepositoryReader** classes is an attempt to fully utilize the advantages that the Jetty HTTP transport and SAT-based conflict resolver give us. It's also required in order to implement the version ranges discussed in [Mercury Version Ranges](#). This page is a brainstorm design of the Dependency building algorithm.

Dependency builder is given a root GAV and it's task is to:

1. build a tree of all possible dependencies
2. resolve conflicts (remove same GAs from the tree) to produce a flat set of GAVs that constitute calculated classpath (or dependency graph in case of OSGi) for a given scope

Given that #1 is done, the algorithm is described [here](#), this page concentrates on #1 - how "dirty" dependency tree is constructed.

First, **DependencyTreeBuilder** is given a List of Repositories (Not Set - ordering matters!) and implementation of **DependencyProcessor**. **DependencyProcessor** abstracts reading and parsing of GAV metadata a.k.a. POMs. Because we don't need entire POM, just interpolated dependencies section of it, we can to try break them out of POM in the future - there was a discussion on dev@ list about it.

Then **DependencyTreeBuilder** is given an entry point (GAV) and is asked to construct the "dirty" tree. An here it starts - if, say, root is a:a:1.2.3, should we try to resolve [1.2.3,) or stick to [1.2.3,1.2.3] ? We can clear this ambiguity by always treating the root of the tree as [V,V]. But how far should that go? If I am developing a multi-module project a:a:1.2.3-SNAPSHOT, I expect my root and all project internal dependencies to be on the same version, which is 1.2.3-SNAPSHOT. I can achieve it with either explicit [V,V] range definition, or have some magic in the **DependencyTreeBuilder**. The latter is always bad as sooner or later it will be abused by smart users to do something else and possibly break the dependency tree construction. This consideration leaves only one option open - always use [V,V]. But without a tool to automatically process those (release plugin for one) - it quickly becomes a burden for users.

I would capture the above as a conclusion #1: **need a tool to manipulate constructs like [V,V] - create, rename, increment, show**

Next: for each range [1.2.3,2.0.0) **DependencyTreeBuilder** creates a set of possible candidates: 1.2.3,1.2.4,1.3.0,1.5.4. Then it treats all of them as roots of subsequent trees. If one of 1.2.4 dependencies is missing - entire 1.2.4 sub-tree be eliminated from resulting tree.

We can also do a little better optimization here: once a tree level is created, we can