Dependency Graphing

Dependency Graphing

We have a listener on the artifact resolver that can be used to track how the dependencies are resolved, which versions are used, and which are eliminated. Currently, only a simple textual representation is available at runtime. We want to be able to do some reporting based on this.

There will be several usage patterns for the resulting graphs.

- Identifying dependency exclusions.
  Is it excluded, and where did exclusion originate.
- Identifying dependency version decisions.
  If maven picks another version, which one, and why?
- Identifying dependency owner.
  If dependency is present in project, how did it get there?
  By current project, by transitive dependency, or by parent pom concepts?
- Identifying transitive dependency complexity.
  Multi module project is getting out of hand, what dependencies can i collapse safely into one dep?
- Identify relocated dependencies.
  If a dependency was relocated, identify it.
- Identify missing or bad dependencies.
  If dependency is not found, flag it.

We should allow the user to specify how the graph should be shown.

- With nodes representing a unique groupId:artifactId and the edges representing the scope/version ?
- With nodes representing a fully unique groupId:artifactId:version and the edges representing the scope?

What should be included?

- Just the active dependencies?
- The excluded dependencies, but flagged to indicate that they are excluded.
- Optional dependencies.
- All scopes? (indicate scope somehow. by color?)
- Plugins too.
- Build Extensions.
- Reports.

Current work

Eclipse plugins

Teh Eclipse plugin Q4E includes dependency graph and dependency analysis views.

Google SoC 2007

Piotr Tabor and Peter Kolbus will create this summer (as a Google Summer of Code participants - with Jason van Zyl and Carlos Sanchez as mentors respectively) a software system that will allow, during the build process of a project, to automatically generate diagrams of chosen aspects of the project.
http://docs.codehaus.org/display/MAVENUSER/Maven+Diagram+Maker

Grafo

Carlos has put together a initial version done with http://prefuse.sourceforge.net/

- http://people.apache.org/~carlos/grafo/graphview.html
- http://people.apache.org/~carlos/grafo/radialgraphview.html

It's under the maven sandbox at https://svn.apache.org/repos/asf/maven/sandbox/grafo

plexus-graph-visualization

Jason van Zyl is planning on overhauling the entire gathering and resolution of artifacts using the plexus-graph library. A visualization suite has already been created for it.

- http://svn.codehaus.org/plexus/plexus-sandbox/trunk/plexus-components/plexus-graph/
- http://svn.codehaus.org/plexus/plexus-sandbox/trunk/plexus-components/plexus-graph-visualization/

Other Resources

From Joakim:

- http://www.graphviz.org/ - Graphviz support exists in the plexus-graph-visualization
- plexus-graph-visualization - has providers for prefuse, graphviz, and touchgraph.
- Grouping / Clustering support is important for large projects.
- Transitive Reduction views of large complex trees are important to get rid of the noise.

Graphviz ( ) is an excellent package for creating graphs and graph images, but it is a native application. If this route is chosen, how do we handle graphviz? As a dependency, or as a configuration parameter to the pre-installed binary?

From Carlos:

- http://www.workingfrog.org/ a 3D view of java projects and dependencies
- http://prefuse.sourceforge.net/ the gallery shows amazing examples with interactive graphs in applets (BSD license)
- http://touchgraph.sourceforge.net/

From Jason:

- http://jung.sourceforge.net/ - Java Universal Network/Graph Framework

From Wim:

- http://www.xml.com/pub/a/2004/09/08/tree.html
- http://www.linguiste.org/syntax/tree/drawer/
- http://www.svgopen.org/2005/papers/ComparisonXML2SVGTransformationMechanisms/

They show how to convert an xml tree structure to a SVG image.

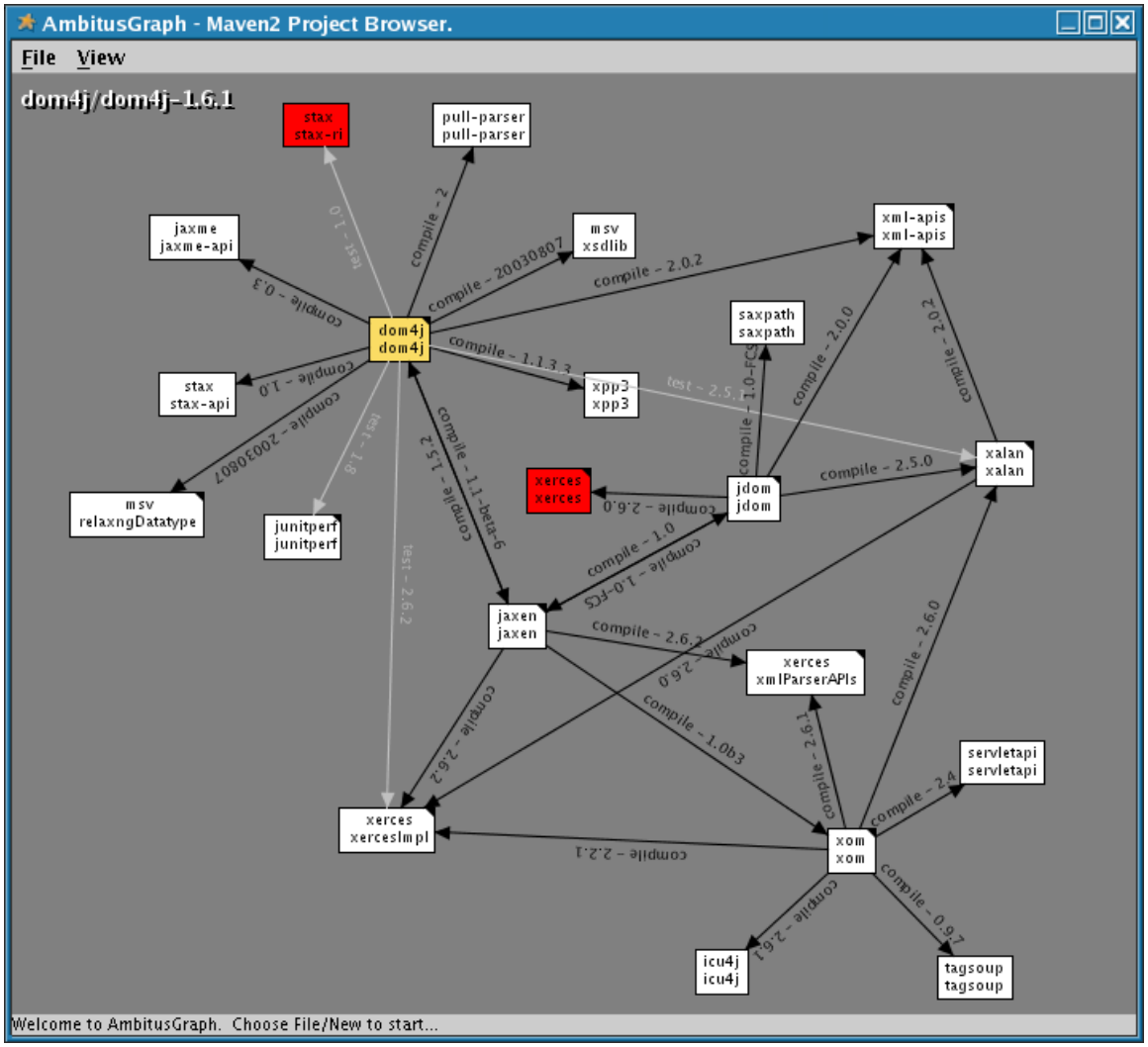
From Milos Kleint:

- I've written 2 graphs for the Netbeans integration. Both are based on the Netbeans graph library.

1. One shows the multiproject module structure. Details here. Currently just shows the included modules, but showing the non included ones and allowing interaction can be added in the future.
2. The second one shows the dependencies of a single project. Details and screenshot Scopes are

differenciated by color, links from main artifact are highlighted, haven't included version checking so far.

From Joakim:
Ambitusgraph: https://svn.sventech.com/repos/joakim/projects/trunk/ambitusgraph/

> *Currently an interactive dependency viewer experiment.*
> *Example of current functionality.*
>
> 
>