GraphTK 1.0 - User Manual

Julius E. Adorf

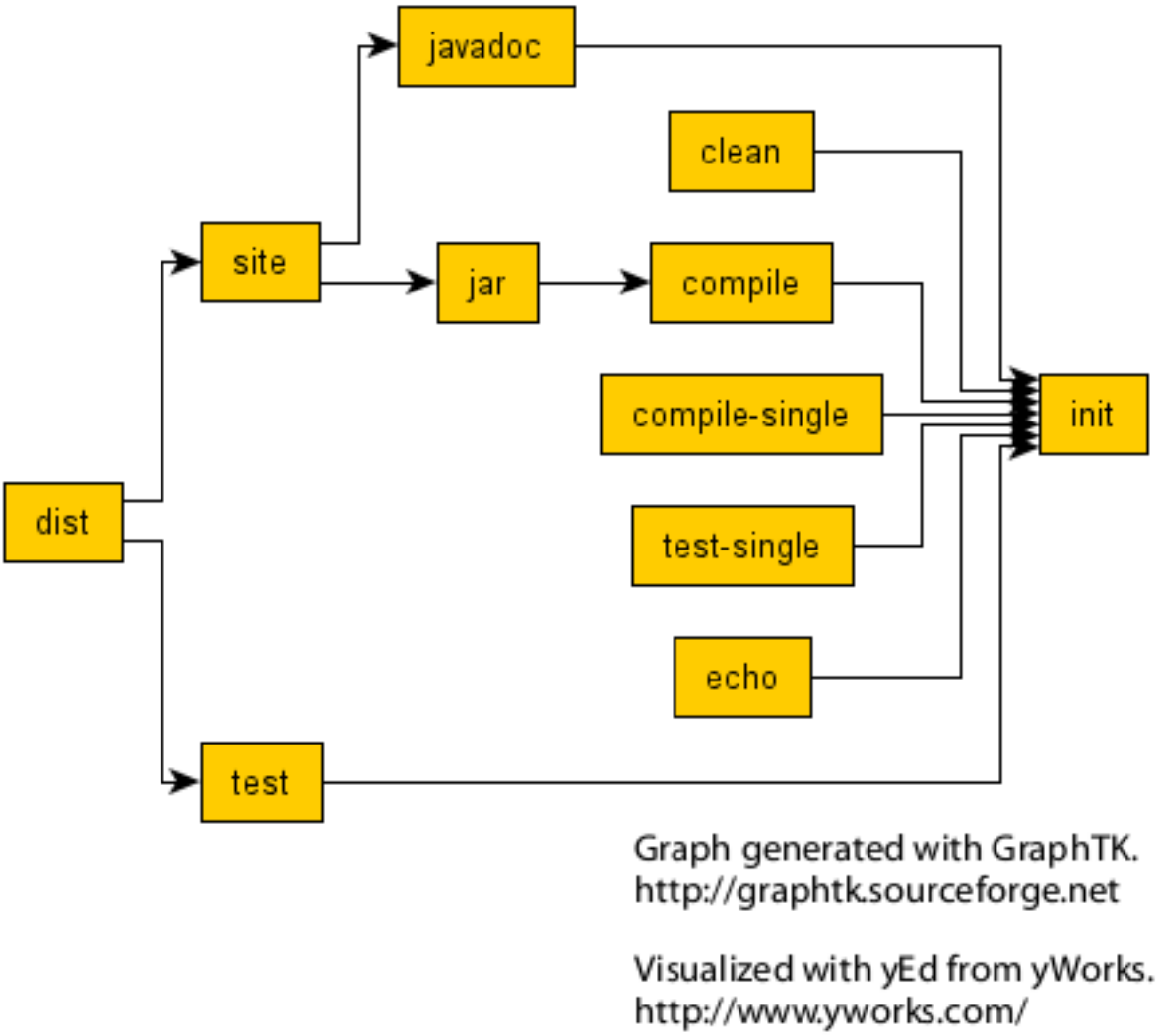

Table of Contents

About GraphTK

Introduction

GraphTK is a graph generation toolkit. Its main application areas are the analysis of build systems (e.g. Make or Ant), and the analysis of Java package dependencies.

The graphs may subsequently be visualized or processed with any external tool that understands one of the various graph formats supported by GraphTK.

GraphTK is an open source project hosted by SourceForge.net and published under the GNU General Public License. You may freely download and use this toolkit.

GraphTK may be used as a bridge between architectures that are based on graphs.

Target audience

The current version of GraphTK targets developers who are getting tired of doing dependency analyses that could be performed automatically.

Programming language

The toolkit is written in pure Java making it portable and platform-independent. Java developers may also use this toolkit as a library within their software packages.

About this manual

This manual describes the GraphTK software package. It is intended to be a reference for end users.

As GraphTK is not able to visualize the generated graphs, often an external application is necessary. There are many graph displaying tools; throughout this manual, we will use yEd.

Installation instructions

Requirements

GraphTK requires Java 5.0 to run. If you do not have installed Java JDK 5.0 yet, please download it from http://java.sun.com/j2se/1.5.0/download.jsp. If you are an end user, the Java Runtime Environment (JRE) is sufficient.

Steps

Simply extract the contents of the compressed ZIP file to the desired location, and you are ready to go.

Generating graphs

Build-system graphs

A build system is usually defined by one or several build files, each containing executable targets. Their corresponding build-system graphs are directed acyclic graphs (DAGs) describing the relationship between the various targets.

There are two basic graphs representing build systems: the workflow graph and the dependency graph. They only differ by the direction of the graph edges.

We use the following convention: A node represents a target, an edge denotes a dependency/workflow relation between two specific targets.

Workflow graphs

The workflow graph describes the order in which targets are called when a particular target is invoked. Usually, the graph is filtered, so that if A depends on B and C, and B depends on C, there is no edge connecting A with C.

Dependency graphs

The dependency graph describes the dependencies between the targets. Usually, the graph is filtered, so that if A depends on B and C, and B depends on C, there is no edge connecting A with C.

Java graphs

Package-package dependency graphs

Package-package dependency graphs model the dependencies between Java packages. They are directed graphs showing all packages your source packages depend on. As a package usually depends on many other packages, the resulting graphs can be very complex. In order to avoid having a graph with too many vertices and edges, you may limit the set of included packages.

Unlike the Ant and Make graphs, the generated package graph contains every edge. So, if package A depends on B and C, and B depends on C, there is an edge connecting A with C.

Importing packages

All packages contained in the source directories are regarded as importing packages. A vertex representing an importing package may have both incoming and outgoing edges.

Imported packages

All packages the source files depend on are called imported packages. A vertex representing an imported package never has an outgoing edge, only incoming edges.

Graph formats

GraphTK supports several formats for representing graphs: Ant, GML, XGML, GraphML and GraphML for yEd. If you need another format, please post a feature request. See section "Support" for details.

The format identifier is a name that is used throughout GraphTK to identify the graph format.

Note that not every format is equivally suitable for using it with other applications.

Ant

Description

An Ant project using the XML format. The document will not contain any tasks - it is only a stub with empty targets and dependencies.

A vertex becomes an Ant target, the outgoing edges represent its dependencies.

Identifier

Ant

Specification

http://ant.apache.org/

GraphML

Description

A XML graph format.

The graphs generated by GraphTK are not suitable for yEd Graph Editor.

Identifier

GraphML

Specification

http://graphml.graphdrawing.org/

GraphML for yEd

Description

A XML graph format.

This format is suitable for yEd Graph Editor and contains yEd-specific data.

Identifier

GraphML-yed

Specification

http://graphml.graphdrawing.org/

GML

Description

A graph format in plain text.

Identifier

GML

Specification

http://infosun.fmi.uni-passau.de/Graphlet/GML/

XGML

Description

XGML is a XML-like version similar to the GML format.

Identifier

XGML

Specification

http://www.yworks.com/products/yfiles/doc/developers-guide/xgml.html

Interfaces

GraphTK provides an interface for Ant, for the command line and comes with a graphical user interface.

Graphical user interface

GraphTK provides a simple GUI. You may either execute <graphtk>/lib/graphtk-1.0.jar or using the <graphtk>/bin/graphtk.bat / <graphtk>/bin/graphtk.sh to launch this little application.

Almost every component provides a tooltip text describing its function

Generating graphs

This section shows how to generate graphs using the graphical user interface.

Build system graph

- Choose "Generate Build System Graph"
- Select a build file
- Specify the type of the build file
- Change options (optional step)

Java package-package dependency graph

- Choose "Generate Package Graph"
- Add source folders

Graph viewers

A graph viewer is a graph-processing tool. Usually, it displays graphs. As GraphTK does not come with an internal viewer, it needs an external tool.

Registering a new graph viewer

Got to 'Preferences' and add a new graph viewer. Enter the necessary information.

The name can be chosen freely. There are two variables that may be used when constructing the argument: %path (path to a file that stores a graph) and %format (the preferred file format).

Command Line Interface

You can execute most of the functions using the command line interface. In the <graphtk>/bin folder you will find several batch/shell scripts. Use the flag -h for help.

Eventually - for easier access - you could add the <graphtk>/bin directory to your path. Additionally, in order to access the scripts from other directories, you must add an environment variable named GRAPHTK_HOME containing an absolute path to the <graphtk> folder.

Ant interface

You can execute most functions directly from Ant. This toolkit provides several Ant tasks which must be defined in your Ant file before using them.

Please first visit the section "Generate graphs", which provides a general overview about the graph generators.

Attributes that all graph tasks have in common:

- target: Denotes the output file the generated graph should be written to. If it is not specified, it is set to a default value. The different tasks have different default targets. The target attribute is not required, but if you do not specify it and the default file already exists, the task fails. The reason for this behaviour is to prevent you from accidentally overwriting an existing file.
- overwrite: Specifies whether to overwrite an existing target file in any case.
- format: The format attribute specifies the output format Please visit section "Formats" to find out which formats are supported. It is recommended that you specify this attribute.
- graphMode: Specifies whether to generate a workflow or a dependency graph (workflow | dependency). It is recommended that you specify this attribute.

Build system graph task

Attributes:

- system: Specifies the build system type. This attribute is required. Supported build systems are Ant and Make.
- file: Denotes the build system file. If this attribute is not specified, the task looks for a file with a common name depending on the type of the build system. For Ant, the default file name is "build.xml", for Make this is "Makefile".
- filter: Specifies whether to filter the graph, removing all unnecessary edges. An edge [A, C] is superfluous if A is connected indirectly with C.

Note: for Ant, the default file name is "build.<format-extension>", for Make this is "Makefile.<format-extension>".

```
<taskdef name="build-graph"
         classpath="graphtk-1.0.jar"
         classname="org.municware.graphtk.buildsys.BuildSystemGraphTask" />

<build-graph system="Ant"
             file="build.xml"
```

```
                target="build/build.gml"
                graphMode="workflow"
                format="GML"
                filter="true" />

<build-graph system="Make"
                file="Makefile"
                target="build/Makefile.graphml"
                graphMode="dependency"
                format="GraphML"
                filter="true" />
```

Java package graph task

Use nested path elements to specify the source directories.

```
<taskdef name="package-graph"
         classpath="graphtk-1.0.jar"
         classname="org.municware.graphtk.java.PackageGraphTask" />

<package-graph target="build/package-deps.graphml"
               format="GraphML">
    <path location="one/src">
    <path location="two/src">
</package-graph>
```

Java interface

As GraphTK is written in Java, you may access its functions directly from within your Java code. It is designed for easy usage as simple things should be simple.

For further information, please read the developer manual.

External libraries

Ant

Apache Ant is a Java-based build tool.

http://jakarta.apache.org/commons/cli/.

CLI

CLI is a useful library when writing a command-line interface.

http://jakarta.apache.org/commons/cli/.

JGraphT

JGraphT is used for the modelling of the graphs.

http://jgrapht.sourceforge.net/.

Swing Layout

Swing Layout is a useful library when writing a graphical user interface.

https://swing-layout.dev.java.net/.

Known limitations

Makefile parsing

The included makefile parser does not support all features of a makefile.
Make parser

- It does not understand macros.

There may also be some other limitations, which are not known yet.

Java package graph

The implementation of the source code scanner is quite straight-forward but not correct. The following conditions must be fulfilled to guarantee the generator works properly:

- import statements must be on separate lines
- import statements must not have comments between keyword, statement and finishing semi-colon
- package names must start with a lower case character
- class names must start with a capital letter

Static imports (Java 1.5.0) are ignored. Future versions will hopefully fix those issues.

Support

If you are running into problems, please have a look at the FAQ list on our web-site. If this does not solve your problem, please post a support request at our SF website, or send an e-mail to graphtk(AT)municware.de.

If you find a bug or like to have a specific feature implemented, feel encouraged to contact us and help us to improve this toolkit.

Links

Project home page: http://graphtk.sourceforge.net/

Project SF home page: http://www.sourceforge.net/projects/graphtk

Bug reporting: http://sourceforge.net/tracker/?group_id=161303&atid=819295

Support requests: http://sourceforge.net/tracker/?group_id=161303&atid=819296

Feature requests: http://sourceforge.net/tracker/?group_id=161303&atid=819298