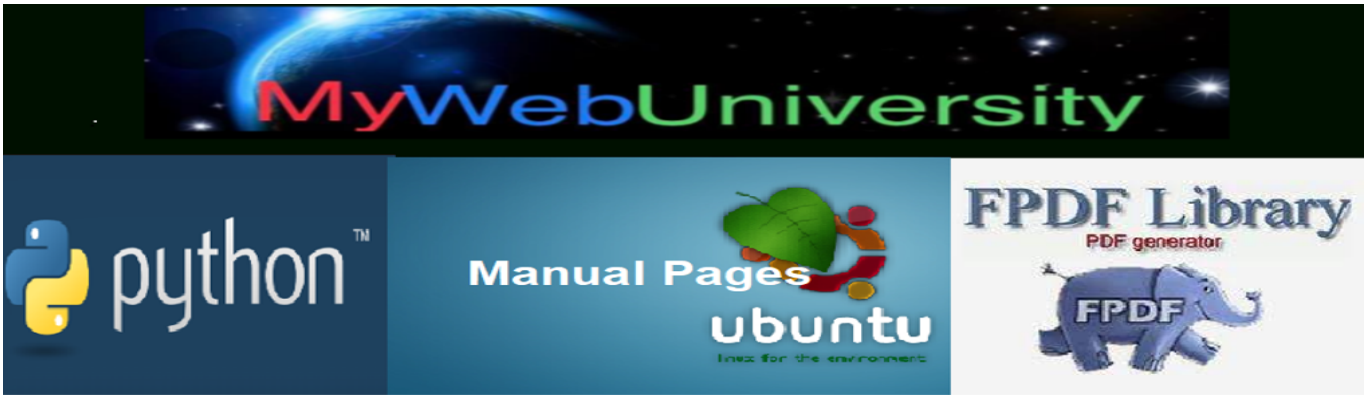

Rocky Enterprise Linux 9.2 Manual Pages on command 'jdeps.1'

$ man jdeps.1

JDEPS(1) JDK Commands JDEPS(1)

NAME

jdeps - launch the Java class dependency analyzer

SYNOPSIS

jdeps [options] path ...

options

Command-line options. For detailed descriptions of the options that can be used,

see

? Possible Options

? Module Dependence Analysis Options

? Options to Filter Dependences

? Options to Filter Classes to be Analyzed

path A pathname to the .class file, directory, or JAR file to analyze.

DESCRIPTION

The jdeps command shows the package-level or class-level dependencies of Java class files.

The input class can be a path name to a .class file, a directory, a JAR file, or it can be

a fully qualified class name to analyze all class files. The options determine the out?

put. By default, the jdeps command writes the dependencies to the system output. The

command can generate the dependencies in DOT language (see the -dotoutput option).

POSSIBLE OPTIONS

-? or -h or --help

Prints the help message.

-dotoutput dir or --dot-output dir

Specifies the destination directory for DOT file output. If this option is speci? fied, then the jdepscommand generates one .dot file for each analyzed archive named archive-file-name.dot that lists the dependencies, and also a summary file named summary.dot that lists the dependencies among the archive files.

-s or -summary

Prints a dependency summary only.

-v or -verbose

Prints all class-level dependencies. This is equivalent to

-verbose:class -filter:none

-verbose:package

Prints package-level dependencies excluding, by default, dependences within the same package.

-verbose:class

Prints class-level dependencies excluding, by default, dependencies within the same archive.

-apionly or --api-only

Restricts the analysis to APIs, for example, dependences from the signature of pub? lic and protected members of public classes including field type, method parameter types, returned type, and checked exception types.

-jdkinternals or --jdk-internals

Finds class-level dependences in the JDK internal APIs. By default, this option analyzes all classes specified in the --classpath option and input files unless you specified the -include option. You can't use this option with the -p, -e, and -s options.

Warning: The JDK internal APIs are inaccessible.

-cp path, -classpath path, or --class-path path

Specifies where to find class files.

--module-path module-path

Specifies the module path.

--upgrade-module-path module-path

Specifies the upgrade module path.

--system java-home

Specifies an alternate system module path.

--add-modules module-name[, module-name...]

Adds modules to the root set for analysis.

--multi-release version

Specifies the version when processing multi-release JAR files. version should be an integer >=9 or base.

-q or -quiet

Doesn't show missing dependencies from -generate-module-info output.

-version or --version

Prints version information.

MODULE DEPENDENCE ANALYSIS OPTIONS

-m module-name or --module module-name

Specifies the root module for analysis.

--generate-module-info dir

Generates module-info.java under the specified directory. The specified JAR files will be analyzed. This option cannot be used with --dot-output or --class-path op? tions. Use the --generate-open-module option for open modules.

--generate-open-module dir

Generates module-info.java for the specified JAR files under the specified directo? ry as open modules. This option cannot be used with the --dot-output or --class-path options.

--check module-name [, module-name...]

Analyzes the dependence of the specified modules. It prints the module descriptor, the resulting module dependences after analysis and the graph after transition re? duction. It also identifies any unused qualified exports.

--list-deps

Lists the module dependences and also the package names of JDK internal APIs (if referenced). This option transitively analyzes libraries on class path and module path if referenced. Use --no-recursive option for non-transitive dependency analy? sis.

--list-reduced-deps

Same as --list-deps without listing the implied reads edges from the module graph. If module M1 reads M2, and M2 requires transitive on M3, then M1 reading M3 is im? plied and is not shown in the graph.

--print-module-deps

Same as --list-reduced-deps with printing a comma-separated list of module depen? dences. The output can be used by jlink --add-modules to create a custom image that contains those modules and their transitive dependences.

--ignore-missing-deps

Ignore missing dependences.

OPTIONS TO FILTER DEPENDENCES

-p pkg_name, -package pkg_name, or --package pkg_name

Finds dependences matching the specified package name. You can specify this option multiple times for different packages. The -p and -e options are mutually exclu? sive.

-e regex, -regex regex, or --regex regex

Finds dependences matching the specified pattern. The -p and -e options are mutu? ally exclusive.

--require module-name

Finds dependences matching the given module name (may be given multiple times). The --package, --regex, and --require options are mutually exclusive.

-f regex or -filter regex

Filters dependences matching the given pattern. If give multiple times, the last one will be selected.

-filter:package

Filters dependences within the same package. This is the default.

-filter:archive

Filters dependences within the same archive.

-filter:module

Filters dependences within the same module.

-filter:none

No -filter:package and -filter:archive filtering. Filtering specified via the -filter option still applies.

--missing-deps

Finds missing dependences. This option cannot be used with -p, -e and -s options.

OPTIONS TO FILTER CLASSES TO BE ANALYZED

-include regex

Restricts analysis to the classes matching pattern. This option filters the list of classes to be analyzed. It can be used together with -p and -e, which apply the pattern to the dependencies.

-P or -profile

Shows the profile containing a package. This option is deprecated and may be re? moved in a future release.

-R or --recursive

Recursively traverses all run-time dependences. The -R option implies -fil? ter:none. If -p, -e, or -f options are specified, only the matching dependences are analyzed.

--no-recursive

Do not recursively traverse dependences.

-I or --inverse

Analyzes the dependences per other given options and then finds all artifacts that directly and indirectly depend on the matching nodes. This is equivalent to the inverse of the compile-time view analysis and the print dependency summary. This option must be used with the --require, --package, or --regex options.

--compile-time

Analyzes the compile-time view of transitive dependencies, such as the compile-time view of the -R option. Analyzes the dependences per other specified options. If a dependency is found from a directory, a JAR file or a module, all classes in that containing archive are analyzed.

EXAMPLE OF ANALYZING DEPENDENCIES

The following example demonstrates analyzing the dependencies of the Notepad.jar file.

Linux and macOS:

```
$ jdeps demo/jfc/Notepad/Notepad.jar
Notepad.jar -> java.base
Notepad.jar -> java.desktop
Notepad.jar -> java.logging
  <unnamed> (Notepad.jar)
    -> java.awt
    -> java.awt.event
    -> java.beans
```

```
   -> java.io
   -> java.lang
   -> java.net
   -> java.util
   -> java.util.logging
   -> javax.swing
   -> javax.swing.border
   -> javax.swing.event
   -> javax.swing.text
   -> javax.swing.tree
   -> javax.swing.undo
```

Windows:

```
C:\Java\jdk1.9.0>jdeps demo\jfc\Notepad\Notepad.jar
Notepad.jar -> java.base
Notepad.jar -> java.desktop
Notepad.jar -> java.logging
  <unnamed> (Notepad.jar)
   -> java.awt
   -> java.awt.event
   -> java.beans
   -> java.io
   -> java.lang
   -> java.net
   -> java.util
   -> java.util.logging
   -> javax.swing
   -> javax.swing.border
   -> javax.swing.event
   -> javax.swing.text
   -> javax.swing.tree
   -> javax.swing.undo
```

EXAMPLE USING THE --INVERSE OPTION

```
$ jdeps --inverse --require java.xml.bind
```

Inverse transitive dependences on [java.xml.bind]

java.xml.bind <- java.se.ee

java.xml.bind <- jdk.xml.ws

java.xml.bind <- java.xml.ws <- java.se.ee

java.xml.bind <- java.xml.ws <- jdk.xml.ws

java.xml.bind <- jdk.xml.bind <- jdk.xml.ws