LatexDoclet API Documentation

Robert McDermid

Mar 3, 1999

Contents

0.1 LatexDoclet

java.lang.Object

---

public *LatexDoclet*
extends Object

This class is a doclet that generates Javadoc help as a LaTex file. LaTex is a macro package for Donald Knuth's excellent Tex type-setting package. It is available on virtually every platform. This doclet generates code that conforms to the LaTex 2e distribution. It requires one non-standard package, the fancyhdr package.

Javadoc documentation is frequently enhanced by adding html tagging to the comments. Obviously, this will not do anything useful in a LaTex file, so this package attempts to make a reasonable translation for most of the tags commonly found in Sun's documentation. Fortunately, it is fairly easy to map these tags into LaTex, but it is not always perfect.

This doclet adds the following command line options to javadoc:

-f	Specify the output filename. Defaults to javadoc.tex
-title "title"	Specify the title for the documentation File. This will be displayed on the title page
-docauthor	Specify the name of the document author. This will be displayed on the title page.
-twoside	Specifies that the document will eventually be printed double-sided. Tbe output will be formatted appropriately for this.
-nodetails	The detailed constructor, method, and field information will not be shown. Only the summaries will be shown. This results in a good reference type document.
-nosummary	The summaries will be omitted, but method, constructor, and field details will be shown. This results in a slightly shorter document.

0.1.1 Field Summary

Type	Description
private static int	**allClassesRef** Holds the id of the panel listing all classes in the generated file
private static String	**author** The document author - used on the title page.
private static Hashtable	**classIds** This hashtable is used to match heading ids to fully qualified class names.

Type	Description
private static int	**currentPackageId** Holds the link id of the package currently being processed
private static boolean	**definitionListFirstLine** This flag is used when converting definition lists from html to latex.
private static RootDoc	**doc** This is the RootDoc object passed to the start method.
private static String[]	**ourPackages** An array of all the package names documented in the IPF being generated.
private static PrintWriter	**out** The output is written to this Writer.
private static String	**outputFileName** The name of the output file.
private static int	**packageListId** Holds the id of the panel with the list of all packages
private static int	**ref** This variable is incremented every time a heading id is specified, so that each eading id will be unique.
private static boolean	**showDetails** Flag indicates whether to show the details of constructors, methods, and fields.
private static boolean	**showSummary** Falg indicates whether to show the summaries of constructors, methods, and fields.
private static String	**title** The title to use for the latex file.
private static boolean	**twoSide** Flag indicating whether document should be formatted for double-sided printing or not.

0.1.2 Constructor Summary

Description
LatexDoclet()

0.1.3 Method Summary

Returns	Description
public static String	**dehyphenate(java.lang.String str)** Prevents latex from hyphenating the words, by appending a mandatory hyphen point at the end of all the words in the string.
public static int	**getClassId(com.sun.javadoc.ClassDoc c)** Gets the id for a given class.
public static int	**getClassId(java.lang.String className)** Gets the id for a given class.
public static String	**getExceptionsString(com.sun.javadoc.ExecutableMemberDoc method)** Returns a string with a comma-separate list of all the exceptions that may be thrown by a method.
public static String	**getLink(java.lang.String className, java.lang.String text)** Gets Latex for reference to specified class.
public static String	**getParamString(com.sun.javadoc.ExecutableMemberDoc method)** Returns a string containing all the parameters for a method.
public static void	**header()** Writes the latex header.
public static boolean	**isPackageInternal(java.lang.String pack)** Determines if the specified package is one being processed in this file.
public static int	**optionLength(java.lang.String option)** Returns how many option words specify a given tag.
public static void	**printClassTree(com.sun.javadoc.ClassDoc doc)** Outputs latex for superclass hierarchy for a class.
public static void	**processClass(com.sun.javadoc.ClassDoc c)** Produces the latex for a particular class.
public static String	**processCommentText(java.lang.String in, boolean inSummary)** Process the comment or tag text to convert html tagging to Latex commands.
public static void	**processConstructors(com.sun.javadoc.ConstructorDoc[] docs, int startRes)** Write out a the constructor details.
public static void	**processFields(com.sun.javadoc.FieldDoc[] docs, int startRes)** Write out a field details.

Returns	Description
public static void	**processMethods(com.sun.javadoc.MethodDoc[] docs, int startRes)** Write out the method details.
public static void	**processPackage(com.sun.javadoc.PackageDoc p, int id)** Processes an individual package.
public static String	**processSymbols(java.lang.String in)** This method is called to convert certain characters in a string to their symbolic representation in latex.
public static void	**readOptions(java.lang.String[][] options)** Processes our extra command line options.
public static boolean	**start(com.sun.javadoc.RootDoc _doc)** Called to start processing documentation.
public static void	**trailer()** Writes the trailer for the latex file.

0.1.4 Fields

allClassesRef

private static int ***allClassesRef***

Holds the id of the panel listing all classes in the generated file

author

private static String ***author***

The document author - used on the title page. Defaults to an empty string

classIds

private static Hashtable ***classIds***

This hashtable is used to match heading ids to fully qualified class names. We use these to look p the ids for @see tags and other links internal to the generated ipf file.

currentPackageId

private static int ***currentPackageId***

Holds the link id of the package currently being processed

definitionListFirstLine

private static boolean ***definitionListFirstLine***

This flag is used when converting definition lists from html to latex. On the first line, e do not start the line with \\. On other lines we do.

doc

private static RootDoc ***doc***

This is the RootDoc object passed to the start method. We keep a static reference to it ecause it has the DocErrorReporter in it that we may need to use.

ourPackages

private static String[] ***ourPackages***

An array of all the package names documented in the IPF being generated.

out

private static PrintWriter ***out***

The output is written to this Writer.

outputFileName

private static String ***outputFileName***

The name of the output file. This defaults to javadoc.tex, but can be overridden y the -f command line option.

packageListId

private static int ***packageListId***

Holds the id of the panel with the list of all packages

ref

private static int ***ref***

This variable is incremented every time a heading id is specified, so that each eading id will be unique.

showDetails

private static boolean ***showDetails***

Flag indicates whether to show the details of constructors, methods, and fields.

showSummary

private static boolean ***showSummary***

Falg indicates whether to show the summaries of constructors, methods, and fields.

title

private static String ***title***

The title to use for the latex file. Defaults to "Javadoc Documentation"

twoSide

private static boolean ***twoSide***

Flag indicating whether document should be formatted for double-sided printing or not.

0.1.5 Constructors

LatexDoclet

public ***LatexDoclet****()*

0.1.6 Methods

dehyphenate

public static String ***dehyphenate****(java.lang.String str)*

Prevents latex from hyphenating the words, by appending a mandatory hyphen point at the end of all the words in the string. This process removes all extra spaces.

Parameters:

str	The string to de-hyphenate

Returns:
The string with all words having \- appended

getClassId

public static int ***getClassId****(com.sun.javadoc.ClassDoc c)*

Gets the id for a given class. Only works on classes that are going to be in the file we are generating. It finds it by looking it up in the hashtable classIds.

Parameters:

c	The class to get the id for.

Returns:
The id of the specified class.

getClassId

public static int ***getClassId****(java.lang.String className)*

Gets the id for a given class. Only works on classes that are going to be in the file we are generating. It finds it by looking it up in the hashtable classIds.

Parameters:

className	The fully qualified class name to get the id for.

Returns:
The id of the specified class.

getExceptionsString

public static String
getExceptionsString*(com.sun.javadoc.ExecutableMemberDoc method)*

Returns a string with a comma-separate list of all the exceptions that may be thrown by a method.

Parameters:

method	The method to get the exception list for.

Returns:
The comma-separated list of exceptions.

getLink

public static String ***getLink****(java.lang.String className, java.lang.String text)*

Gets Latex for reference to specified class. This produces a string containing a complete Latex reference string, if the class is one of the ones we are processing. Otherwise, it just returns the name of the class.

Parameters:

className	Fully qualified class name of the class to get the reference for
text	

Returns:
String containing the reference text.

getParamString

public static String ***getParamString****(com.sun.javadoc.ExecutableMemberDoc method)*

Returns a string containing all the parameters for a method. The parameters are separated by commas and surrounded by brackets.

Parameters:

method	The method to get the parameter list for

Returns:
String with the parameter list

header

public static void ***header****()*

Writes the latex header. In here we set our document class, load some extra packages, set up the page headers, create the title page and table of contents, and so forth.

isPackageInternal

public static boolean ***isPackageInternal****(java.lang.String pack)*

Determines if the specified package is one being processed in this file. If the package is not internal then references to it cannot be done.

Parameters:

pack	The package name to check for

Returns:
true if the package is being documented in the latex file being generated.

optionLength

public static int ***optionLength****(java.lang.String option)*

Returns how many option words specify a given tag. This is part of the command-line option processing for javadoc. Any unregonized tags are passed in here, and it returns how many symbols that tag should be. For example, for -f which takes a filename as an argument, it would return 2 (1 for the tag itself, and one for the filename).

Parameters:

option	The option to get length for

Returns:
The length for the option.

printClassTree

public static void ***printClassTree****(com.sun.javadoc.ClassDoc doc)*

Outputs latex for superclass hierarchy for a class.

Parameters:

doc	The class to print the chart for

processClass

public static void ***processClass****(com.sun.javadoc.ClassDoc c)*

Produces the latex for a particular class. This consists of writing out the class section with the description of the class, and field, constructor and method summaries, and then writing out the text for each individual field, method, and constructor.

Parameters:

c	The class to document

processCommentText

public static String ***processCommentText****(java.lang.String in, boolean inSummary)*

Process the comment or tag text to convert html tagging to Latex commands. Sun, in particular, uses a lot of html tags in its comments to format them nicely when converted to javadoc format. Fortunately, many of these can easily be easily converted to Latex. Tags which are not recognized are omitted. This is not perfect, but usually what comes out is reasonably decent looking. processSymbols is also called by this method to fix up any tricky symbols.

Parameters:

in	The text to process
inSummary	If the text to be processed is from the summary, set this to true. This prevents things like lists from being translated, which sometimes causes problems for latex, because the end tag will not be included because of truncation for the summary.

Returns:

Cleaned up text.

processConstructors

public static void ***processConstructors****(com.sun.javadoc.ConstructorDoc[] docs, int startRes)*

Write out a the constructor details. This method is called by processClass to write out documentation for each constructor for the class.

Parameters:

docs	Array of constructor information
startRes	The reference id to use for the first constructor. Add one for second constructor, and so forth.

processFields

public static void ***processFields****(com.sun.javadoc.FieldDoc[] docs, int startRes)*

Write out a field details.

Parameters:

docs	Array of field information
startRes	The reference id to use for the first field. Add one for second field, and so forth.

processMethods

public static void ***processMethods****(com.sun.javadoc.MethodDoc[] docs, int startRes)*

Write out the method details.

Parameters:

docs	Array of nethod information
startRes	The reference id to use for the first method. Add one for second method, and so forth.

processPackage

public static void ***processPackage****(com.sun.javadoc.PackageDoc p, int id)*

Processes an individual package. This means we start a new chapter and list all the classes in the package, with references. Then we detail each individual class in separate sections.

Parameters:

p	The package to document
id	The id to use for the label. These have already been predetermined in the calling routine.

processSymbols

public static String ***processSymbols****(java.lang.String in)*

This method is called to convert certain characters in a string to their symbolic representation in latex. For example, you cannot leave a backslash in clear test, because it will be interpreted as a latex command. So, it is replaced with it's symbolic representation. There are quite a few special characters in Latex (good Unix-style design- use as many cryptic symbols as possible in the syntax!) but I think this handles them all.

Parameters:

in	The string to process

Returns:

Processed string.

readOptions

public static void ***readOptions****(java.lang.String[][] options)*

Processes our extra command line options. There are several extra command line options supported by this doclet, and this scans through the options presented, and handles them appropriately. No error checking is performed at

this time.

Parameters:

options	Array of options

start

public static boolean ***start****(com.sun.javadoc.RootDoc _doc)*

Called to start processing documentation. This is the entry point for the doclet, and it handles outputting all the preliminary pages such as the package list, and sets up the class ids. It then processes each package one by one by calling processPackage.

Parameters:

_doc	The document object to process

Returns:

true on success.

trailer

public static void ***trailer****()*

Writes the trailer for the latex file. Basically just prints the index and the \end{document} tag.

Index