



**Drop-in Hypertext Help  
for Java™ Applets and Applications**

**Version 2.0 for Java 1.1**

jHelp Software and Documentation  
Copyright © 1997 by William Giel / Moondog Software

## Introduction

jHelp is a Java™ package that makes it easy to add on-line help documentation to applets and applications.

jHelp provides extensive support of HTML 2, letting technical writers create attractive and functional documentation without incurring excessive help system overhead and code size.

jHelp implements a superset of HTML 2, with new tags that extend the specification's ability to control font typefaces, colors and sizes (similar to HTML 3.) For authors creating help documentation with HTML 3.2 editors, jHelp supports the alignment attribute in division, paragraph and header tags. The package also defines special index and keyword tags to permit automatic generation of a navigable index list using a helper application provided with the package.

jHelp is true to the Java paradigm. It is coded entirely in Java, without any native code dependencies. It runs on any platform that supports Java. It supports languages other than English, including Japanese, Chinese and Korean (tested on NT.) It frees your products from dependency on the presence of a specific brand of browser to display help files.

jHelp is licensed royalty-free, permitting unlimited distribution of the compiled package as a help or document-viewing subsystem in your program. Source code is available.

jHelp Technical Support has been praised for its quality and prompt attention to licensees. Moondog Software is committed to continuing to improve jHelp, expanding its capabilities while maintaining compatibility with previous versions, and striving to make jHelp the standard Java help system for significant Java products.

William Giel  
d.b.a. Moondog Software  
Milford, CT 06460

<http://w3.nai.net/~rvdi/bgiel/bill.htm>  
[bgiel@ct2.nai.net](mailto:bgiel@ct2.nai.net)

---

Java and other Java-based names are trademarks of Sun Microsystems, Inc. and refer to Sun's family of Java-branded technologies.

## Table of Contents

<b>Installation</b>	<b>1</b>
Unpacking	1
<i>Zip Files</i>	1
Placing jHelp in the CLASSPATH	1
Deploying on the WWW	2
JAR Files	2
<b>Overview of jHelp's Public Classes</b>	<b>3</b>
htmlHelpFrame	3
htmlHelpPanel	3
htmlHelpIndexFrame	3
htmlHelpIndexBuilder	4
htmlResources and htmlExcludeWords	4
<b>Applets vs. Applications</b>	<b>5</b>
Significant Differences	5
URLS	5
Local Files	6
<b>Using Automated Indexing</b>	<b>7</b>
Using htmlHelpIndexBuilder	8
Usage Example	8
INDEX Tags	9
KEYWORD Tags	10
Level Delimiters	10
Headers	11
Document Titles	11
Backward Compatibility	11
Indexing Tags Example	11
<b>Using the Index Wizard</b>	<b>13</b>
<b>Managing Help Document Directories</b>	<b>18</b>
jHelp's Constructors	18
<i>Using Zips and Jars</i>	19
Path Separators	19
<i>Path separators in Zips and Jars</i>	19
Setting BASE Directories	19
<i>Base directories with Zips and Jars</i>	20
Using htmlLocator Methods with an Application	21

<b>Odds 'n' Ends</b>	<b>22</b>
Extending htmlHelpPanel	22
Document Caching	23
Dynamic HTML	23
<i>Setting Dynamic Content</i>	23
<i>Displaying Dynamic Content</i>	24
Minimizing Distribution Size	25
<b>Ordering, Upgrades, &amp; Technical Support</b>	<b>26</b>
Upgrades	26
Technical Support	26
Ordering and Licensing jHelp	26
<b>Appendix A - HTML DETAILS</b>	<b>27</b>
Unsupported	27
Supported	27
Coded Characters	28
<b>Appendix B - Standard License</b>	<b>29</b>
<b>Appendix C - Current Licensing Options</b>	<b>31</b>

## Installation

The jHelp package's distribution is provided as a compressed ZIP file.

Included with the actual jHelp package classes are sample applets and applications that demonstrate deployment of jHelp, sample help documents, javadoc documentation, and supplemental HTML documentation with which you can quickly get started.

Once you have extracted the zip to your system, you should browse the `readme.html` file in the zip's root directory for information to quickly get you started with the demos.

## Unpacking

Java requires support of case-sensitive, long filenames. Thus any unpacking utility should support both of these. The venerable PKZip 2.04 DOS utility *does not*. Windows users should obtain either WinZip95 or PKZip 2.5 for Windows 95/NT. On other platforms, be sure to use a utility that provides equivalent functionality. The JAR tool provided with the JDK should be able to extract the files in the zip.

## Zip Files

When you open jHelp's distribution zip file with WinZip or PKZip, you should see that the files are arranged in several subdirectories within the zip file. You can simply unpack the files to a directory or folder of your choice, however be sure to check or enable the option to use the directory names contained in the zip. This will create the proper directory structure within your destination directory for running the example applets and applications.

If you use the JAR tool, place the zip in a new directory and extract with JAR. This should create the proper directory structure to run the demos. You can delete the distribution zip once you've unpacked it.

## Placing jHelp in the CLASSPATH

jHelp's package is contained in the jar file, `jhelp.jar` in the distribution zip's `lib` directory. This jar contains the entire jHelp package, along with an Index Wizard package and supporting `moondog` package. When you distribute jHelp with your application, you can safely remove the `moondog` and `jhelp.index` packages, as well as `htmlHelpIndexBuilder.class` to reduce the size of the package.

When you ultimately deploy jHelp with your application, there are essentially two ways in which jHelp can be accessed. One way is to create a jHelp directory below your application's class directory, into which the jHelp package classes should be installed (these are the class files that all begin with "html...".) Another way is to place the jHelp package directory in the classpath.

For example, a system may already have a directory into which third-party packages are installed. Let's say such a directory exists and is named `c:\packages`. Copy the `jhelp.jar` file from the distribution's `lib` directory into your `packages` directory and add it to your classpath with (in Windows 95)

```
set CLASSPATH=.;c:\java\lib\classes.zip;c:\packages\jhelp.jar
```

## **Deploying on the WWW**

Under the current security model deployed by typical browsers, web applets cannot access local packages named in the classpath. Thus if you are using jHelp with a web applet delivered over the Internet, the jHelp package classes should be located in a subdirectory named `jHelp`, immediately below the applet's code base.

## **JAR Files**

Of course, you may choose to provide your applet in a JAR file, in which case the jHelp package directory is simply included within the JAR. The directory structure within the JAR should mirror the directory structure that would be used if the applet classes were not provided as a JAR file. The JAR file is specified using the `ARCHIVE` attribute of the HTML applet tag.

Beginning with jHelp 2.07, you may also package your help documents and images in the same jar as your applet.

## Overview of jHelp's Public Classes

The following section is provided only to describe jHelp's public classes in general terms. For detailed technical documentation, please refer to the javadoc documentation and the sample applets and application.

### htmlHelpFrame

This class creates a pop-up help frame, with a scrolling panel for displaying a help document, forward/back navigation buttons, a search button that permits searching the current document for a word or phrase, a print button that will send the current document to a printer, an index button that will display a pop-up index (if you've generated one, see `htmlHelpIndexFrame`, below,) and a close button that hides the help panel.

The title of the pop-up frame is taken from the title of the help document, as set using HTML's TITLE tags.

By default, printing in applets is disabled although you may enable it with a provided method (refer to javadoc documentation.)

The index button is initially disabled unless an `htmlHelpIndexFrame` object is passed using the method provided. This will then enable indexing (refer to javadoc documentation.)

### htmlHelpPanel

This component is a basic help document scroller that might be described as a jHelp "widget."

It provides a way for you to place a jHelp panel within your own container. If you choose to deploy jHelp in this fashion it will be your responsibility to provide controls that implement jHelp's forward/back navigation methods, indexing, printing, etc.

### htmlHelpIndexFrame

This class provides for a pop-up, alphabetically-sorted, nested index of keywords, terms or topics contained within all the help documents associated with your application. Optionally, if you provide the name of a word list file (generated with the Index Builder) when you construct the index frame, then the index frame will appear as a tabbed-folders style window, with an index, word search, and query panels.

An index is built using jHelp's `htmlHelpIndexBuilder` application, or the Index Builder Wizard, both of which are described below.

## htmlHelpIndexBuilder

This class is a standalone console application that generates an index file for use with an htmlHelpIndexFrame object. It can be run directly from the command line, or from a project-oriented **Wizard** front-end GUI.

These are described in detail in following sections of this documentation.

## htmlResources and htmlExcludeWords

These are extended classes of java.util.ListResourceBundle, provided to facilitate foreign language translation of hard-coded captions, labels, warnings, trivial words and other messages that might be displayed to a user.

htmlResources and htmlExcludeWords provide U.S. English language values for literal strings used throughout the jHelp package. By extending htmlResources, and htmlExcludeWords, developers can provide translated values that will be automatically selected, based on the users current locale.

Source code for all ListResourceBundle subclasses is provided with jHelp, regardless of whether or not a source license is purchased.

For detailed information, please refer to Java's API documentation.



## Applets vs. Applications

Back when the first version of jHelp was created, it was originally intended to be used with applets (and it was called "Applet Help Browser.") As developers began producing significant Java applications, it quickly became apparent that there was a need for an application-compatible help system and Applet Help Browser was revised to provide application support.

Since then, Applet Help Browser was completely redesigned and rewritten from scratch (and renamed to jHelp.) With enhanced functionality, the package is about three times the size of the early version (currently about 120 KB versus 40 KB) which somewhat diminishes its suitability for applet implementations.

Notwithstanding the above, jHelp still supports use with applets. However, there are some differences and limitations associated with applet implementations of jHelp.

### Significant Differences

The principle differences involve the methods that jHelp uses to retrieve help documents and images. Applets typically cannot access local files, nor is it reasonable to expect that an applet user will maintain help documents locally for an occasionally-visited web applet.

When you construct a jHelp object, you must pass the instance of your program in the constructor, either an Applet instance for applets, or a valid Frame instance for applications. jHelp will select appropriate methods for retrieving files based on this instance (using the instanceof test.)

## URLS

In either applets or applications, you can specify a file as a fully qualified URL, e.g. `http://www.net.com/helpdocs/doc.html`. jHelp will recognize this as a URL, and attempt to retrieve it using URL methods. However, keep in mind that applet socket connections are generally restricted to their server of origin, and an applet probably will not be able to access a remote server. Applications are typically not subject to this restriction.

Even if you do not fully qualify a file as a URL, applet implementations of jHelp will use URL methods to retrieve files. Basically, jHelp will build a URL for the file, using the applet's code base. Thus, if you have downloaded the applet from

`http://www.net.com/applet/`

and jHelp needs a file "docs/help.html," it will look for this file as

`http://www.net.com/applet/docs/help.html`

## **Local Files**

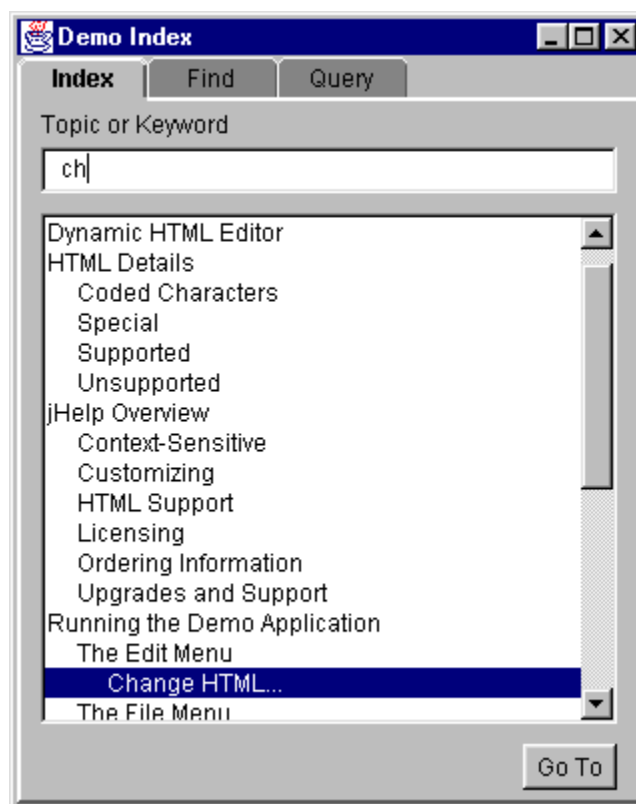
Applets usually are prohibited from accessing local files on the user's system for well known security reasons. Because of this, jHelp uses URL methods to retrieve help documents from an applet by default, even if the named document "looks like" a file. In applets, jHelp will build a URL with the file name as described above.

It is possible to override this by explicitly naming your help document using a "file://" prefix, however in most cases this will generate a security exception and jHelp will not be able to retrieve the file.

The "file://" prefix is really provided for a different reason, which is described in more detail in another section of this documentation.

## Using Automated Indexing

htmlHelpIndexBuilder is a helper application that is used to build an index file and optional word list for display in htmlHelpIndexFrame. It is a java console application, executed at a command line. You may also use the Index Wizard, a project-oriented GUI front-end to the console application. The Index Wizard is described in the following section.



**jHelp's Index Frame**

htmlHelpIndexBuilder compiles index entries and global word lists and stores them as serialized files of htmlHelpIndexData objects. This class contains a numeric index that points to the index term's position within the help document, relative to tags, images and other words.



**IMPORTANT** - The numeric index that is used to determine the position within the document depends upon the content of the document. Therefore, if any help documents are revised, added to, or deleted from the help document set, it will be necessary to recompile the index file.

## Using htmlHelpIndexBuilder

Assuming the jHelp package is in the default CLASSPATH of the java runtime, htmlHelpIndexBuilder is executed with the following general syntax:

```
java jHelp.htmlHelpIndexBuilder [options] outfile infile
```

where:

*outfile* is the name of the index file to be created, fully qualified with a path if desired.

*infile* is an ASCII text file containing the names of the help files to be indexed, one file name per line. These should be the file names only, and not qualified with a path. The path to the help files is set with the -d switch described below.

Options:

Options are set using command line switches, preceded with either a '-' or '/' character, and are summarized in the table below:

-? or /?	Displays usage instructions.
-h- or /h-	Causes header tags to be ignored when creating index entries.
-h+ or /h+	Causes header tags to be used in generating index entries (default.)
-t- or /t-	Do not include the document title with the index entry.
-t+ or /t+	Include document titles in the index (default.) This is useful where duplicate terms are indexed that are contained within different documents.
-w- or /w-	Do not generate a word list
-w+ or /w+	Generate a word list (default) with the same file name as the index file, but with the extension ".wdx"
-d:path or /d:path	The path for the helpfiles named in inputfile. Note that there is no space separating the switch from the path. Note that a trailing path separator is required (see example below.)

## Usage Example

A typical command line for executing htmlHelpIndexBuilder on a Windows platform might be as follows:

```
java jHelp.htmlHelpIndexBuilder -h- -t-  
    -d:e:\myapp\helpdocs\ e:\myapp\helpdocs\help.idx  
    e:\myapp\helpdocs\doclist.txt
```

The above line will generate an index, and save it as the file `e:\myapps\helpdocs\help.idx`. It will look for a list of document files in the file `e:\myapps\helpdocs\doclist.txt`. Header tags within the help documents will be ignored, and not used to generate index entries. Document titles will not be displayed alongside of the index entries. And finally, all help files listed in `doclist.txt` are located in the directory `e:\myapp\helpdocs\`. Note the trailing path separator provided for the source directory.

Some aspects of the syntax, such as path separators, will of course vary from the example depending upon your operating system.

## INDEX Tags

The `<INDEX>` and `</INDEX>` tags are used to create index entries from contained text. In its simplest form,

```
<INDEX>an index item</INDEX>
```

will create a top level index item containing

### **an index item**

Index entries can be grouped level and indented. For example:

```
<INDEX>index item</INDEX>
<INDEX VALUE="index item">another item</INDEX>
<INDEX VALUE="index item">another second level item</INDEX>
```

results in:

### **index item**

**another item**

**another second level item**

### Adding

```
<INDEX VALUE="index item;another item">third level
item</INDEX>
```

produces

### **index item**

**another item**

**third level item**

**another second level item**

## KEYWORD Tags

The following syntax for setting off keyword index entries is implemented:

```
<KEYWORD VALUE="index item">
```

produces a 1st level index entry,

**index item**

Similar to the INDEX tag, adding

```
<KEYWORD VALUE="index item;another item">  
<KEYWORD VALUE="index item;another second level item">
```

produces secondary index items:

**index item**  
    **another item**  
    **another second level item**

Adding

```
<KEYWORD VALUE="index item;another item;third level item">
```

produces

**index item**  
    **another item**  
        **third level item**  
    **another second level item**

## Level Delimiters

By default, super-index items will be delimited by a semicolon. This can be overridden with a DELIM attribute if the semicolon is inappropriate, for example

```
<INDEX VALUE="first level|second level" DELIM="|">an  
item</INDEX>
```

or

```
<KEYWORD VALUE="first level|second level|an item" DELIM="|">
```

## Headers

If headers are enabled when building an index, then H1 items will be considered (and rendered) as first level items. H2, H3, H4... H6 will be rendered as indented sub-levels of the preceding higher level heading.

This behavior will essentially mimic a word processor's mechanism for generating a table of contents from header paragraphs.

## Document Titles

Document titles may still be optionally displayed alongside an index item.

## Backward Compatibility

The above syntax is backward compatible with jHelp 2.06 at the HTML level. However, these changes resulted in a revised `htmlHelpIndexData.class`, and a revised `htmlHelpIndexFrame.class` and correctly displaying an index from 2.06 will require rebuilding the index, as well as using the new version of `htmlHelpIndexFrame`. Method calls and constructors are unchanged.

## Indexing Tags Example

The following is a short HTML document demonstrating use of the above tags.

```
<HTML>
<HEAD>
<TITLE>Indexing Example</TITLE>
</HEAD>
<BODY>
<H1> Creating an Index</H1>
<P><STRONG><INDEX>htmlHelpIndexBuilder</INDEX></STRONG> is a
Java console application that automatically builds an
alphabetically sorted index from a list of help
documents.</P>
<KEYWORD VALUE="Running htmlHelpIndexBuilder">
<P>htmlHelpIndexBuilder is run from the command line. Syntax
for using htmlHelpIndexBuilder is as follows:</P>
<PRE>
java htmlHelpIndexBuilder [options] outputfile inputfile
</PRE>
<P><STRONG><INDEX>Command Line Options</INDEX></STRONG>
include enabling/disabling header tag indexing, including
the document name alongside of the index entry, naming a
source directory for the listed output files, and a help
screen.</P>
</BODY>
</HTML>
```

If we run `htmlHelpIndexBuilder` on the above HTML document, with default options to include header tags in the index, and the document title, the following sorted, indexed entries will be created:

**Command Line Options (Indexing Example)**  
**Creating an Index (Indexing Example)**  
**htmlHelpIndexBuilder (Indexing Example)**  
**Running htmlHelpIndexBuilder (Indexing Example)**



## Using the Index Wizard

First, ensure that the `jhelp.jar` is in the `CLASSPATH`. For example

```
set CLASSPATH=%CLASSPATH%;d:\YourDirectory\lib\jhelp.jar
```

would append `d:\YourDirectory\lib\jhelp.jar` to the current `CLASSPATH` on a Windows system (assuming that you extracted the distribution zip to a folder named `YourDirectory` on drive `d`.)

Then, at a system prompt, enter

```
java jHelp.index.Wizard
```

The above assumes that the java runtime interpreter is in your executable `PATH`. If not, you may have to execute with something like

```
c:\jdk1.1.3\bin\java jHelp.index.Wizard
```

depending, of course, upon where the java runtime interpreter is installed on your system.

Note that "jHelp.index.Wizard" is case sensitive. If your system is properly configured, the Index Wizard will appear, shown below.

### Project

The Index Wizard is project oriented. Your settings, such as the name of an index file to create, the help document directory, included documents and options will be saved so that you can easily rebuild your index, include new documents, and edit your settings.

By default, if you browse for an existing project, the wizard will look for files that have the `.iwp` extension. You can simply enter the name of a new project (`.iwp` will be appended if necessary) to create a new project in the current working directory. Or, you can use the browse button to navigate to a particular directory, type in the name of a new project in the file dialogs name field and press Open.

### Index File Name

If you open an existing project, this field will initialize to the index file of the project. You can create a new index file by entering the name in this field (`.idx`, the default extension, will be added if necessary.) You can also browse to a directory and enter the name of your index in the file dialog's name field, and press Open.



### The Index Wizard's Opening Panel

#### Document Directory

In this field, you enter the path to your help documents. The wizard will attempt to set this for you, based on either the current project, or target index directories. If the value displayed is not the correct directory, enter the path to your help files. You do not have to append a trailing slash, but doing so does no harm.

#### Include Headers in Index

Checking this option will enable use of Header tags (H1, H2, etc.) to create nested index entries from the header text, similar to the way a word processor uses header paragraphs to generate a table of contents.

#### Show Document Titles in Index

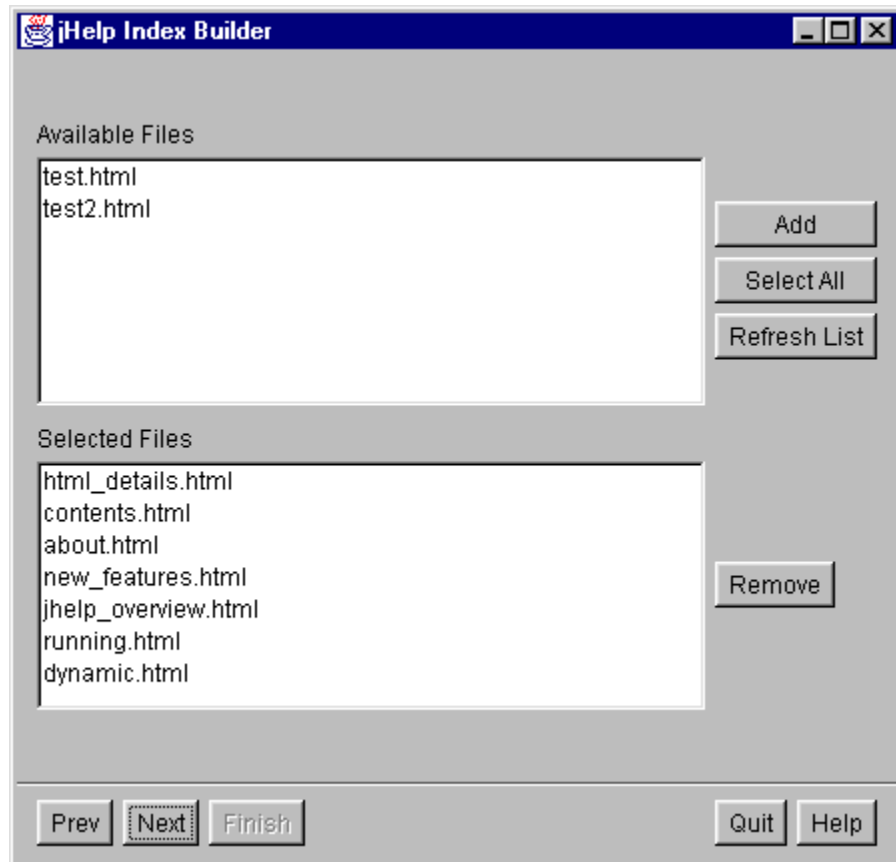
This option will display the document title in parentheses next to each index item.

#### Generate Word List

Checking this option will generate a word list, used for global word/phrase searches.

**Next...**

When you have provided, or reviewed the above information, press the "Next" button near the lower left corner of the Wizard Frame.

**The Wizard's File Selection Panel**

The next panel lets you select and add document files to your project. You may also remove project files. To add files to the project, you can double click on the file displayed in the "Available Files" list, single click on each file to add and click on the "Add" button to add the multiple selection, or click "Select All" and then click the "Add" button to add the entire list.

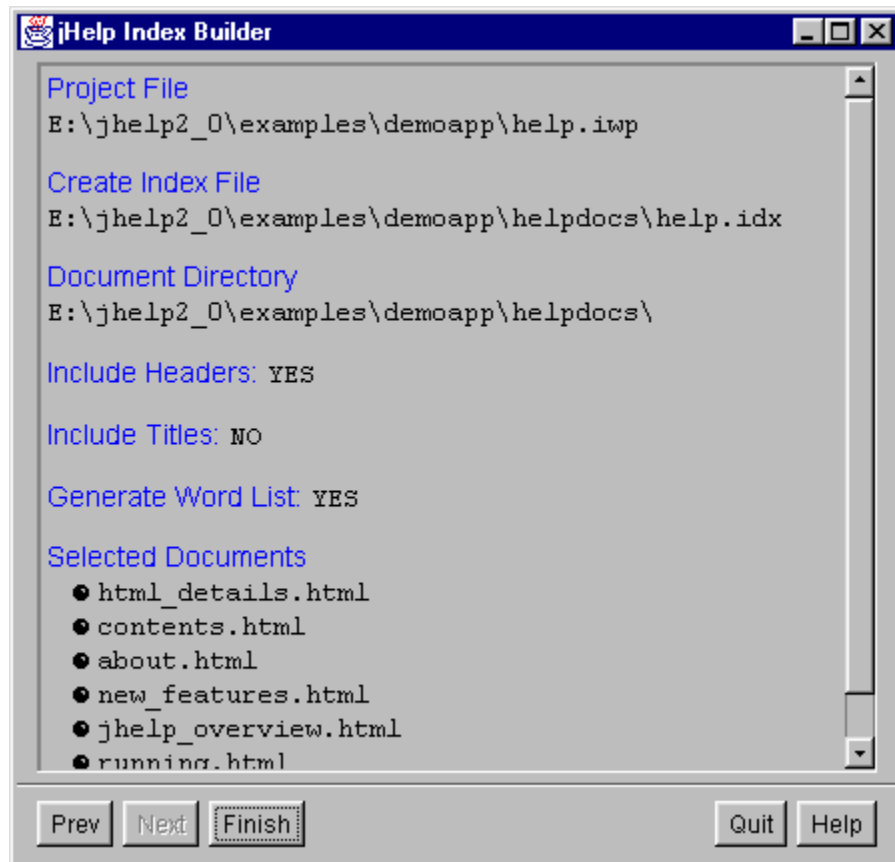
To remove files from a project, double click on individual files in the "Selected Files" list, or single click on multiple files and then click on the "Remove" button.

If you add a new HTML file to the document directory while this panel is visible, pressing the "Refresh List" will bring it into the "Available Files" list.

**Next...**

Press the "Next" Button when you are finished selecting/removing document files.

The next panel lets you review the settings, file names, and selected documents prior to building the index.

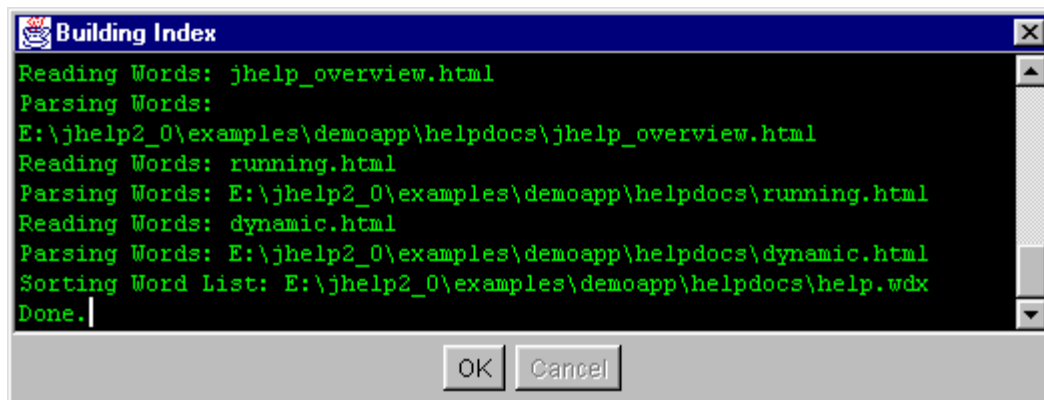


**The Wizard's Review Panel**

### **Finish...**

Review the data, and if everything is satisfactory, press "Finish." You will then be asked if you want to save the current project settings.

Press "Yes" to save, or "No" to proceed without saving. The Index Wizard will then invoke jHelp's index builder, and you'll be able to watch the build in the dialog that appears.



### The Wizard's Progress Display

When jHelp has finished parsing the files, the "OK" button will enable and you can close the progress window.

## Managing Help Document Directories

It would be silly to expect application developers to store help documents in the same directory as an application's class files. Typically, help documentation will be placed in its own directory, probably a subdirectory of the application's main directory. jHelp is designed to accommodate this.

Starting with jHelp 2.06, you may now package your application's help documents and optional index file in a compressed zip or jar file. You may even pack your documents in the same jar that contains your application's class files. The `htmlLocator` class provide methods to let you determine your resource locations programmatically, whether your program is jarred, zipped, or provided as unarchived classes.

Note that separately zipped or jarred help files are not supported in applet implementations of jHelp. However, you can package your help documents and images in the same jar or zip in which your applet is provided

### jHelp's Constructors

When you construct an `htmlHelpFrame`, `htmlHelpPanel`, or `htmlHelpIndexFrame`, one of the values that is passed is the name of a help document that is the document initially loaded into the jHelp object. (You may also pass a dynamic HTML string, but this discussion is not concerned with that aspect. Dynamic HTML is described in another section of this documentation.) Optionally you may construct a jHelp object with the name of a zip or jar file (applications only, not necessary or permitted with applets,) if you choose to supply documentation in that format.

The filename that is passed will be taken as is (except in applets.) For example,

```
new htmlHelpFrame(frame, "e:/myapp/docs/contents.html")
```

will cause jHelp to retrieve the file from drive e:, as it is specified.

```
new htmlHelpFrame(frame, "/myapp/docs/contents.html")
```

will retrieve the file from /myapp/docs on the current drive.

```
new htmlHelpFrame(frame, "docs/contents.html")
```

will retrieve the file from docs, relative to the current working directory.

Applets will always use URL methods, and will attempt to build a URL from the file specification (relative to the code base of the applet using

`java.lang.Class.getResource()`, unless preceded with an `http://` protocol specifier.)

**NEW in 2.06** - You can construct an empty jHelp panel or frame by passing null as the filename argument.

### Using Zips and Jars

To use a zip or jar, when you construct your jHelp object, instead of supplying a document, provide the name of the zip or jar file that contains your documents and associated images. Note that this only applies to applications.

Your zip/jar file should contain your documents, and they may be organized in a directory structure within the zip, exactly as you would deploy them without a zip file.

### Path Separators

Since Java is intended to be deployed on different platforms, which probably use different path separator characters, it would be limiting to hard-code platform specific file names in an application.

For example, `docs\contents.html` would work on a Windows system, but would be inappropriate for Solaris.

To accommodate platform independence, jHelp supports use of '/' as a universal file separator. You can specify your files using forward-slashes as file separators, regardless of your platform.

At runtime, jHelp will resolve forward-slash characters to system-dependent path separators.

### Path separators in Zips and Jars

If you are using a zip/jar to contain your help documents, it is especially important that you use forward slashes as path separators.

### Setting BASE Directories

Although you must provide an explicit file specification in the constructors for jHelp objects, it would be impractical and bulky to require a full specification every time a new document was called into the help browser.

To facilitate calling help documents simply by name without a path, two methods are provided:

```
htmlHelpFrame.setShowBase(String path)
```

```
htmlHelpPanel.setGoToBase(String path)
```

After constructing a help frame or panel, these methods are used to set a default directory for retrieving help documents.

For example, suppose an `htmlHelpFrame` object is constructed with

```
htmlHelpFrame help =  
    new htmlHelpFrame(thisFrame, "docs/help/contents.html");
```

Instead of explicitly referring to new documents with the "docs/help/" path, you would set the help base directory with

```
help.setShowBase("docs/help/");
```

Following this, you can display documents from this directory by simply calling for them by name, as in

```
help.show("files.html");
```

which would actually load "docs/help/files.html." If the `jHelp` object is an `htmlHelpPanel`, then use `setGoToBase(String path)`.

Of course, you can override the base directory by preceding a file with the "file://" prefix, and then providing a fully qualified file specification for the document.

In a similar fashion, the base will not be applied to documents named with the "http://" prefix, as those files will be loaded as URL's over a network.

### **Base directories with Zips and Jars**

If you are supplying your documents in a zip or jar, use `setShowBase()` or `setGoToBase()` the same as if your documents are unpacked. For example, suppose you construct an `htmlHelpFrame` object with

```
htmlHelpFrame help =  
    new htmlHelpFrame(thisAppFrame, "myapp.jar");
```

You then use `setShowBase`,

```
help.setShowBase("docs/help/");
```

and then, you want to display a document with

```
help.show("about.html");
```

Based on the above, `jHelp` will look for a zip entry named



`docs/help/about.html`

in the jar file `myapp.jar`.

### Using `htmlLocator` Methods with an Application

If your developing an application, it is likely that you will package it in a jar or zip file, that your users will place in their classpath. It is recommended that you package your help documents in the same jar or zip as your application.

You can easily locate your applications jar using the static methods of `jHelp.htmlLocator`. These methods work even if you do not jar/zip your application, but provide your helpdocs in your app's main directory (or a subdirectory of it.)

For example, suppose your application is in `myapp.jar`, and your help files are in a `docs` subdirectory within the jar. The following code will construct a `jHelp` object, and set the base directory. (This code will also work if you extract the jar'd files.)

```
htmlHelpFrame help;
```

```
help = new htmlHelpFrame( this,  
                          jHelp.htmlLocator.getZip(getClass(),"docs/about.html"),  
                          385, 400);  
help.setShowBase( jHelp.htmlLocator.getPath(getClass(),"docs/about.html")  
                  + "docs/");
```

## Odds 'n' Ends

A few important topics concerning jHelp remain to be discussed.

### Extending htmlHelpPanel

While primarily intended to be used as a help viewer, jHelp's components are also finding use as fancy read-only scrollers. In such cases, the developer might want to provide methods for handling hyperlinks in displayed documents that do application specific tasks, beyond the default processing.

For example, you might be using an extended htmlHelpPanel to display an index of names in a database, each name being a hyperlink. Or you might want to perform special handling for a mailto hyperlink. Or, you might want to spawn a web browser for hyperlinks that reference WWW URL's.

To facilitate this, developers may extend htmlHelpPanel, and override the handleLink method.

Sample code is provided below:

```
public class ExtendedPanel extends htmlHelpPanel
{
    public ExtendedPanel(Object instance)
    {
        super(instance,null);
    }

    public void handleLink(htmlLinkEvent e)
    {
        // htmlLinkEvent contains the href attribute
        // of the hyperlink that was clicked.
        String command = e.getLinkCommand();

        if(command.startsWith("mailto")){
            //Do your own thing,,,
        }

        // Otherwise, let jHelp do its regular thing
        else{
            // The "super" class name is not really
            // necessary below, but used only for clarity.
            super.handleDefaultLink(e);
        }
    }
}
```

## Document Caching

Remember we said that jHelp originally was intended to be used with applets? In order to eliminate the need for redundant downloads of previously loaded files, and also to reduce time spent re-parsing help files, jHelp caches help documents in memory by default.

With a small system of help files, this will probably not be a problem. However, with an extensive system of help files, a user navigating through a large number of files might consume an undesirably large amount of system resources.

To minimize the drain on resources, jHelp provides methods to limit the number of cached documents.

```
htmlHelpFrame.setMaxCache(int number_of_files);
```

```
htmlHelpPanel.setMaxCache(int number_of_files);
```

With these methods, you can specify the maximum number of documents that will be cached in memory. jHelp will dispose of the oldest document in the cache each time the maximum number is reached.

## Dynamic HTML

Support for dynamically generated HTML content is a valuable feature that lets developers use jHelp objects as sophisticated viewers of text generated by the applet/application.

### Setting Dynamic Content

Dynamic HTML content should be contained within a single String object, and passed to jHelp using either

```
htmlHelpFrame.addHTMLString(  
    Object instance,  
    String content,  
    String name)
```

for pop-up frames, or

```
htmlHelpPanel.addHTMLString(  
    Object instance,  
    String content,  
    String name)
```

for htmlHelpPanel widgets.

## Displaying Dynamic Content

If you simply want to display some non-persistent HTML, pass null for the name parameter of the methods. This will create a reusable `htmlVector` object (jHelp's internal representation of your help document, consisting of tags, images, and text) named `htmlHelpPanel.DYNAMIC`

To display the dynamic HTML, refer to it with this default name,

```
htmlHelpFrame.show(htmlHelpPanel.DYNAMIC);
```

or for the widget version,

```
htmlHelpPanel.goTo(htmlHelpPanel.DYNAMIC);
```

If you want to create several dynamic `htmlVector` objects, and possibly refer to them individually, you must give each a unique name. You can overwrite a named dynamic HTML object by calling the `addHTMLString()` method with the same name used previously.

To refer to a named dynamic `htmlVector` object, you must precede the name with the dynamic: prefix when using the name with `show()`.

This tells jHelp not to look for this name as a file to load. For example, if you created a dynamic HTML object with

```
htmlHelpFrame.addHTMLString(instance, anHTMLString, aName);
```

then you would show it with

```
htmlHelpFrame.show("dynamic:" + aName);
```



**Important Note:** If you are limiting the number of cached documents (see **Document Caching**, above,) you will probably have to reset your dynamic content from its String source, as you will not be able to depend upon the persistence of a named dynamic element in the cache.

## Minimizing Distribution Size

An important consideration for redistribution of jHelp classes with an application is the elimination of unnecessary package files.

When you provide the jHelp package with your application, the following are NOT required for end-users:

```
jHelp.index.*;  
jHelp.htmlHelpIndexBuilder.class  
moondog.*
```

The above packages and class files are used only for building compiled indexes and word lists, and are not needed to use jHelp as a subsystem in your program.

## **Ordering, Upgrades, & Technical Support**

Moondog Software provides a liberal upgrade policy, and has been praised for the quality and prompt response to questions of a technical nature.

### **Upgrades**

jHelp licensees are entitled to one year of free upgrades from the date of delivery of jHelp. Licensees are automatically notified by email of significant version releases and bug fixes, with instructions for obtaining the latest package over the Internet.

Source code licensees may request source upgrades by email.

After the one-year period expires, licensees will have the opportunity to purchase upgrades and renew the one-year upgrade/support period at a substantial discount off the standard license fee.

### **Technical Support**

jHelp Tech Support is provided directly by its author:

bgiel@ct2.nai.net

### **Ordering and Licensing jHelp**

jHelp Licenses can be purchased by email or FAX of company Purchase order to:

Email: bgiel@ct2.nai.net

FAX: 203.877.5586

License effective upon receipt of PO, Moondog Software will submit invoice, terms of payment net 30 days unless other arrangements are made. jHelp product will be shipped via email. Please designate recipient and contact person for software.

Credit cards are not accepted at this time.

For information pertaining to licensing options, or ordering jHelp, please contact the author:

bgiel@ct2.nai.net

## Appendix A - HTML DETAILS

### Unsupported

The following HTML tags or attributes are **not** supported:

ISINDEX  
LINK  
FORM  
META  
NEXTID  
TAB  
TABLE  
XMP  
COMPACT attribute in lists  
WIDTH attribute in PREformatted text  
WIDTH, HEIGHT, ALT, ISMAP attributes of IMG.

For IMG tags, note that only BOTTOM and MIDDLE values for ALIGN are supported. ABSMIDDLE is treated as MIDDLE. Default image alignment is BOTTOM.

### Supported

The following HTML tags which are not defined in HTML 2 **are** supported:

#### **<BASE HREF=...>**

HREF=path - Can be used to override the default base path of the document for links.

#### **<BASEFONT FACE=... SIZE=...>**

FACE=typefacename - Used to override the default font face (Serif.) It is recommended that Java 1.1 names be used, Serif, Sansserif, and Monospaced.

SIZE=size - Override the size of the default font (3.) Note that the value used is the HTML-style size, and not the actual pointsize.

#### **<DIV ALIGN=...></DIV>**

ALIGN=LEFT, RIGHT and CENTER - Division alignment is implemented to facilitate generating documents with HTML 3.2 editors.

#### **<H# ALIGN=...></H#>**

ALIGN=LEFT, RIGHT and CENTER - Header alignment is implemented to facilitate generating documents with HTML 3.2 editors.

**<INDEX VALUE=... DELIM=...></INDEX>**

A special tag used to set off a word or phrase that will be used in generating an index. The index tag is described in detail in the section on using `htmlHelpIndexBuilder`.

**<KEYWORD VALUE=... DELIM=...>**

VALUE="word or phrase" - A special tag used to include a word or phrase in the index, but not displayed in the document. The keyword tag is described in detail in the section on using `htmlHelpIndexBuilder`.

**<OL START=...>**

START=number - overrides default starting number(1) in ordered lists. In nested list, will produce corresponding alphabetic character, e.g. 1=A, 2=B, etc.

**<P ALIGN=...></P>**

ALIGN=LEFT, RIGHT and CENTER - Paragraph alignment is implemented to facilitate generating documents with HTML 3.2 editors.

**<S></S>**

Strikethrough contained text.

**<UNDERLINE ON> and <UNDERLINE OFF>**

Turns hyperlink underlining on or off.

**Coded Characters**

Coded characters include those characters of the HTML 2 document set as set forth in ISO-8859-1 and specified in "SGML Declaration for HTML."

While not a part of the HTML 2 specification, the following named entities may be used in lieu of standard coded characters:

&lt; - less-than

&gt; - greater-than

&quot; - double-quotation mark

&nbsp; - non-breaking space

&copy; - copyright symbol

&reg; - registered trademark;



## Appendix B - Standard License

### DEVELOPERS' LICENSE AGREEMENT FOR jHELP SOFTWARE

Copyright (C) 1997 by William Giel, referred to herein as the DEVELOPER.

**IMPORTANT-READ CAREFULLY:** This License Agreement ("LA") is a legal agreement between you (either an individual or a single entity) and the DEVELOPER of the software product(s) identified above ("SOFTWARE PRODUCT" or "SOFTWARE"). The SOFTWARE PRODUCT includes computer software, the associated media, any printed materials, and any "online" or electronic documentation. By installing, copying or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this LA. If you do not agree to the terms of this LA, the DEVELOPER is unwilling to license the SOFTWARE PRODUCT to you. In such event, you may not use or copy the SOFTWARE PRODUCT, and you should promptly contact the DEVELOPER for instructions on return of the unused product(s) for a refund.

#### SOFTWARE PRODUCT LICENSE

The SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold.

1. GRANT OF LICENSE. This LA grants you the following rights:

\*Software. You may use, redistribute and include jHelp classes with any software product deploying it as a help system, or similar function.

Usage and inclusion are understood to mean deployment of jHelp as a component of an application or applet to enhance the functionality of said applet or application, together with redistribution of jHelp's compiled class files along with said application or applet.

Source code, if licensed, is not redistributable.

2. DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS.

\*Limitations on Reverse Engineering, Decompilation and Disassembly. You may not reverse engineer, decompile, or disassemble the SOFTWARE PRODUCT, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation. Licensed source code may be modified as required for specific uses, however such modification shall relieve DEVELOPER of any obligations to provide technical support for the SOFTWARE PRODUCT.

\* Termination. Without prejudice to any other rights, the DEVELOPER may terminate this LA if you fail to comply with the terms and conditions of this LA. In such event, you must destroy all copies of the SOFTWARE PRODUCT and all of its component parts.

3. UPGRADES. If the SOFTWARE PRODUCT is an upgrade, you now may use that upgraded product only in accordance with this LA. If the SOFTWARE PRODUCT is an upgrade of a component of a package of software programs which you licensed as a single product, the SOFTWARE PRODUCT may be used and transferred only as part of that single product package.

4. OEM COPYRIGHT. All title and copyrights in and to the SOFTWARE PRODUCT (including but not limited to any images and "applets," incorporated into the SOFTWARE PRODUCT), the accompanying printed materials, and any copies of the SOFTWARE PRODUCT, are owned by the DEVELOPER.

5. OEM U.S. GOVERNMENT RESTRICTED RIGHTS. The SOFTWARE PRODUCT and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of the Commercial Computer Software-Restricted Rights at 48 CFR 52.227-19, as applicable.

**6. THE DEVELOPER HEREBY DISCLAIMS ANY AND ALL LIABILITY, EXPRESS OR IMPLIED IN CONJUNCTION WITH THE USE OR INABILITY TO USE THIS SOFTWARE. THIS DISCLAIMER OF LIABILITY EXTENDS TO ANY PHYSICAL OR PECUNIARY DAMAGES, INCLUDING THE LOSS OF BUSINESS INFORMATION, ANTICIPATED PROFITS, OR OTHER DAMAGES, EVEN IF ADVISED OF SUCH DAMAGES.**

## Appendix C - Current Licensing Options

Licensing Options as of September 15, 1997

jHelp is neither freeware nor shareware. It is available to developers with a royalty-free license for unlimited distribution of the compiled package as a help or document viewing sub-system of a program. Source code is available.

Option	Redistribution	Price with Source	Price without Source
Evaluation	0	\$ 0.00	Not Available
Standard	Unlimited	\$ 650.00	\$ 450.00

All prices are in U.S. Dollars.

Source code is not redistributable under any of the available options.

jHelp Licensees are entitled to unlimited email or telephone technical support and free upgrades for a period of one-year from the shipping date of the product.