Contents

DITA and User Assistance

This topic describes the relationship between DITA and user assistance and Help systems in particular.

DITA can be used in the process of creating user assistance, and especially of Help systems, but is not currently (and may never be) used as a user assistance format itself.

DITA is a storage and authoring format, not a delivery format; it is a presentation-neutral format. The separation of content and form is fundamental to DITA's design; content is written in DITA, and must be transformed to a reading format before it can be delivered to the reader.

In principle, content written in DITA can be *transformed* to any reading format. In practice, it's not that simple. Before DITA can be transformed, a transformation process has to be devised. Many DITA authoring and publishing tools come with standard transformers for most common delivery formats, such as PDF, RTF and HTML. The *DITA Open Toolkit*, an open source collection of utilities and documentation to help writers work with DITA, includes basic transformers for PDF, RTF, HTML, DocBook, Eclipse Help, and Microsoft® HTML Help.

User assistance content is not defined by its format. A document in Microsoft HTML Help format isn't necessarily a Help system; user assistance is defined by the nature of the content. Conversely, user assistance content doesn't have to be delivered in a *traditional* Help format.

DITA promotes a single-source approach to documentation, so user assistance may commonly be one of a number of deliverables produced from a *repository* of information topics. The process of producing simple Help systems from DITA content using the standard DITA Open Toolkit transformers is straight-forward. It is a little more complicated to deliver such DITA-generated content for context-sensitive Help, but still readily achievable. Likewise, in principle, it is a trivial matter to incorporate DITA content into embedded user assistance and user interface elements, using standard XML tools and techniques. There is not yet a standard approach to user assistance, so there is also no standard way of using DITA in this way. Different organisations tend to develop their own individual, custom approaches, using inhouse technical expertise to do so.

Moving beyond simple Help systems, however, is currently difficult, but not impossible. The DITA Technical Committee is developing some changes to the DITA standard to allow these processes to be simplified. However, the apparent simplicity or complexity of using DITA for Help authoring will be in future determined by the capabilities of DITA editing and publishing tools. When it comes down to it, DITA is just a standard, and good tools are needed to work with good standards.

Definition of DITA Help

This topic details how the OASIS DITA Help Subcommittee defines the term *DITA Help*.

DITA Help, as defined by the OASIS DITA Help SC (DHSC), is:

A set of recommendations for the design and implementation of commonly recognized user assistance components using the DITA architecture. These components include, but are not limited to, navigation components, context-sensitive linking, embedded Help, browse sequences, associative links, window definitions. The recommendations can be used as a foundation for the development of authoring models and tools to support solutions in a variety of formats and for a variety of platforms.

Developing DITA-based Help for Existing Help Environments

This topic introduces solutions for moving your DITA source content into one of several existing help run-time environments.

If your company delivers Help information to a commonly used help viewer or run-time help environment, you have options for authoring content in DITA and for transforming that content into ready-to-run help deliverables.

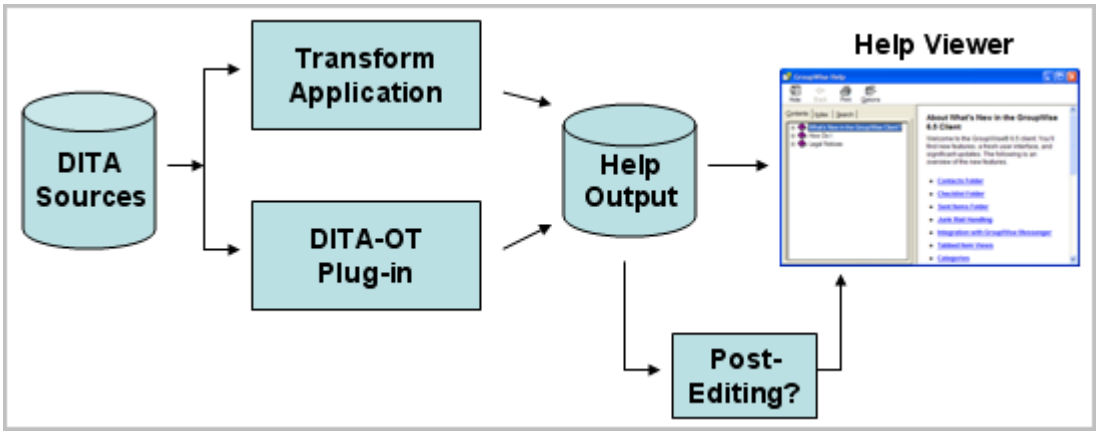


Figure 1: DITA Source to Existing Help Environments

The following topics in this section of the *Best Practices Guide* explain how to use one or more of these plug-ins or transform applications to generate Help from DITA sources.

Help Environment	Notes
Adobe AirHelp Plug-in	The AirHelp plug-in does ...
CSHelp Plug-in	The CSHelp plug-in does ...
IBM EclipseHelp Plug-in	The EclipseHelp plug-in does ...
Microsoft HTMLHelp Plug-in	The HTMLHelp plug-in does ...
PTC PTCHelp Application	The PTCHelp plug-in does ...
HyperWrite WinAnt Help Outputs	WinAnt generates multiple Help outputs ...

CSHelp Plug-in

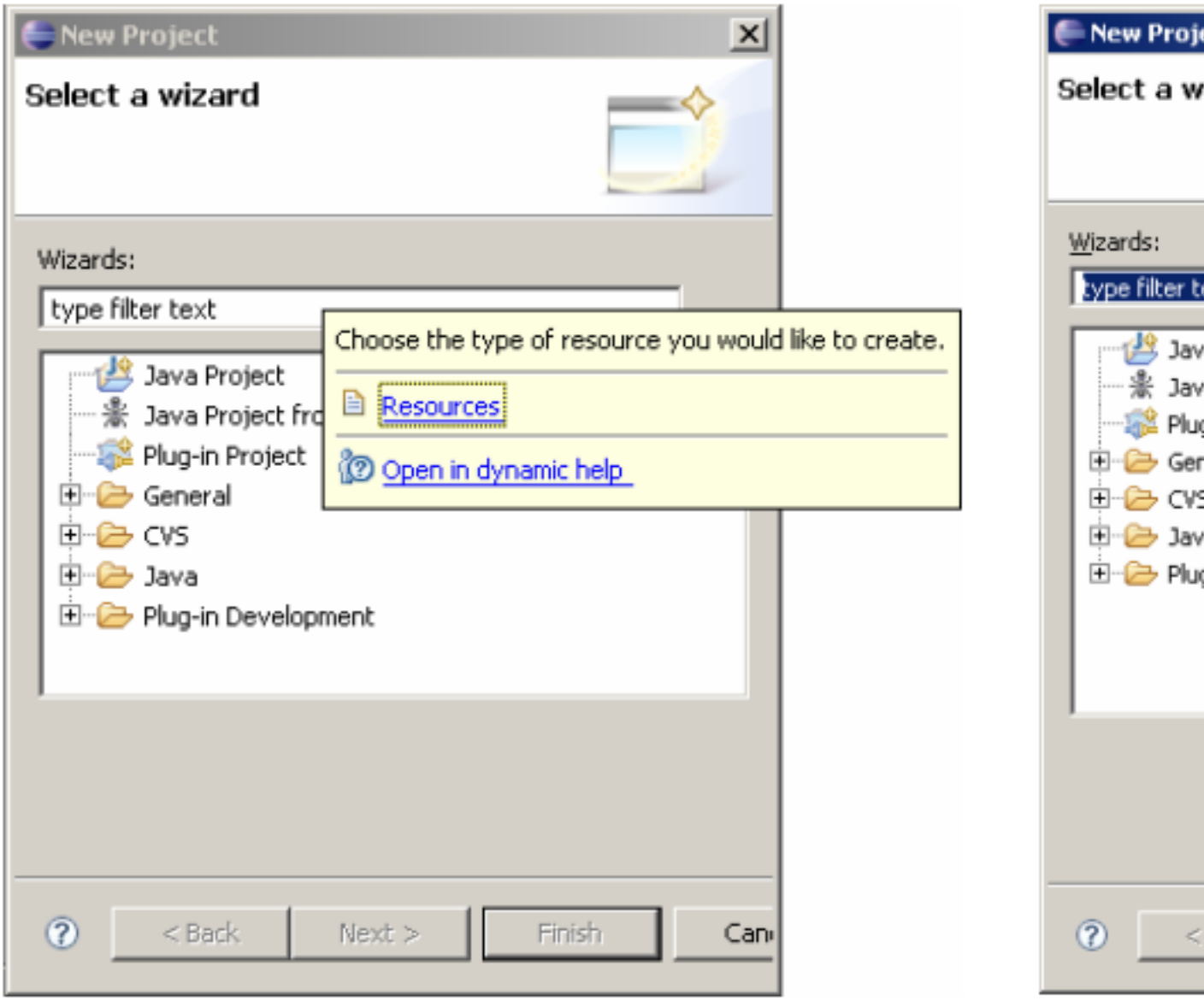
This topic provides an overview of the DITA Open Toolkit (DITA-OT) plug-in named cshelp (for context-sensitive help).

The cshelp plug-in generates contexts files in the format that Eclipse-based applications use for context-sensitive help. The standard DITA-OT transforms can be used to produce XHTML output. This chapter focusses on producing Eclipse contexts files.

Overview

If you develop code plug-ins that extend the Eclipse user interface, or develop documentation for development teams that do, you either already are or should be incorporating context-sensitive help into the user interface.

The Eclipse user interface allows you to display context-sensitive help as an infopop (left) or in a dynamic help view. The latter option includes information and functionality in addition to the contents of the context-sensitive help topic.



For more information on contexts files and developing context-sensitive help for Eclipse, refer to the documentation for the current Eclipse release, available on *http://www.eclipse.org*.

The cshelp plugin was developed by a team of writers representing the various software brands in IBM, led by the author. It is currently in use by products in several brands that develop applications for the Eclipse environment.

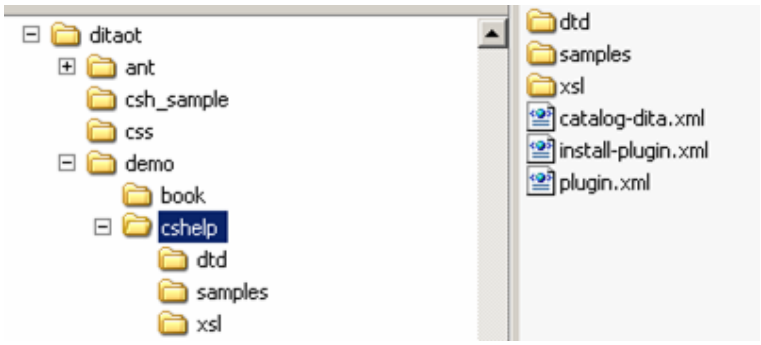
Setup and configuration

The cshelp plug-in is available from the DITA-OT site on sourceforge.net:

http://sourceforge.net/project/showfiles.php?group_id=132728

Package	Release	Date	Notes / Monitor
dita-ot	DITA Open Toolkit 1.4	August 12, 2007	
dita-ot under Apache License	DITA OT 1.2.2 ASL	May 22, 2006	
Plug-in: apiref	apiref-0.8	February 23, 2006	
Plug-in: cshelp	cshelp1.1	April 2, 2007	
Plug-in: dockbook2dita	docbook2dita 1.0	October 26, 2006	

Download and unzip the plug-in into your DITA-OT installation (typically, the `<DITA-OT>/demo` directory).



To complete the installation, open a command prompt and run the Ant integrator build file from your DITA-OT directory (`ant -f integrator.xml`).

Authoring

The structure of a cshelp DITA file mirrors that of an Eclipse contexts file. A contexts file is an XML file that contains one or more context elements inside a containing contexts element. Each context element contains a unique ID, the text of the context-sensitive help topic, and optionally, links to help topics in the help system.

```
<?xml version="1.0" encoding="UTF-8"?>
<?NLS TYPE="org.eclipse.help.contexts"?>
<contexts>
    <context id="new_wizard_selection_wizard_page_context">
        <description>
         Choose the type of resource you would like to create.
        </description>
        <topic label="Resources" href="concepts/concepts-12.htm"/>
    </context>
    …
</contexts>
```

Similarly, a cshelp DITA file contains one or more cshelp elements inside a containing cshelp element (which is otherwise unused). Each nested cshelp element contains a unique ID, text, and related links.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cshelp PUBLIC "-//OASIS//DTD DITA CSHelp//EN"
 "..\demo\cshelp\dtd\cshelp.dtd">
<cshelp id="csh_outer_container" xml:lang="en-us">
  <title>sample1_context</title>
  <shortdesc></shortdesc>
  <csbody></csbody>
  <cshelp id="new_wizard_selection_wizard_page_context">
    <title></title>
   <shortdesc>Choose the type of resource you would like to create.</shortdesc>

    <csbody>
    </csbody>
    <related-links>
      <link format="htm" href="concepts/concepts-12.htm" scope="peer">
```

```
        <linktext>Resources</linktext>
      </link>
    </related-links>
  </cshelp>
  ...
</cshelp>
```

It is important to note that the only highlighting markup that can be used inside Eclipse context-sensitive help is the bold () tag, and the only formatting options are the carriage return and the space key. All of these are permissible only inside the description element. When sourcing context-sensitive help in DITA, all of the highlighting and formatting markup options that are available in the shortdesc and body elements of the topic type may be used. The transform that produces a contexts file from the DITA source produces output that conforms to the restrictions of the contexts file. In most cases, the output approximates what would be seen in HTML output, but some DITA elements are simply ignored:

- Table and simpletable
- Image (only the text in the <alt> attribute or tag will display)
- Object
- Figure (only the text in the <desc> tag will display)
- Footnote
- Indexterm, indextermref
- Xref

Use as many DITA files per Eclipse plug-in that you want, and create a simple DITAMAP file inside the plug-in that points to each of them. Include any copyright information in the DITAMAP; this information will appear in comment tags in each generated contexts file. Note, however, that relationship tables (reltables) cannot be used to create related links when producing Eclipse contexts files.

Integration

Refer to the Eclipse documentation for instructions on incorporating context IDs in code plugins. This information is typically in Platform Plug-in Developer Guide > Programmer's Guide > User assistance support > Help > Context-sensitive help > Declaring a context ID.

http://www.eclipse.org/documentation/

Output

When processing the DITAMAP file, there are two parameters that need to be set:

- Use the switch that identifies a separate XSL file, and point to the dit2context.xsl file in the cshelp/xsl directory. For example, if you are using Ant file:

  ```
  <property name="args.xsl"
  value="${dita.dir}${file.separator}demo${file.separator}cshelp${file.separator}xsl${file.separator}dit2context.xsl"/>
  ```

- Use the switch that indicates the output extension of the generated file will be XML. For example, if you are using Ant:

  ```
  <property name="args.outext" value="xml"/>
  ```

The contents of the generated XML file should resemble the example contexts file in the Authoring section.

Summary

Context-sensitive help is an integral part of the user assistance for software applications. If you already use DITA to source your Eclipse documentation plug-ins, sourcing the content-sensitive help in DITA allows you to maintain a consistent authoring and build environment and consistently formatted output.

Jeff Antley

OASIS DITA Help Subcommittee

Eclipse Help

This topic provides an overview of how to use the DITA Open Toolkit to develop for the Eclipse help system.

According to the Eclipse Web site, the Eclipse integrated development environment (IDE) is an "open source platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle." Eclipse can be installed on Windows, Mac OS X, or Linux (32–bit or 64–bit).

The Eclipse help system is itself extensible, allowing additional documentation to be delivered using Eclipse's plug-in framework. It can be used in three different modes: workbench, standalone, and infocenter. Workbench mode is for documentation for tools integrated into the Eclipse IDE. Standalone mode serves a similar purpose, but is used for software that is not Eclipse-based. Infocenter mode allows the Eclipse help system to deliver topics through the World Wide Web. Local instances of the Eclipse help system can also retrieve and integrate content from a remote server running help in infocenter mode.

The Eclipse help system contains search and indexing features, allows the triggering of workbench actions or Eclipse code from within help topics, supports dynamic content and link filtering (when using XHTML), and supports globalization. You can even use Eclipse to manage the development, building, and testing of your DITA-sourced documentation plug-ins.

Overview

The Eclipse help displays HTML- or XHTML-formatted help and organizes the help topics according to the XML-formatted Table of Contents (TOC) files that are provided in each plug-in that contains documentation.

For more information on developing help for Eclipse, refer to the documentation for the current Eclipse release, available on *http://www.eclipse.org*.

Setup and configuration

DITA Open Toolkit

The DITA Open Toolkit contains the transformations necessary to produce all of the files required for Eclipse help plug-ins. No special setup or configuration is necessary. Refer to installation and configuration instructions within the DITA Open Toolkit to set it up.

Eclipse

Download Eclipse from *http://www.eclipse.org*. Installation involves unpackaging an archive file to any location on your machine. A Java Runtime Environment (JRE) is required. Eclipse manages your development resources in workspaces, that also can be any location on your machine. You are prompted to select or create a workspace location to use each time you start Eclipse.

Eclipse documentation plug-ins

Documentation for the Eclipse help system can be included in any plug-in, as long as the plug-in extends Eclipse's `org.eclipse.help.toc` extension point. It is up to the individual organization whether to include documentation inside code plug-ins or in their own plug-ins. There are several advantages to the latter, such as unambiguous ownership or if content will be globalized.

Plug-ins are named according to Sun's Java package naming guideline (for example, `org.eclipse.help`). Plug-ins that contain only documentation typically have `.doc` at (or near) the end of the plug-in name.

Within a plug-in, two XML files are required in the root (a manifest and a table-of-contents (TOC) file). Any number of content files may be included, in any location within the plug-in. The TOC file can be generated from a DITAMAP file, and HTML files will be generated from DITA files.

Plug-ins can be delivered to the Eclipse runtime environment as folders or as Java archive (JAR) files. Documentation within folders may exist in archives (for example, doc.zip). Archives cannot be nested (that is, a doc.zip cannot be included in a plug-in delivered as a JAR file).

Authoring

Author DITA topics as you would normally. You can include any number of topic files, folders, DITAMAPs, or other file-based resources that can be delivered within browser environment (for example, images or multimedia).

When creating links to other topics (XREFs or LINKs), note links to topics in other plug-ins should be coded as follows:

```
PLUGINS_ROOT/PLUGIN_ID/path/to/target.html
```

Where `PLUGINS_ROOT/` indicates that the target file is in another plug-in and `PLUGIN_ID` is the ID of the plug-in as declared in the manifest file. Refer to the Eclipse documentation regarding Help server and file locations in Help content: *http://www.eclipse.org/documentation/*.

Integration

To integrate your content into Eclipse, manually create a plugin.xml file that points to one or more TOC files in the plug-in, and a manifest.mf file (in a META-INF folder).

Plugin.xml file

This example plugin.xml file is similar to the one provided with the Garage sample in the DITA-OT. This example shows the minimum amount of information required to declare a TOC file to the org.eclipse.help.toc extension point.

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin>
 <extension point="org.eclipse.help.toc">
  <toc file="hierarchy.xml"/>
 </extension>
</plugin>
```

Manifest.mf

In current versions of Eclipse, some of the manifest information, such as the plug-in ID, is separated into the `manifest.mf` file. Manifest.mf is stored in the plug-in in a folder named `META-INF` For the Garage sample, here is a possible manifest:

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Garage Plug-in
Bundle-SymbolicName: org.dita.garage.doc
Bundle-Version: 1.0.0
```

Note that what was the plug-in ID in the plugin.xml file is referred to as the Bundle-SymbolicName in the manifest.mf file.

TOC file

TOC files are generated from DITAMAP files. You may include any number of TOC files in a plug-in, as long as they are declared in the plugin.xml file.

From the Garage sample, here is hierarchy.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<?NLS TYPE="org.eclipse.help.toc"?>

<toc label="Garage (hierarchy)" topic="concepts/garagetasks.html">
<topic label="Garage Tasks" href="concepts/garagetasks.html">
<topic label="Organizing the workbench and tools" href="tasks/organizing.html"/>
<topic label="Taking out the garbage" href="tasks/takinggarbage.html"/>
<topic label="Spray painting" href="tasks/spraypainting.html"/>
<topic label="Washing the car" href="tasks/washingthecar.html"/>
</topic>
<topic label="Garage Concepts" href="concepts/garageconcepts.html">
<topic label="Lawnmower" href="concepts/lawnmower.html"/>
<topic label="Paint" href="concepts/paint.html"/>
```

```
<topic label="Shelving" href="concepts/shelving.html"/>
<topic label="Tool box" href="concepts/toolbox.html"/>
<topic label="Tools" href="concepts/tools.html"/>
<topic label="Water hose" href="concepts/waterhose.html"/>
<topic label="Wheelbarrow" href="concepts/wheelbarrow.html"/>
<topic label="Workbench" href="concepts/workbench.html"/>
<topic label="Windshield washer fluid" href="concepts/wwfluid.html"/>
</topic>
</toc>
```

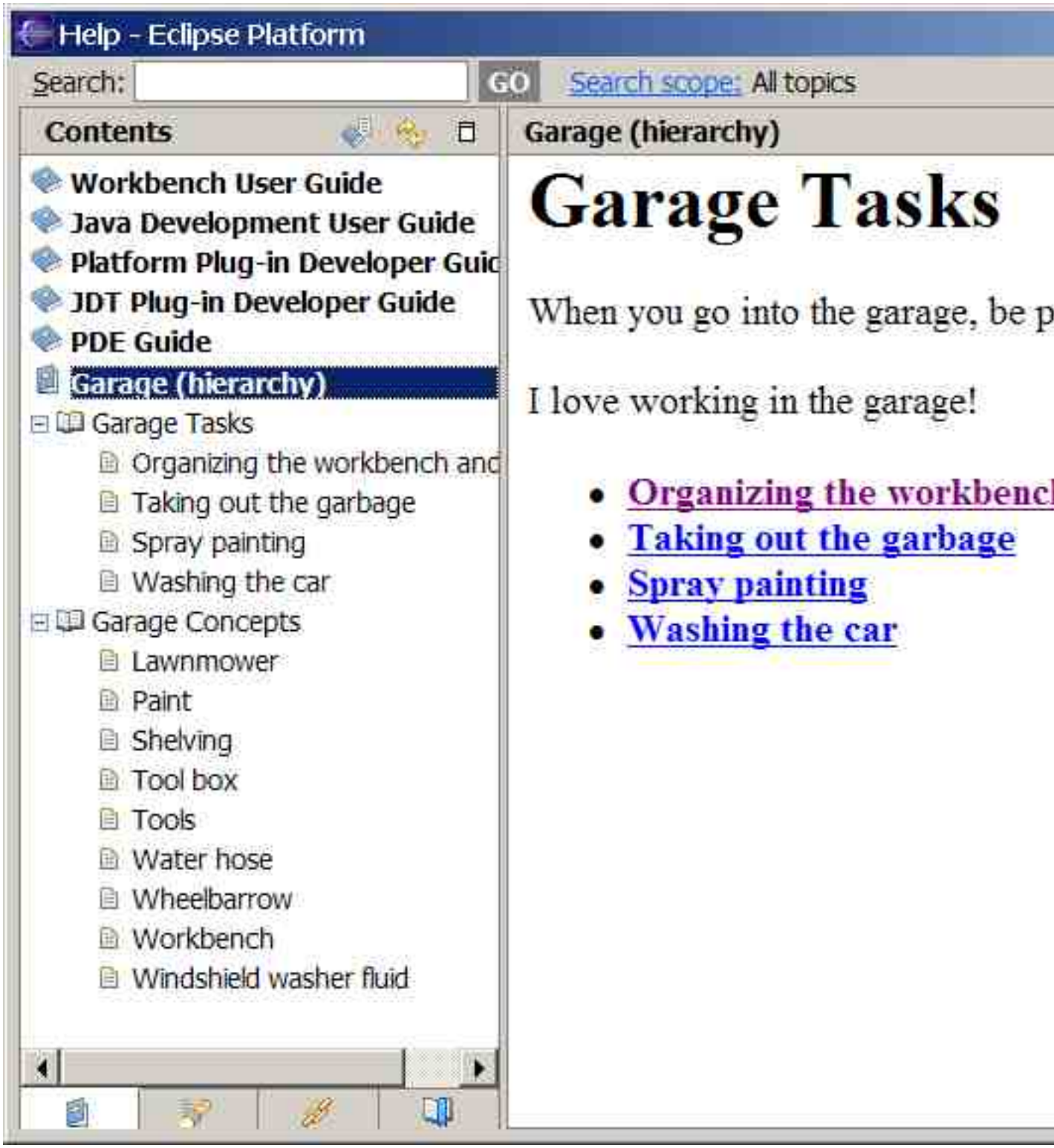
Building DITA content in Eclipse

It is possible to use Eclipse as your IDE for developing, building, and testing your Eclipse plug-in projects. Refer to the DITA Open Toolkit User Guide topic entitled "Running DITA builds in Eclipse."

Output

Detailed instructions exist in the DITA-OT help for producing Eclipse help. Refer to DITA Open Toolkit User Guide topic entitled "Processing to Eclipse help targets."

The image shows the content after it has been transformed and the plug-in has been added to a basic Eclipse runtime environment.



If you intend to take advantage of active help or the dynamic content capabilities within the Eclipse help system, such as link filtering, be prepared to create your own post-processing transforms to incorporate the required markup.

Summary

Whether creating documentation for an Eclipse-based application or one that simply uses the Eclipse help system to deliver user assistance, consider DITA to source content. Since Eclipse help is one of the key output targets for the DITA Open Toolkit, setup, configuration, and processing are all straightforward and well-documented. For additional instructions on such advanced Eclipse help capabilities as active help, dynamic content, or remote help, refer to the current Eclipse documentation, located at *http://www.eclipse.org/documentation/*.

Jeff Antley

OASIS DITA Help Subcommittee

LMI AIR Help Plugin

This topic provides information about generating AIR Help from DITA.

The DITA-OT AIR Help plugin is not yet publicly available.

Overview

AIR Help is an evolving new option for delivering online user assistance. This Help delivery format does not specifically define appearance or functionality, but refers to any user assistance application that was developed with Adobe's AIR (Adobe Integrated Runtime) technology. Applications developed using AIR can be installed on the Mac, Windows, and Linux operating systems, and are highly customizable, making this an ideal technology from which to develop an online Help deliverable.

Currently the only commercially available product for generating AIR Help is Adobe's "RoboHelp Packager for AIR." This (beta) product allows you to export an AIR Help file from a RoboHelp WebHelp project. At this time, there is no connection between RoboHelp and DITA other than the option of authoring DITA files in FrameMaker and importing the FrameMaker files into RoboHelp.

The other option for creating AIR Help is custom development. You can take any browser-based set of HTML files and "wrap" them up in an AIR application. The most basic AIR application is one that contains an embedded web browser (based on WebKit) which essentially lets you present your HTML-based help in your custom browser.

The LMI AIR Help plugin provides a direct path from DITA to AIR Help. This implementation of AIR Help provides the following features:

- Tabbed navigation panels for Contents, Index, and Search
- Full text search that allows for both local and remote search indexes
- Forward, Back, Home buttons as well as Next/Previous Browse buttons that follow the structure of topics in the Contents tab
- The ability to make context sensitive calls from an external application
- Position and size of window is preserved between sessions
- Sync with TOC functionality

Additionally, you may customize the Help interface to add features or modify the appearance as needed.

Setup and configuration

<<WHERE TO GET THE PLUGIN .. SETUP REQUIREMENTS>>

For the most basic output, no extra effort is needed. Run your DITA map through the plugin, and it will ..

- generate the "local" full text search index
- compile the SWF
- build batch files for testing and final AIR file creation

The following data is used from the map file when building the AIR file:

- /map/topicmeta/copyright
- /map/topicmeta/prodinfo/prodname
- /map/topicmeta/prodinfo/vrmlist/vrm (the last one)
- /map/topicmeta/prodinfo/prognum
- /map/topicmeta/othermeta/@name='remote-search-index-url' (@content specifies the location for the remote search index which is loaded on startup)

The Index is built from available indexterm elements.

After a successful build, just run the `airhelp-test.bat` batch file to view and test the new AIR Help file. When you're ready to build the final AIR file, just run the `airhelp-packager.bat` batch file.

Authoring

<<AUTHORING NEEDS .. ??>>

Integration

<<HOOKING UP FOR CONTEXT HELP>>

Output

<<SAMPLES>>

Summary

<<MORE INFO>>

*PTC Help [TBS]

PTC Help

PTC Help

Developing Custom DITA-based Help Systems

This topic introduces solutions for moving your DITA source content into one of several existing help run-time environments.

If your company has developed a custom Help viewer or run-time environment, it is unlikely that the default output from an existing DITA Help plug-in will be what you need. This section of the *Best Practices Guide* explains how to DITA plug-ins to generate the base ingredients for a custom help implementation. You will need to invest some amount of post-processing effort, therefore, to integrate output from multiple DITA plug-ins and/or to customize output for exactly what you need.

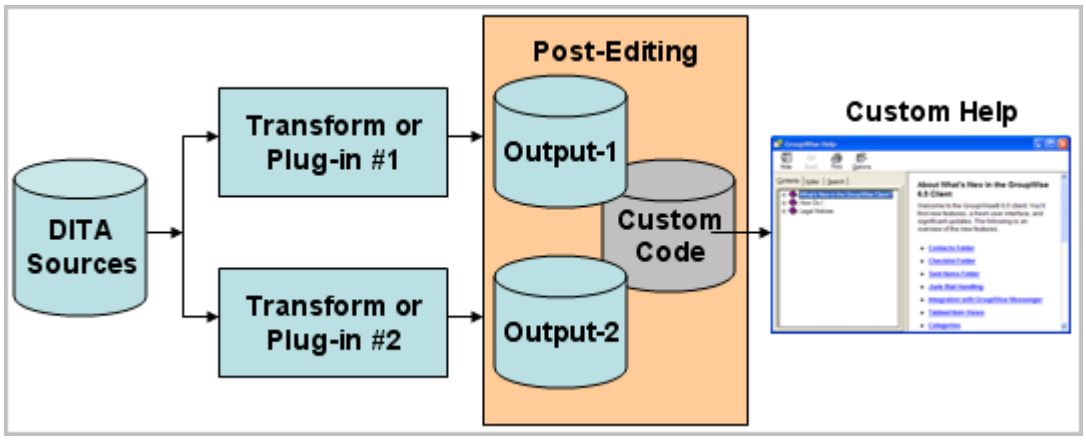


Figure 2: DITA Source to Customized Help Environments

We can't really help you with these post-processing or customization tasks, but we can help you generate the ingredient content and point out some common issues that might affect your customizations.

Plug-in/Tool	Output ...	Useful for ...
CSHelp Plug-in		
Dynamically rendered HTML		
HTMLSearch Plug-in		
TOCJS and TOCJSBIS Plug-in		
VistaHelp-style Expandable Content		
WinAnt Options and Outputs		
XHTML DITA-OT Transform		

DHTML Effects in HTML Generated from DITA

This topic describes an approach to creating expanding text and other DHTML effects in HTML-based output generated from DITA content.

It is common for Help systems to use *layering* techniques to limit the amount of information presented to the reader. The reader chooses to view the information by clicking on a link. Most layering techniques, including expanding text, dropdown text and popup text, are implemented using Dynamic HTML.

Overview

The DITA Open Toolkit HTML transformations do not provide for layering effects. However, some changes to the XSL-T files, and the use of outputclassmetadata in the DITA topic content, along with some judicious use of JavaScript and CSS, can deliver these layering effects.

Authoring Example

In the following example illustrating the technique, a note element is to output as dropdown text, where the note label is used to toggle the display of the note text. The note element is simply marked up with an outputclass distinct attribute value (in this case, `hw_expansion`).

```
< note outputclass="hw_expansion" type="note">Text of the note</note>
```

Without any modification, the DITA OT will transform the note element to a paragraph element with a CSS class of the outputclass value.

XSL-T Changes Example

In the example illustrating the technique, the following changes to the way the XSL-T file processes the note element are made:

```
<xsl:template match="*[contains(@class, 'topic/note ')]">
<xsl:if test="@outputclass='hw_expansion'" >
  <p>
   <a class="pseudolink_up" onclick="hwToggle(this);">Note: </a>
  </p>
</xsl:if>
<div >
<xsl:if test="@outputclass='hw_expansion'" >
 <xsl:attribute name="class">
   <xsl:value-of select="@outputclass" />
 </xsl:attribute>
</xsl:if>
</div>
</xsl:template>
```

JavaScript and CSS Addition Examples

For the layering DHTML effect to work, a JavaScript routine must be written to control the *toggling* of the display of the dropdown text. Likewise, CSS classes used in the layering (in this example, pseudolink_up and hw_expansion) must be defined in the CSS file for the output.

A simple JavaScript toggling routine is as follows:

```
function hwToggle(me) {
 var nextS=getNextSibling(me.parentNode);
 if(nextS.style.display!='block') {
  nextS.style.display='block';
  me.className='pseudolink_down'
 }
 else {
  nextS.style.display='none';
  me.className='pseudolink_up'
 }
}

function getNextSibling(startBrother){
  endBrother=startBrother.nextSibling;
 try {
  endBrother.nextSibling;
 }
 catch(er) {
  return null;
 }
```

```
  while(endBrother.nodeType!=1){
    if(!endBrother.nextSibling){
        //alert('Sorry! No more siblings!');
        return null;
        }
    endBrother = endBrother.nextSibling;
  }
  return endBrother;
}
```

A simple set of CSS settings are:

```
a.pseudolink_up
{
    color: blue;
    cursor: hand;
    border-bottom: blue 1px dashed;
    background-position: left center;
    background-image: url(images/purple_right_sm.jpg);
    background-repeat: no-repeat;
    padding-left: 15px;
}
a.pseudolink_down
{
    color: green;
    cursor: hand;
    border-bottom: blue 1px dashed;
    background-position: left center;
    background-image: url(images/purple_down_sm.jpg);
    background-repeat: no-repeat;
    padding-left: 15px;
}

div.hw_expansion
{
    display:none;
}
```

When the build file is created, references should be made to include the CSS file (containing the CSS rules) and the JavaScript file (containing the toggle routine).

Working Example

The layering technique described here is used in a production environment on a Web site which dynamically renders DITA content as XHTML. (Refer to *Web page using DHTML layering technique*.)

Summary

Although some technical knowledge is required to implement DHTML effects in output, the techniques are not onerously complex. With a small investment in effort, a big payoff in functionality can be achieved.

Dynamic Rendering of DITA into XHTML

This topic describes the concept of dynamic transformation of DITA content on a Web server into XHTML for delivery to any Web browser.

In many cases, Help content is best delivered as HTML. For content authored in DITA, the process is typically to transform the content into HTML (or, more likely, XHTML), and then transfer the output files to a Web browser. In some cases, it may be desirable to host the DITA source files on the Web server, and use server-side processing to dynamically transform the content to HTML when a browser requests the information.

Overview

When DITA content is transformed to a delivery format through the DITA Open Toolkit, or another publishing process, the result is a collection of deliverable files that also need to be managed. The deliverable files (typically XHTML) must be copied to a Web server for delivery. When the source content changes, the output has to be re-generated, and then copied once more to the Web server.

Most Web servers support some sort of server-side processing technology; examples are ASP.Net, PHP and JSP. Many of these technologies are XML-aware, and are capable of handling dynamic XSL-T transformations.

It is possible to set up a system whereby Help content, written and stored in DITA format, is stored on a server, and then dynamically transformed into HTML for delivery.

Setup and configuration

To set up a dynamic transformation system, the services of a developer or someone familiar with a server-side processing and XSL-T is required. The developer will need to devise a menu system (perhaps derived from a ditamap file), and a topic rendering system.

Authoring

Once the system is in place, DITA topics can be either authored directly onto the Web server (perhaps through FTP, WebDAV or a network share).

Limitations

While it is relatively easy to set up a simple menu system and a simple topic rendering system, it is not easy to handle more advanced DITA features such as conrefs, reltables and composite topics. For example, conref elements need to be resolved prior to the XSL-T transformation. In the DITA Open Toolkit, the generation of output comprises a number of stages to first resolve references to build an interim document, and then process the interim document into the output.

However, If the DITA content is simple enough, this limitation is not an impediment.

Example

The *HyperWrite Web site* uses these techniques through ASP.Net to generate the breadcrumb trails, navigation system, menu pages, and articles topics from a ditamap file and numerous DITA concept topics. (Some content topics are also hosted in DocBook format.)

Summary

The primary advantage of the dynamic transformation approach is to minimise file management. The main disadvantage is that the technique is practically limited to simple content structures. Some other dedicated Web delivery methods, such as Eclipse Help, Mark Logic Content Server, or other CMS solutions, may provide greater support for complex content structures.

*HTML-Plus [TBS]

HTML Plus

HTML Plus

HTML Plus

HTMLSearch Plug-in

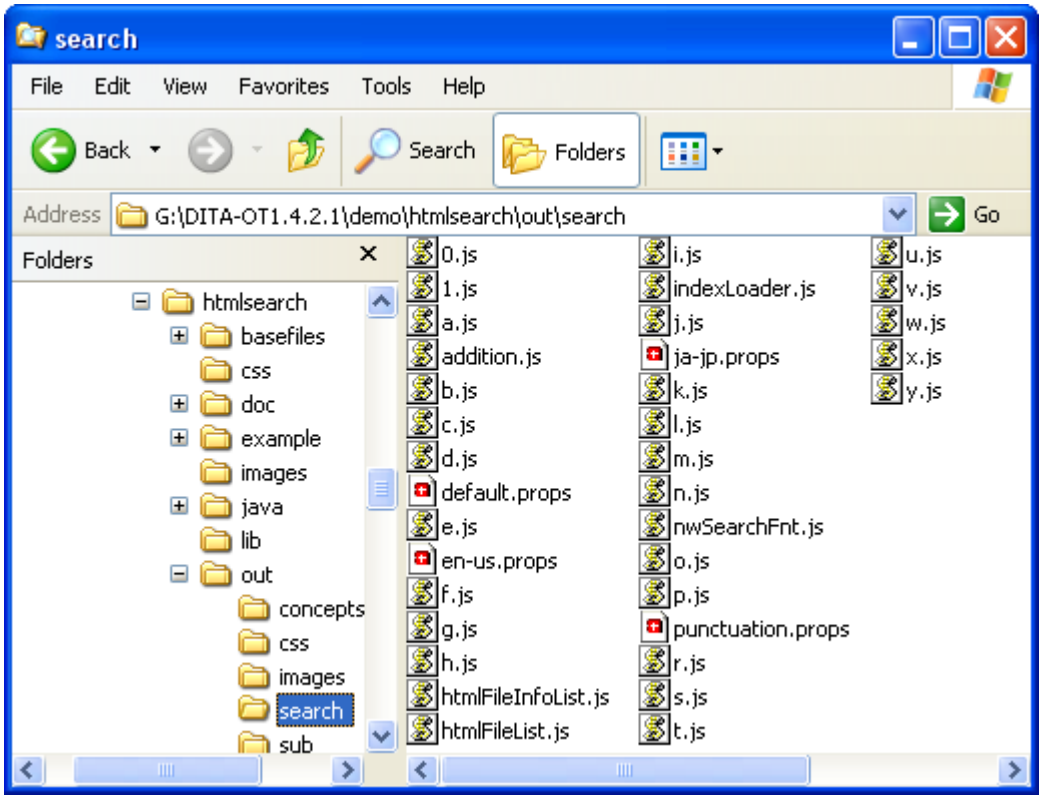
This topic provides an introduction to the DITA plug-in named HTMLSearch.

The HTMLSearch plug-in for the DITA Open Toolkit builds a keyword-based search index and a default frame-based user interface for any collection of DITA topics referenced by a DITA map file.

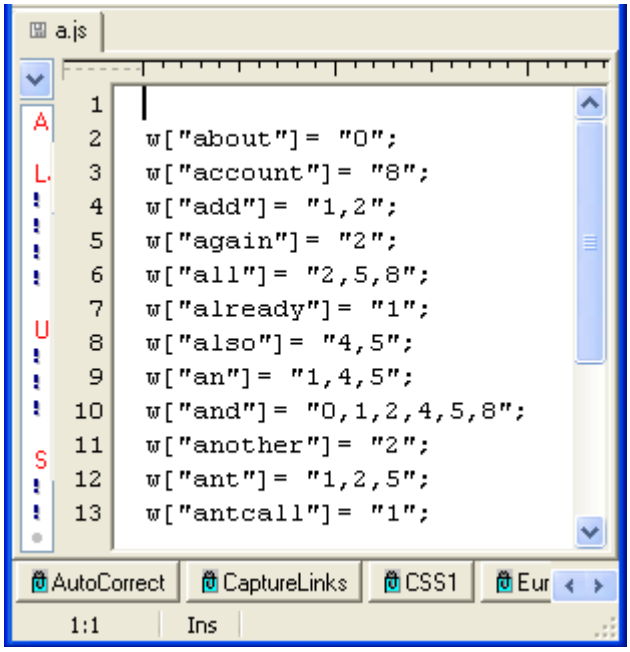
Overview

Nadege Quaine (nquaine@hotmail.com), the contributor of the tocjsbis plug-in, has also contributed this plug-in. As of this writing, the most current version of the plug-in is dated April 8, 2008 and is distributed as `htmlsearch1.04.zip`.

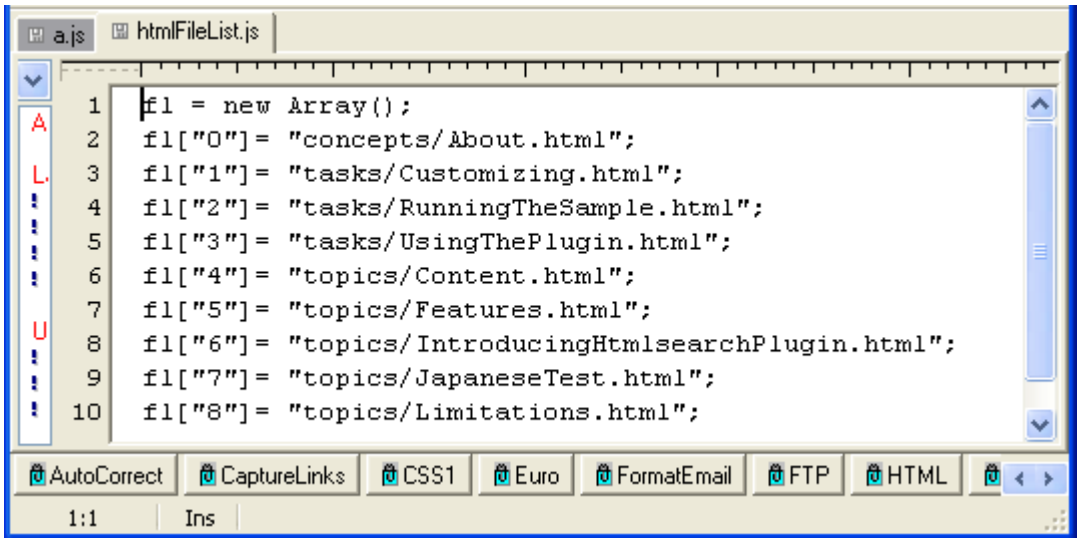
The HTMLSearch plug-in first runs the XHTML transform that comes with the DITA-OT and then runs an indexing application against those XHTML output files. HTMLSearch deposits the keyword index and the supporting JavaScript worker libraries in a subdirectory named `search`.



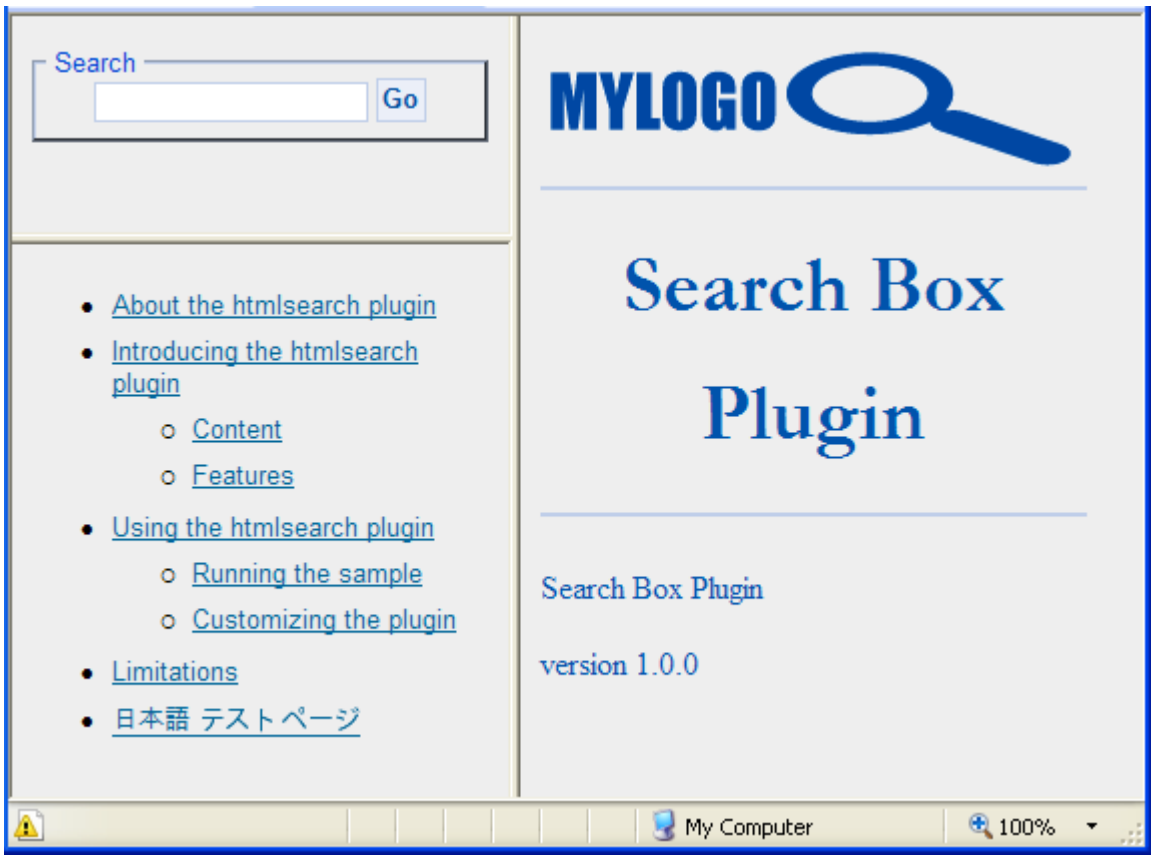
Each of these JavaScript index files correlates discovered keywords to one or more numerals representing particular XHTML output topics.



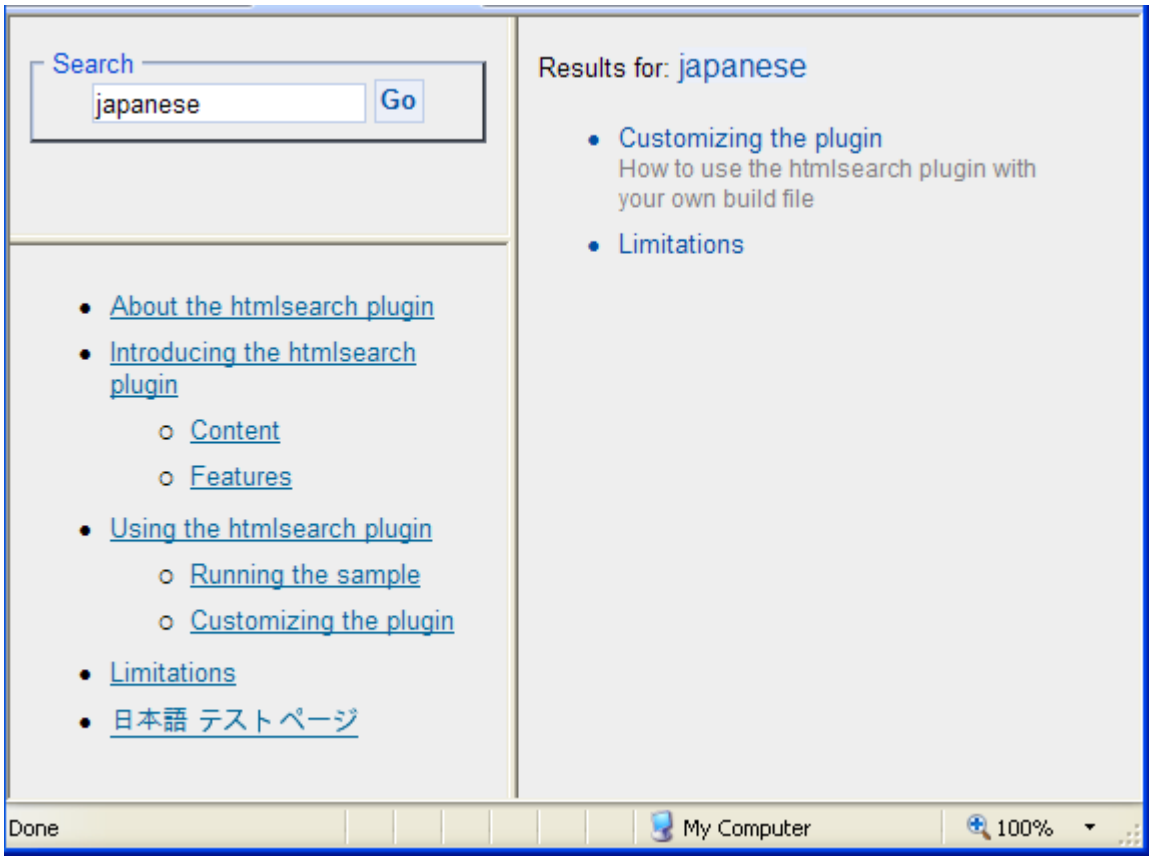
The keyword "ant," for example, appears in three XHTML topics. These target topics are referenced numerically (1, 2, 5) and indexed separately in the file `search\htmlFileList.js`.



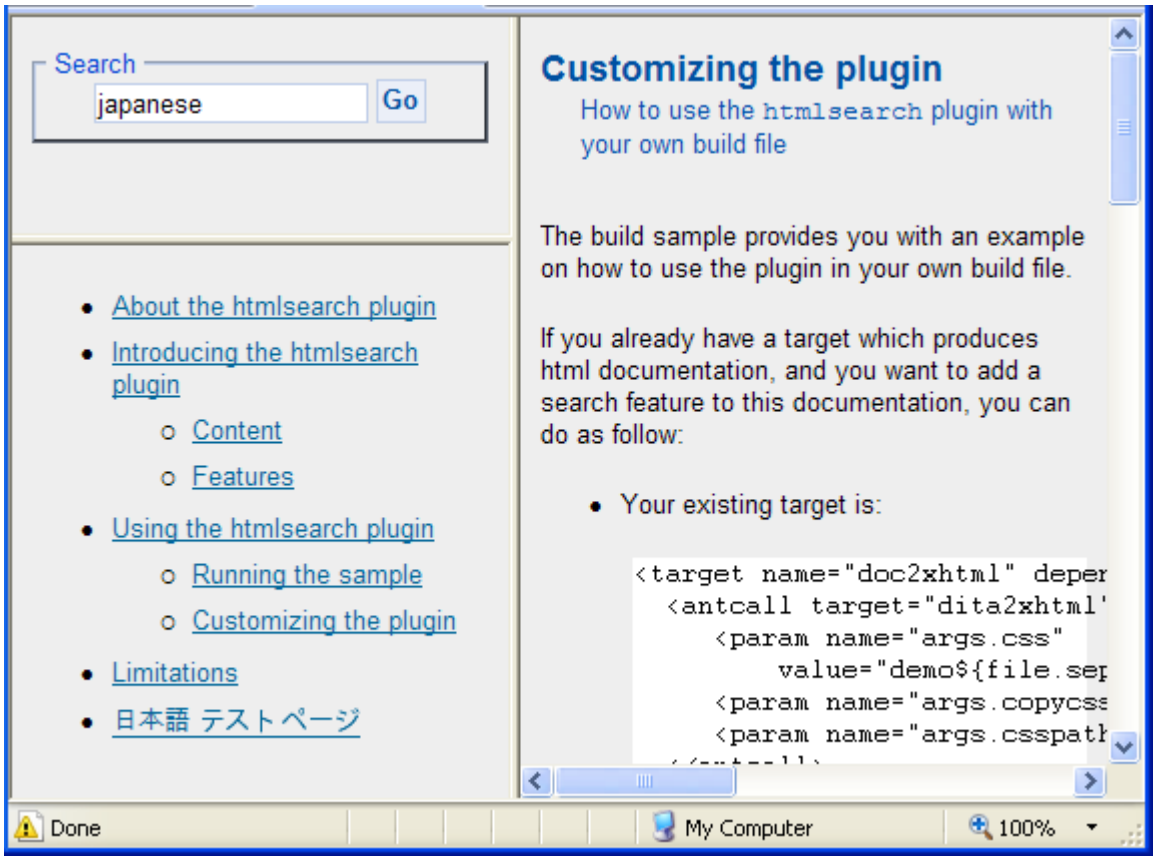
When you open the default HTMLSearch web page (`frameset.html`) in the output directory, it opens three frames containing a keyword input box, some default information about the plug-in, and a splash screen.



Results from a keyword search are displayed in the right-hand pane.



Clicking a hyperlink in the results list displays the target topic containing the keyword in the same right-hand pane.



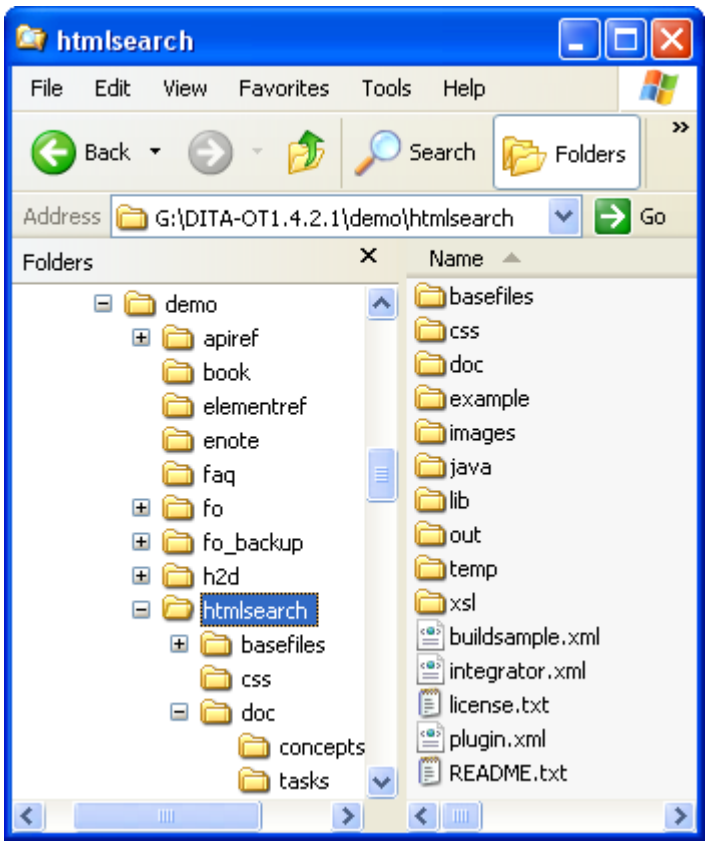
The plug-in works fine in all versions of the DITA-OT because it is really interacting with XHTML output from the DITA-OT, not Open Toolkit resources *per se*. Beyond minor variations in the way the HTML displays in different browsers, HTMLSearch output works reliably in all modern browsers.

Setup and configuration

The HTMLSearch plug-in is a free download from the Yahoo DITA-OT site.

http://tech.groups.yahoo.com/group/dita-users/files/Demos/

After you have unzipped this archive to a local directory, you can browse the `README.txt` file to get a feel for what HTMLSearch offers and how you can install it in your DITA-OT `demo` directory.



Installing the plug-in presents no surprises.

1. Copy the un-archived `htmlsearch` subdirectory into the `demo` subdirectory of your DITA Open Toolkit directory.
2. Open a shell command window from your DITA-OT directory.
3. Enter the following command to integrate the new plugin or plugins with your current DITA-OT environment.

   ```
   ant -f integrator.xml
   ```

4. From the DITA-OT root directory, enter the following command to build the sample DITA topics.

   ```
   ant -f demo/htmlsearch/buildsample.xml xhtml2search
   ```

5. Load the newly generated `demo\htmlsearch\out\frameset.html` file in your browser.

That is about all that is involved with installing and configuring the plug-in.

Authoring

There are no DITA source-level authoring considerations for this plug-in; it works on XHTML output topics generated by the DITA-OT XHTML transform.

Integration

Integrating output from the HTMLSearch plug-in with tocjsbis output or with your local favorite HTML help implementation is possible, but challenging. If you have some experience modifying (or hacking) HTML frames and JavaScript code, have at it. Otherwise, you might consider continuing to use HTMLSearch output as a stand-alone complement to your other output transformations.

Consider the following tips when planning your customization effort.

- *Quotation marks in ditamap navtitles*: If the navtitles in your `.ditamap` file(s) contain quotation marks, remove them before running the HTMLSearch plug-in. HTMLSearch passes these quotation marks through to its indexed output, effectively breaking JavaScript syntax in the `search\htmlFileInfoList.js` file.

- *Directory levels for XHTML output*: The HTMLSearch plug-in writes the hyperlinked results of a keyword search to a frame named `contentwin`. The initial static HTML loaded into that frame lives in the output subdirectory named `sub`. The filepaths in the hyperlinks written to that frame, therefore, are relative to an HTML file in `sub`. If you are using default XHTML output from the tocjs or tocjsbis, you may need to edit the JavaScript code that builds that relative path. Test removing the three characters "`../`" in line 131 of `search\nwSearchFnt.js` if you need to adjust HTMLSearch hyperlink paths to match target directory paths from tocjsbis or other DITA-OT XHTML output transformations.

  ```
  linkString = "<li><a href=\"../"+tempPath+"\">"+tempTitle+"</a>";
  ```

- *Target frame name for keyword search input box*: The HTMLSearch plug-in loads its input box into a frame named "searchwin" in its default interface. To integrate that input box into a different set of named HTML frames, you will need to change the name of the input box frame from "searchwin" to the new target frame name in lines 32, 39, 47, 242, and 248 in `search\nwSearchFnt.js`. For example, change "searchwin" to "*your_frame_name*" in the following line in `search\nwSearchFnt.js`:

  ```
  expressionInput=parent.frames['searchwin'].document.ditaSearch_Form.textToSearch.value
  ```

- *Target frame name for keyword search results*: The HTMLSearch plug-in writes its list of keyword search results (hits) to a frame named "contentwin" in its default interface. To have those results displayed in a different frame in your implementation, you will need to change the name of the input box frame from "contentwin" to the new target frame name in lines 146 and 414 in `search\nwSearchFnt.js`. For example, change "contentwin" to "*your_frame_name*" in the following line:

  ```
  with (parent.frames['contentwin'].document) {
  ```

- *Target frame name for target topics*: The HTMLSearch plug-in displays target topics in the same "contentwin" frame as the search results. If you would like to have *both* the list of search results *and* displayed target topics displayed simultaneously in separate frames, you will need to change that way that HTMLSearch builds hyperlinks. Specifically, you'll need to add an HTML "target=*framename*" attribute to line 131 in `search\nwSearchFnt.js`. For example, insert the string `target='topicwin'` (or *your_frame_name*) into line 131. Here's the original ...

  ```
  linkString = "<li><a href=\"../"+tempPath+"\">"+tempTitle+"</a>";
  ```

  and the update ...

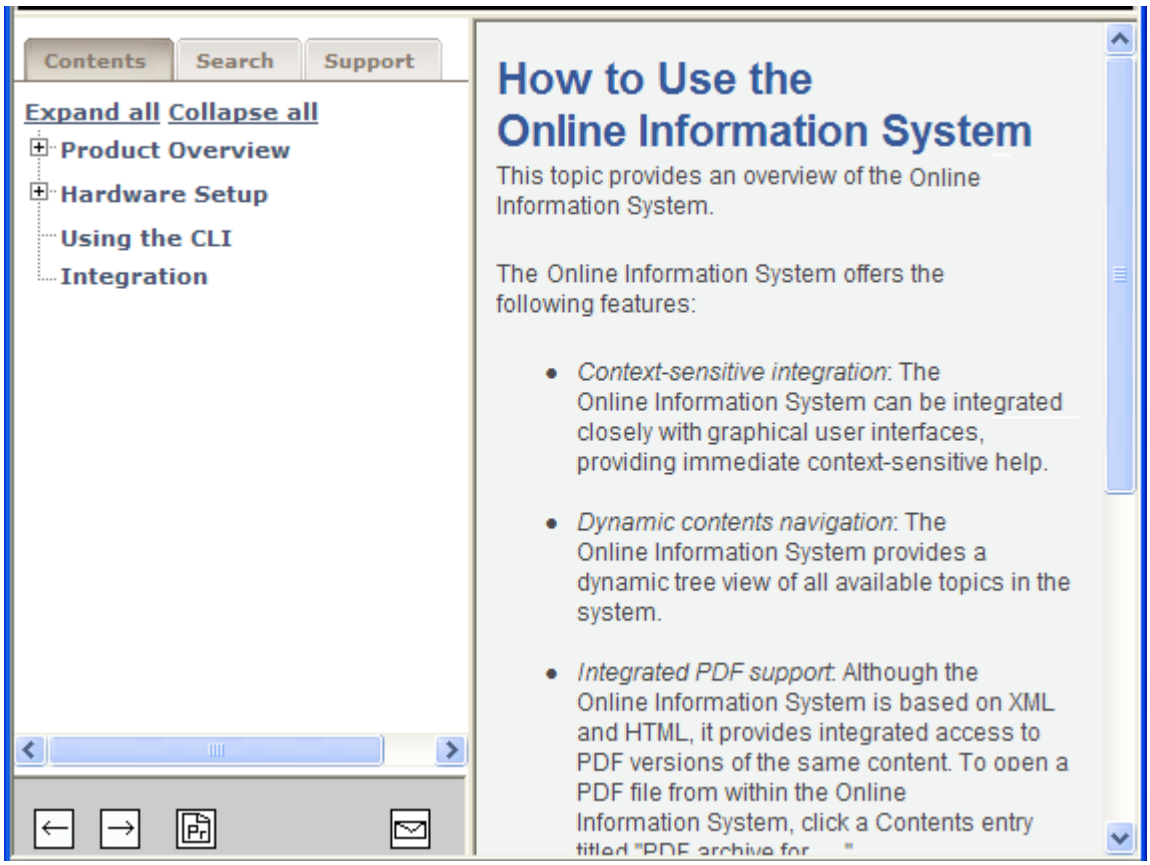
  ```
  linkString = "<li><a href=\"../"+tempPath+"\"
  target='topicwin'>"+tempTitle+"</a>";
  ```

  After this change, HTMLSearch builds hyperlinks in the keyword search results frame that specify a different target window for displaying target XHTML topics.

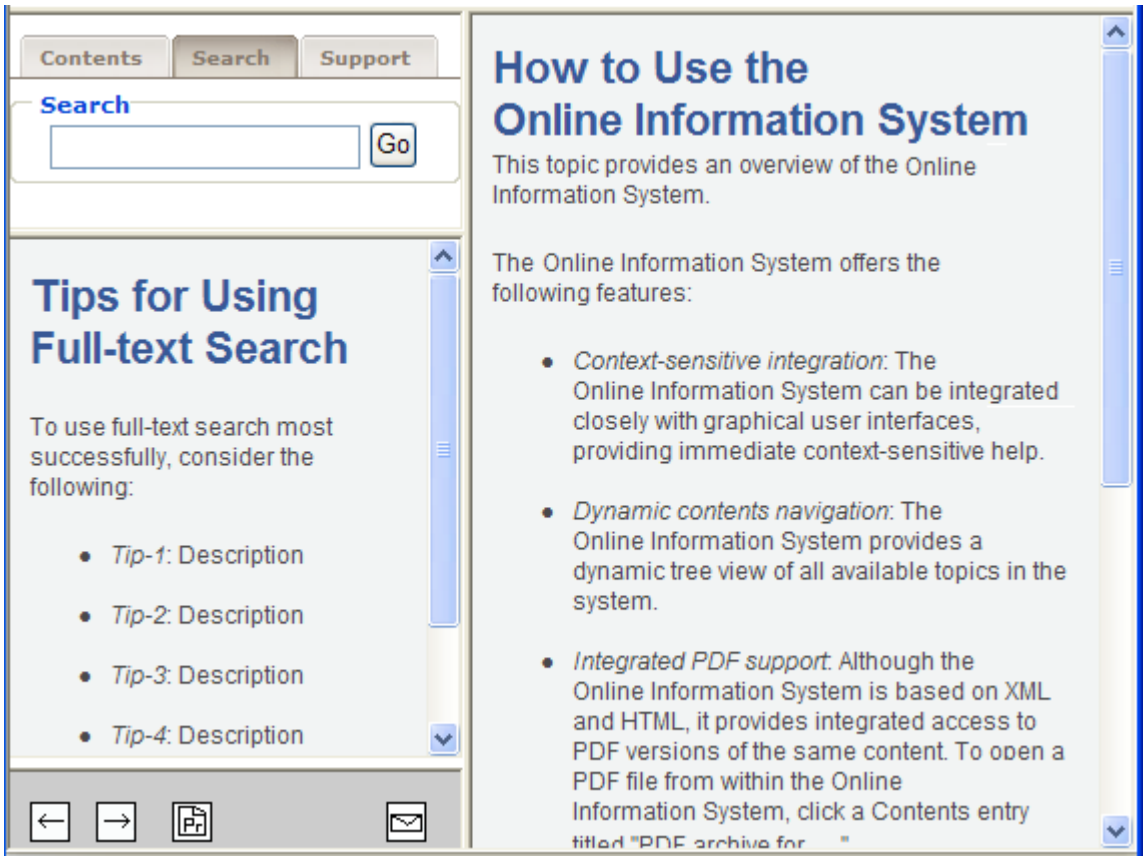
These are the most visible variables that you'll need to consider when customizing HTMLSearch output for your own help implementation.

Output

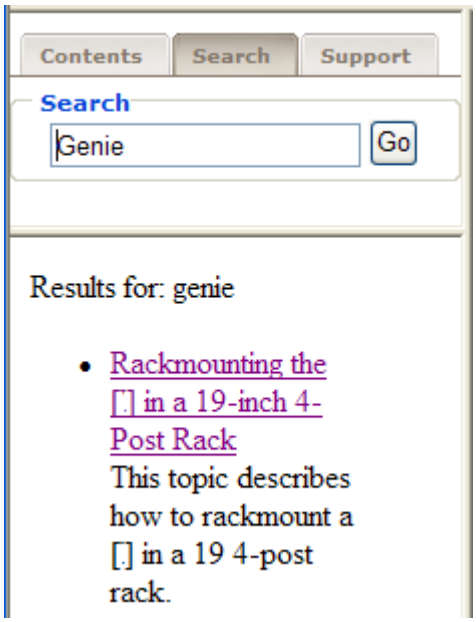
Although the customization process can sound scary, it does produce some very nice results. Here is a prototype Help implementation for my current company. The goal here is to integrate the output from HTMLSearch with output from tocjsbis in a garden-variety three-tab, multi-frame HTML shell. Initially, the search UI is linked to a Search tab.



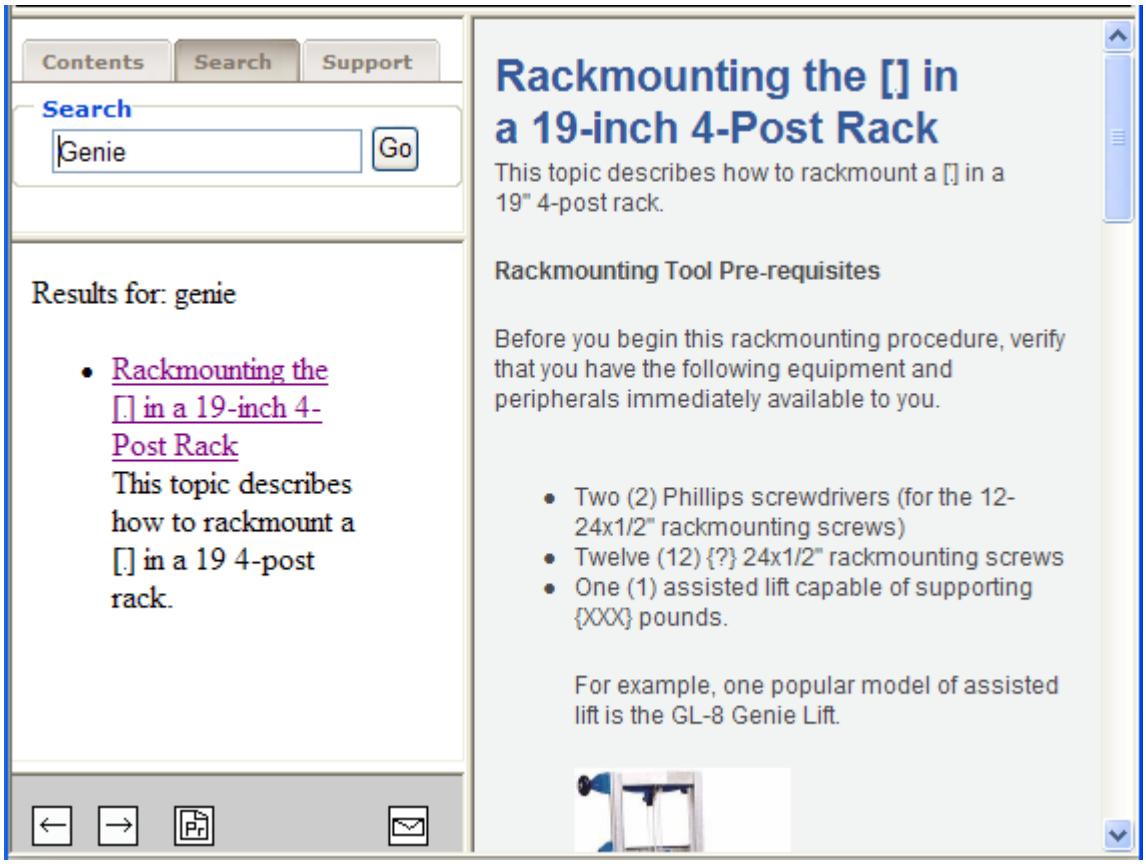
When the customer clicks the Search tab, a re-implementation of the HTMLSearch UI is displayed.



Search results appear in a lower frame in the left-hand navigation frame.



When the customer clicks one of the hyperlinks in the results list, that topic is displayed in the right-hand frame.



Summary

Customizing HTMLSearch to integrate it with tocjsbis or other XHTML output transformations from the Open Toolkit goes a long way toward making DITA-generated Help more usable and familiar to contemporary audiences.

Stan Doherty

OASIS DITA Help Subcommittee

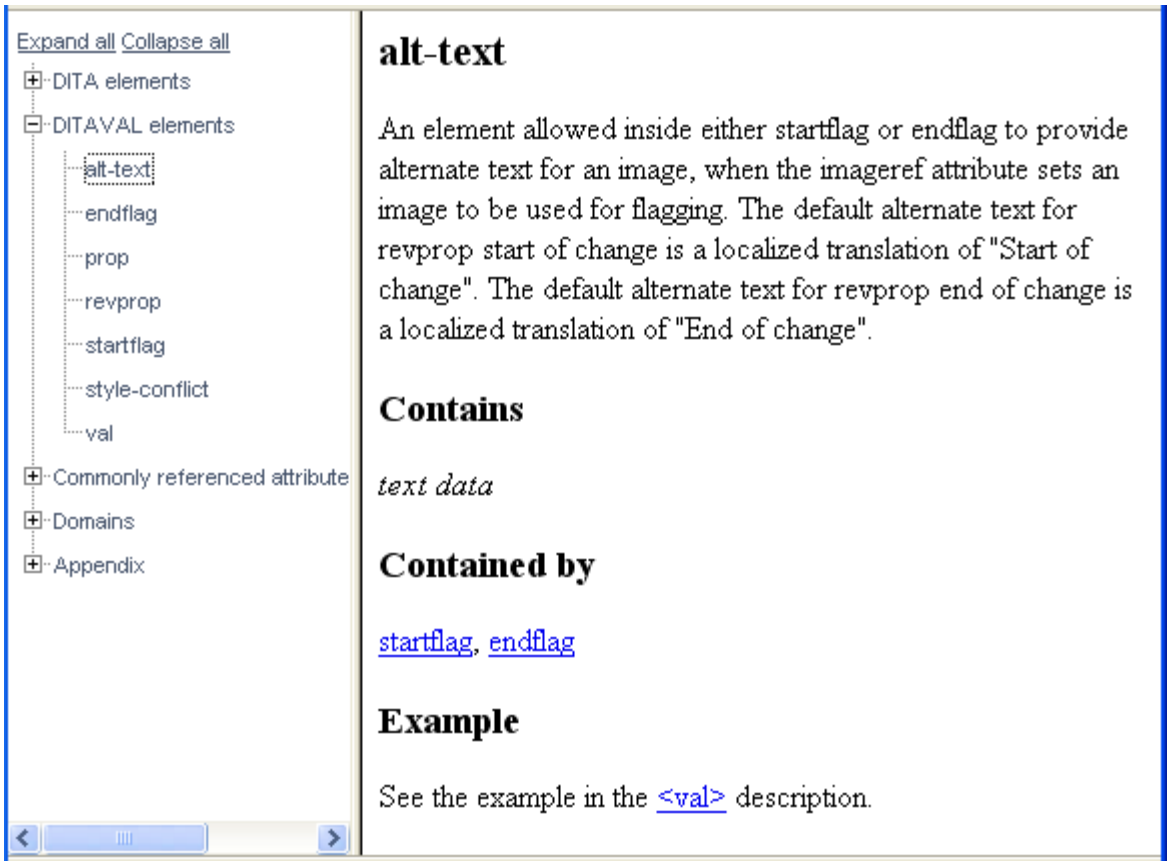
TOCJS and TOCJSBIS Plug-ins

This topic provides an overview of the DITA-OT plug-ins named tocjs and tocjsbis.

The tocjs DITA-OT plug-in and its more recent enhancement named tocjsbis generate a JavaScript-based table of contents page for any DITA topics that you reference in your `.ditamap` file.

Overview

These plug-ins are very popular in the DITA community; we even use them on the DITA Technical Committee for our DITA 1.2 specifications.



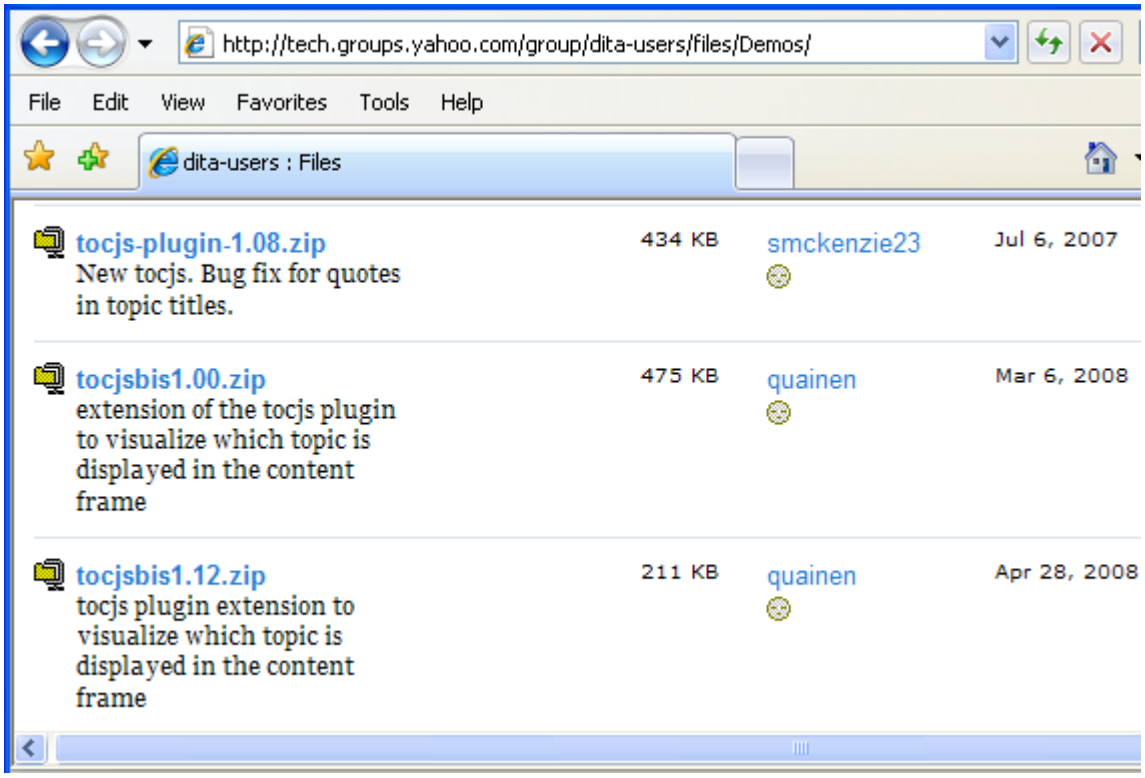
- *tocjs*: The tocjs plug-in was developed by Shawn McKenzie, currently working at Sophos in beautiful Vancouver, BC (Canada). The tocjs plug-in executes after the standard DITA-OT XHTML transform, so each tocjs TOC entry knows the name of its target XHTML topic.
- *tocjsbis*: The tocjsbis plug-in was written by Nadege Quaine and adds the important feature of topic synchronization, i.e. the highlighted topic entry in the TOC updates in sync with the topic being displayed in the contents frame. The plug-in achieves synchronization by adding a unique ID to each XHTML output topic (<meta content="id-tocjsbis_about" name="DC.Identifier" />) and by synchromizing the TOC entry against that topic ID. If you have generated HTML output from RoboHelp or WebWorks Publisher, this technique should be familiar.

If you are generating HTML output of any sort from your DITA sources, you should test one or both of these plug-ins. I use tocjsbis in my context-sensitive Help builds where I work. To the extent that the tree control in tocjs and tocjsbis are based on the Yahoo tree control library, you can customize the way that the final TOC tree displays and behaves in your Help system. They can be tricky, but customizations work.

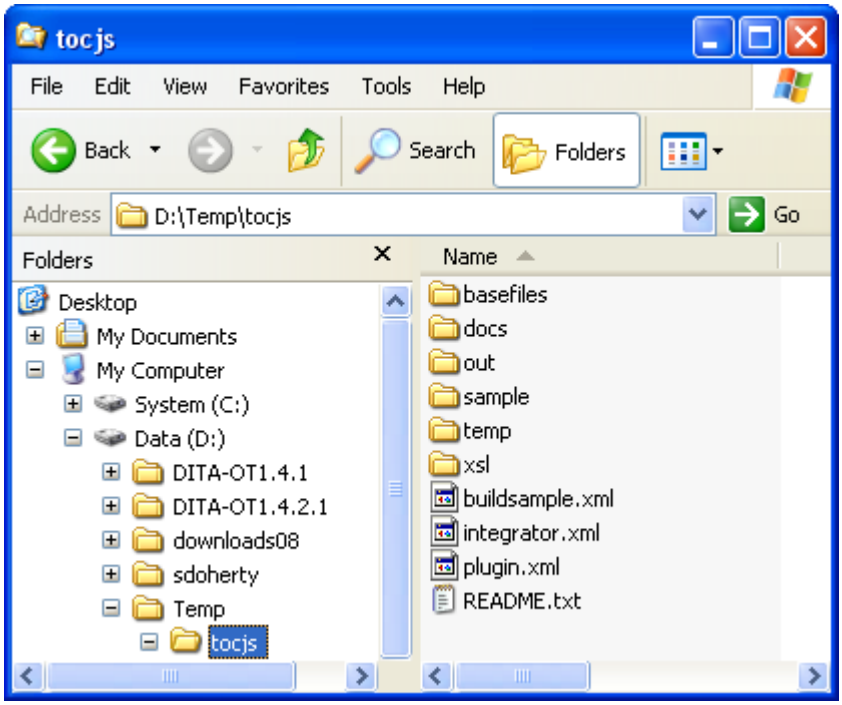
Setup and configuration

The tocjs and tocjsbis plug-ins are free downloads from the Yahoo DITA-OT site.

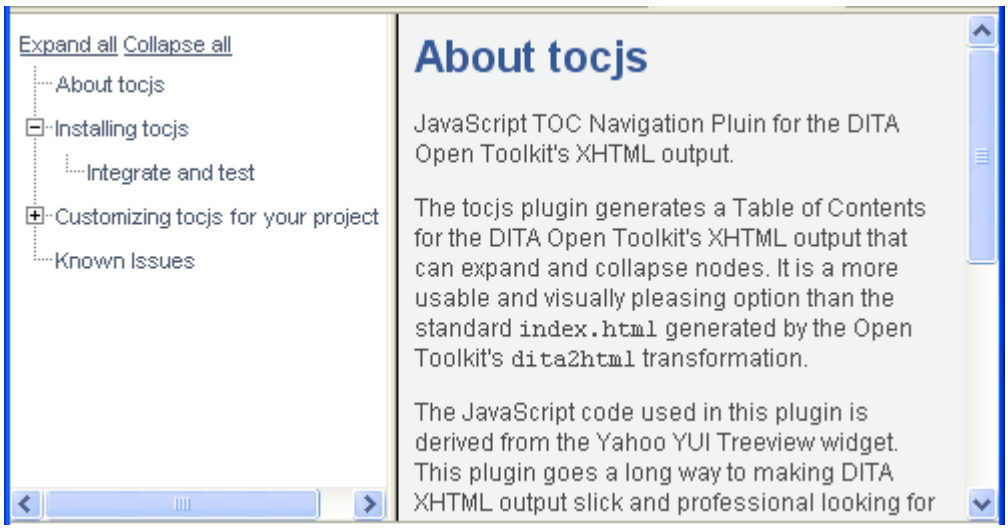
http://tech.groups.yahoo.com/group/dita-users/files/Demos/



After you have unzipped these archives to a local directory, you can browse the documentation to get a feel for what tocjs offers and how you can install it in your DITA-OT directory.

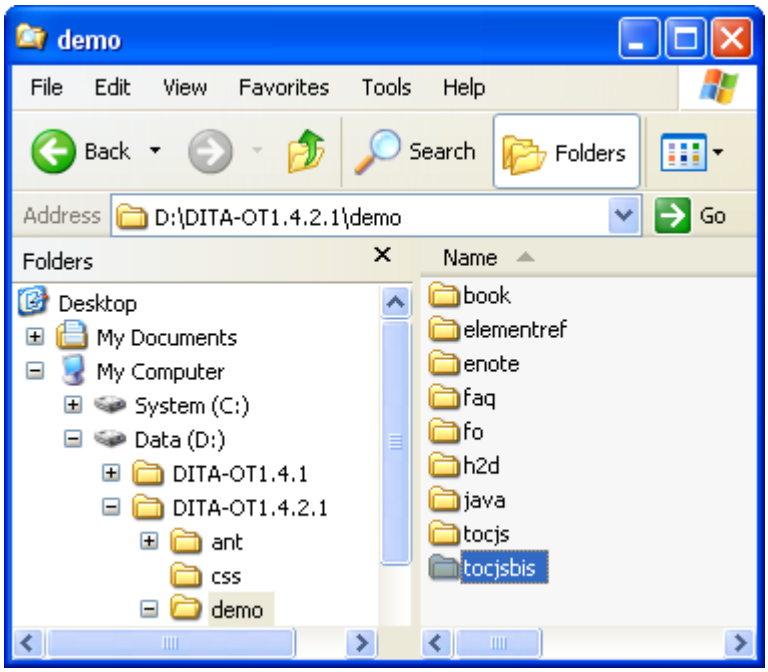


The pre-built documentation for tocjs lives in the `/docs` subdirectory.



Installing tocjs or tocjsbis is very straightforward.

1. Copy the un-achived tocjs or tocjsbis subdirectory into the demo subdirectory of your DITA Open Toolkit directory.



2. Open a shell command window from your DITA-OT directory.
3. Enter the following command to integrate the new plugin or plugins with your current DITA-OT environment.

```
ant -f integrator.xml
```

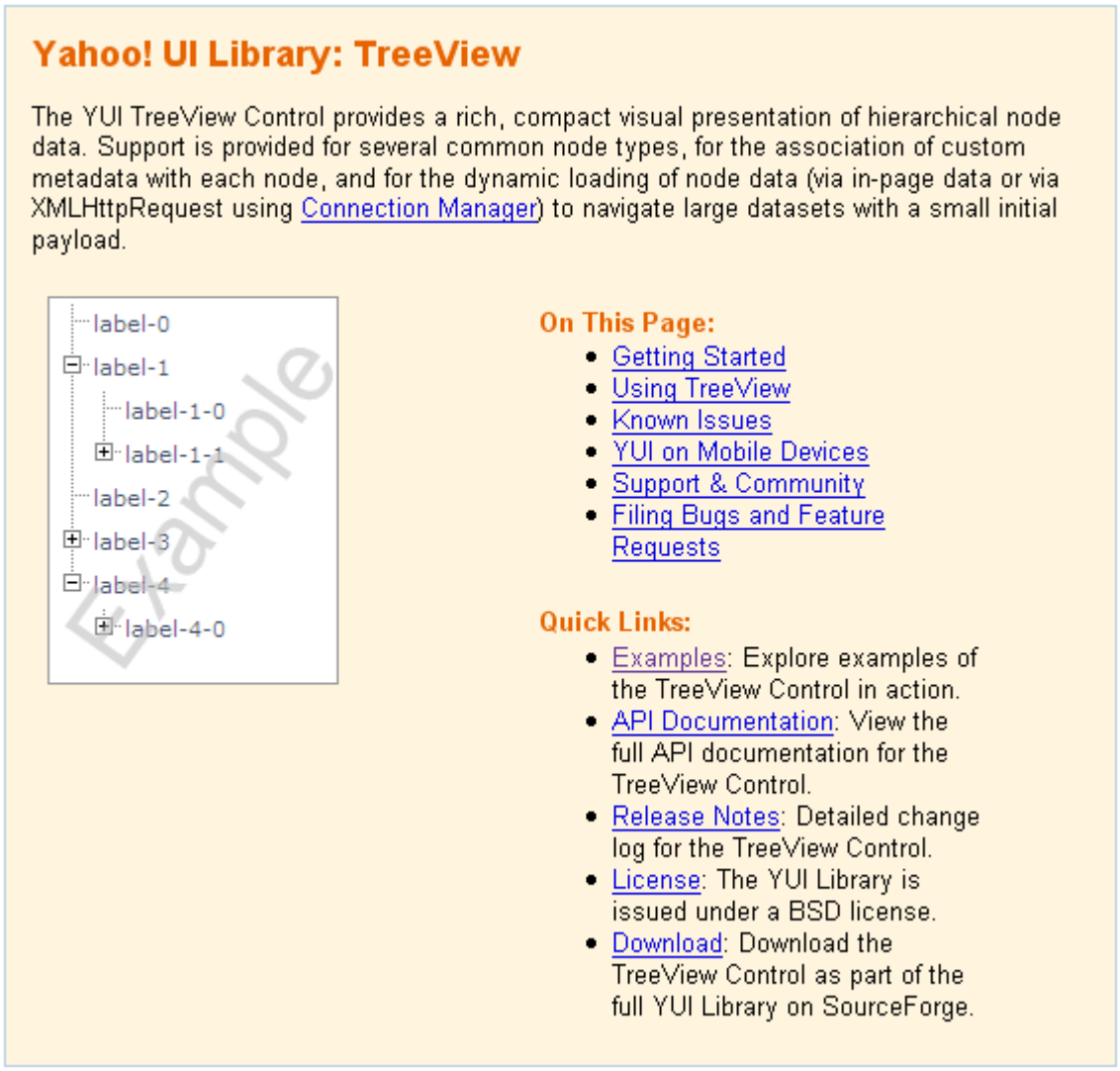
4. Change directory into the `demo/tocjs` or `demo/tocjsbis` subdirectory and enter the following command to build the sample DITA topics.

```
ant -f demo/tocjs/buildsample.xml sample2tocjs
```

5. Load the newly generated `demo\tocjs\out\sample\frameset.html` file in your browser.

If you are considering customizations to tocjs or tocjsbis, consult the documentation for the Yahoo UI tree control at the following URL.

http://developer.yahoo.com/yui/treeview/



That is about all that is involved with installing and configuring tocjs and tocjsbis.

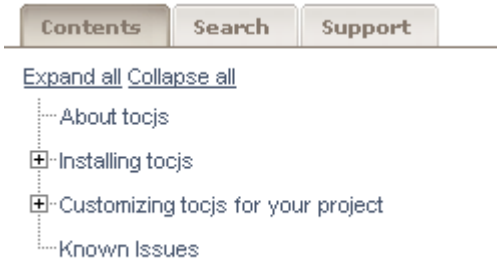
Authoring

These plug-ins piggy-back whatever investment you have already made in authoring your DITA topics and map files. Beyond setting up a new ant script for tocjs or tocjsbis, there is nothing additional required.

Integration

The tocjs and tocjs plug-ins present few integration problems or opportunites, especially as regards managing context sensitivity between Help output and the calling software application. Many DITA Help writers customize the `frameset.html` file that ships with the tocjs plug-in to personalize or brand the final product. Here's what Shawn McKenzie does with tocjs on his Sophos corporate website.

Wrapping other navigational devices around the tocjs output is not difficult. I add the conventional tabs for tri-pane Help systems.



In terms of translation, note that all the text strings associated with entries in the TOC tree are stored in one file named `toctree.js`. These files can be localized, for sure, but they are not very friendly. To make life easier on our sisters and brothers in L10N, you can post-edit this `toctree.js` file to swap JavaScript resource strings for literal strings.

Output

Here again is a link to a live demo of tocjs running at Sophos.

http://ca-repo1.sophos.com/docs/ws1000/

Summary

If you are comfortable customizing XHTML output to build an HTML-based help system, you should consider tocjs and tocjsbis. They are sufficiently lightweight to work in Help systems or web-based portals.

Stan Doherty

OASIS DITA Help Subcommittee

WinANT Options Supporting HTML-Based Output

This topic describes how some features within the WinANT software tool can make generation of HTML-based output from DITA content easier.

The DITA Open Toolkit provides a method for specifying a CSS style sheet, blocks of HTML code to add to the top and bottom of each generated HTML file, and a block code to add to the <head> section of each generated HTML file. However, the method is cumbersome to use from a command line or terminal window invocation of the Apache Ant build processor. The HyperWrite WinANT software tool, which acts as a Windows interface to Ant, makes this method extremely simple.

Overview

WinANT is a Windows program, build with Microsoft Visual Studio .NET 2003 using VB.NET. It serves as an interface to the Ant build utility, for the sole purpose of processing DITA documents.

WinANT allows a user to select build characteristics using normal Windows interface devices such as dropdown lists, radio buttons, tabs and browse buttons. When all the required settings are in place, the program creates the Ant build file, creates a ditaval file (if required), creates a batch file, and then executes the batch file to trigger the Ant build. When Ant has finished the processing, WinANT displays the generated output file. The settings can be saved (as a build file) and later recalled.

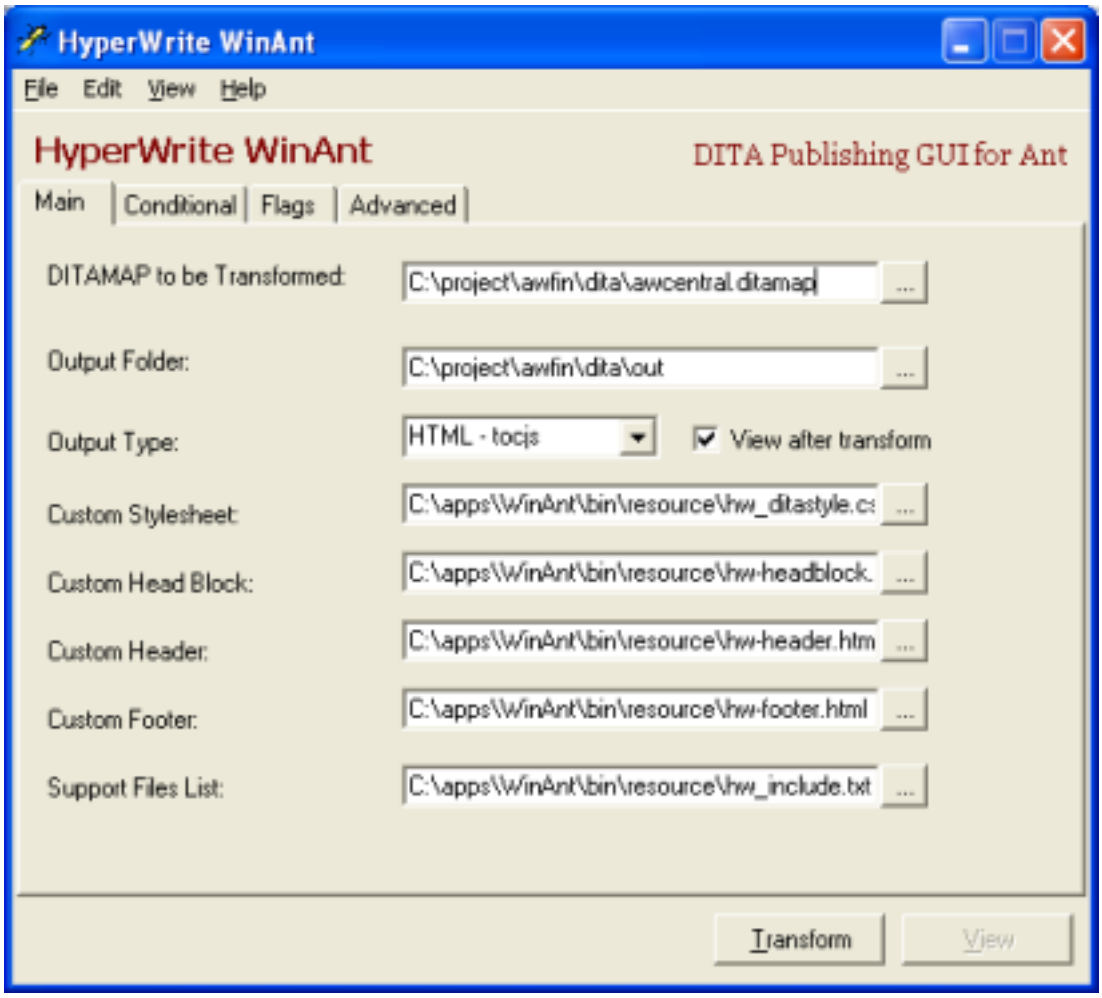


Figure 3: WinANT 1.4 - Main tab

Setup and configuration

WinANT can be downloaded without charge on a "take it as it comes" basis from *http://www.helpml.com/winant_setup.exe*. It installs using a standard Windows installer.

The Ant *arguments* that can be set within WinANT include:

- the folder where the output will be created;
- the type of output;
- a CSS stylesheet to be applied to each output HTML page;
- a file containing HTML code to be added to the <head> section of each output HTML page;
- a file containing HTML code to be added to the top of the <body> section of each output HTML page;
- a file containing HTML code to be added to the bottom of the <body> section of each output HTML page;
- a list of files to be copied to the output folder (or compiled into HTML Help output);
- conditional processing rules;
- images to be used for flagging conditional text;
- the DITA topic file extension used;
- the output HTML file extension to be used; and
- whether content marked as draft will be included in the output.

Authoring

The use of WinANT in the publishing stage does not alter the authoring method.

Publishing

When you are ready to produce HTML-based output from your DITA source, you can process your ditamap file through WinANT.

WinANT supports the following base DITA Open Toolkit and additional plug-ins:

- HTML
- (Microsoft® ®) HTML Help
- PDF
- PDF2
- Eclipse Help
- DocBook
- Word
- HTML with tocjs
- HTML with search

You will need to prepare the CSS stylesheet to use for presentation of the output, as well as any code blocks for the top (**Custom Header**), bottom (**Custom Footer**), and <head> section (**Custom Head Block**).

Note: Make sure your HTML code blocks are well-formed XML. If not, the block will not be included in the output HTML.

If your are producing HTML Help output, you can also nominate an *include file*, which is a simple list of additional files to be compiled into the resultant CHM, in plain text format. If your CSS file references graphics, these graphics files should be listed in the *include file* .

Selecting the CSS and code files

The CSS and HTML code block files are selected on the **Main** tab of WinANT. These fields are:

- Custom Stylesheet
- Custom Head Block
- Custom Header
- Custom Footer
- HTML Help Include File

only active if the **Output Type** field on the **Main** tab is set to *HTML Help*.

Summary

WinANT provides a simpler way of controlling the HTML-based output from DITA content than the standard DITA Open Toolkit command line. Its ability to store settings for future use also help make it a practical tool for DITA publishing.

WinANT Options Supporting Microsoft® HTML Help

This topic describes how some features within the WinANT software tool can make Microsoft® HTML Help generation from DITA content easier.

The DITA Open Toolkit provides a method for nominating context-sensitive Help *header* (or *map*) and *alias* files to be compiled into the CHM file when a Microsoft® HTML Help output is being generated. The method is difficult to use

from a command line or terminal window invocation of the Apache Ant build processor. The HyperWrite WinANT software tool, which acts as a Windows interface to Ant, makes this otherwise difficult method extremely simple.

Overview

WinANT is a Windows program, build with Microsoft Visual Studio .NET 2003 using VB.NET. It serves as an interface to the Ant build utility, for the sole purpose of processing DITA documents.

WinANT allows a user to select build characteristics using normal Windows interface devices such as dropdown lists, radio buttons, tabs and browse buttons. When all the required settings are in place, the program creates the Ant build file, creates a ditaval file (if required), creates a batch file, and then executes the batch file to trigger the Ant build. When Ant has finished the processing, WinANT displays the generated output file. The settings can be saved (as a build file) and later recalled.

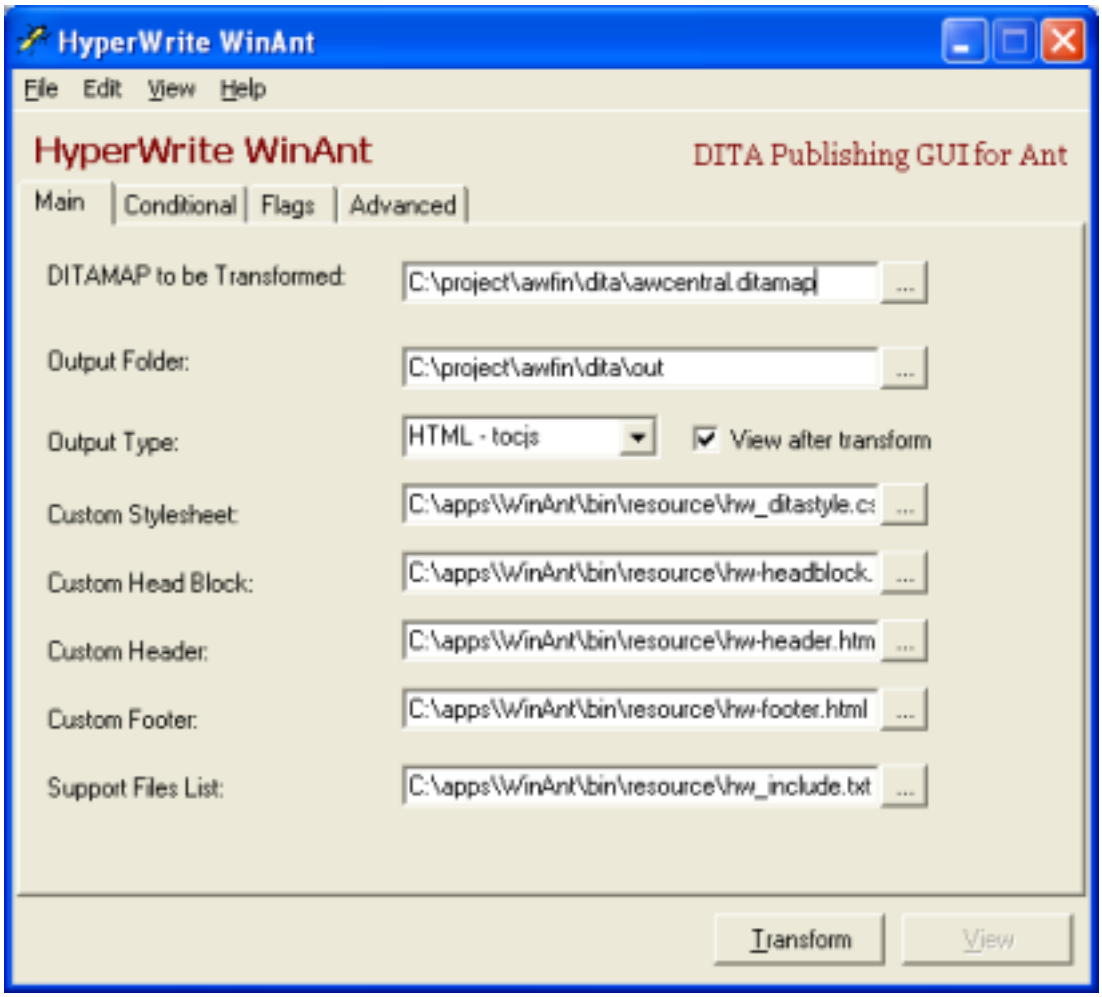


Figure 4: WinANT 1.4 - Main tab

Setup and configuration

WinANT can be downloaded without charge on a “take it as it comes” basis from *http://www.helpml.com/winant_setup.exe*. It installs using a standard Windows installer.

To enable the incorporation of map and alias files in HTML Help output, you have to manually change the following lines in the standard build_dita2htmlhelp.xml DITA OT file from:

```
<param name="HELPMAP" />
<param name="HELPALIAS" />
```

to

```
<param name="HELPMAP" expression="${dita.map.filename.root}.h" />
<param name="HELPALIAS" expression="${dita.map.filename.root}.ali" />
```

Authoring

Nominating the map and alias files in the HTML Help project file effectively *externalises* the context hooks; the context ids and strings are not incorporated into the DITA source files at all. The map and alias files therefore need to be separately authored outside DITA.

Selecting the *map* and *alias* files

The map and alias files are selected in the **Context-Sensitive Help** panel of the **Advanced** tab. These fields are only active if the **Output Type** field on the **Main** tab is set to *HTML Help*.

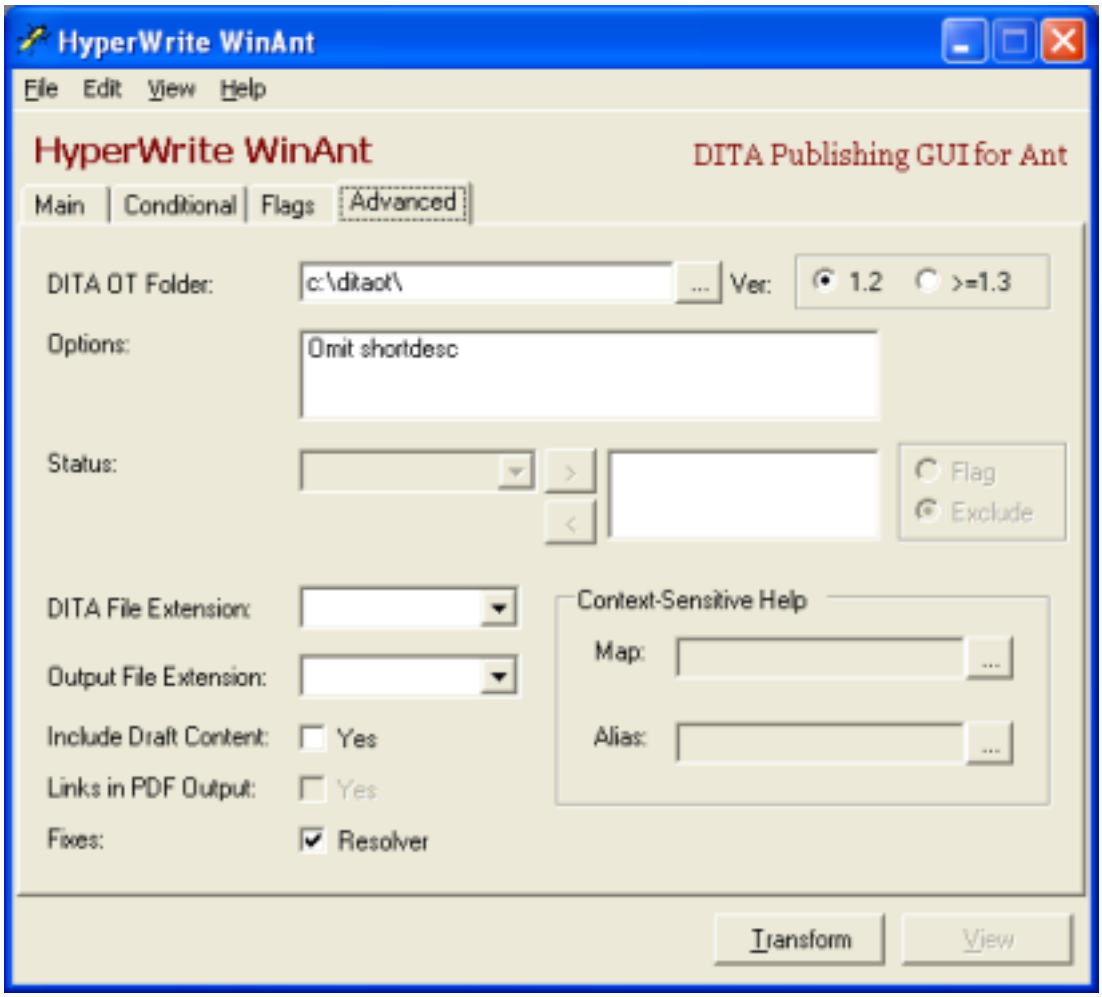


Figure 5: WinANT 1.4 - Advanced tab

Summary

Provided you have the capacity to create the HTML Help map and alias files outside your DITA authoring environment, this technique is a simple way to produce context-sensitive HTML Help. It only requires very minor (and simple) changes to the standard DITA Open Toolkit files.

It is possible that this technique can be combined and integrated with other methods of generating map and alias files from DITA source.

Developing DITA-based Help for Existing Help Authoring Tools

This topic introduces solutions for integrating DITA source (or in some cases DITA output) with existing HATs (Help Authoring Tools).

If your company develops content in multiple authoring tools and uses a HAT (Help Authoring Tool) to integrate those sources into one or more Help delievrables, you can still author in DITA and work with existing HATs.

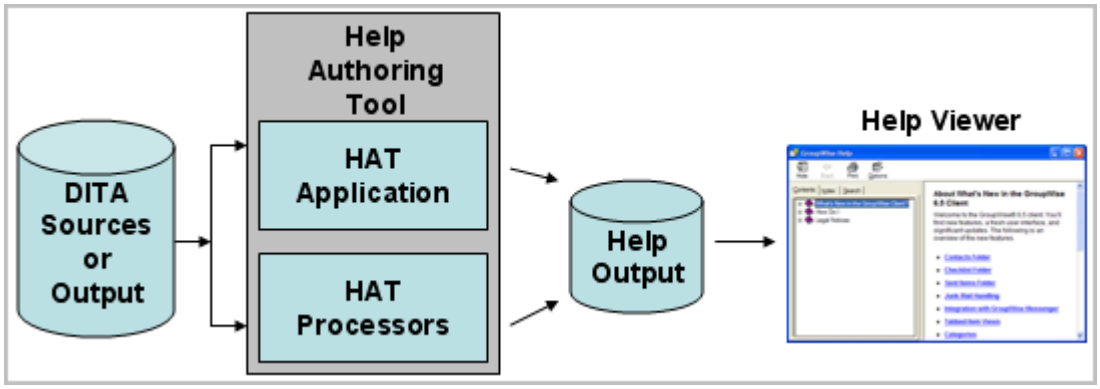


Figure 6: DITA Source to Existing HATs

The following topics in this section of the *Best Practices Guide* explain how to use integrate DITA source files or DITA outout files with existing HATs.

Help Environment	Notes
Adobe RoboHelp	TBD
MadCap Flare	TBD

*RoboHelp [TBS]

RoboHelp

RoboHelp

RoboHelp

*Resources for DITA Help [TBS]

Resources

Resources