

**DITA Help Technologies Guide (DEVELOPMENT
REVIEW 3jan2011)**

Contents

Introduction.....	4
Editorial Preface.....	4
DITA and User Assistance.....	5
Definition of DITA Help.....	5
Contributors.....	6
Document History.....	6
Help Delivery Technologies.....	7
Adobe AIR-based Help.....	7
Browser-based Help.....	8
MadCap DotNet Help.....	10
Eclipse Help.....	10
IDPF ePub Deliverable.....	11
Java Help.....	12
Microsoft HTML Help.....	13
Microsoft Viewer 1.0 Help.....	14
Microsoft WinHelp.....	14
Omni Systems OmniHelp.....	15
Oracle Help.....	15
PDF Deliverable.....	15
PTC Arbortext DMP Help.....	16
Quadralay WebWorks Help.....	17
UNIX Man Pages.....	17
Wiki.....	18
Help Development Tools.....	19
DITA Open Toolkit.....	19
DITA-OT: PDF.....	19
DITA-OT: Browser-based Help.....	19
DITA-OT: Eclipse Help.....	34
DITA-OT: DITA for Publishers Plug-in.....	46
DITA-OT: Microsoft HTML Help.....	47
DITA-OT: Leximation AIR Help Plug-in.....	50
Adobe FrameMaker.....	54
Adobe RoboHelp.....	54
Converting DITA Content to WebHelp using RoboHelp.....	54
DITA Exchange.....	54
JustSystems XMetaL.....	54
MadCap Blaze.....	55
MadCap Flare.....	55
Omni Systems DITA2Go.....	55
SyncRO Soft Ltd Oxygen 12.x Editor.....	55

PTC Arbortext Digital Media Publisher.....	62
Quadralay ePublisher.....	67
SDL XPP.....	67
WinANT Echidna.....	67
WinANT Options Supporting HTML-Based Output.....	68
WinANT Options Supporting Microsoft HTML Help	70
XMLMind DITA Converter.....	73
Help Development Techniques.....	74
Developing Custom DITA-based Help Systems.....	74
Developing DITA-based Help for Existing Help Environments.....	74
DHTML Effects in HTML Generated from DITA.....	75
Dynamic Rendering of DITA into XHTML.....	77
JavaScript-Based Context Sensitive Help.....	78
Resources.....	82

Introduction

When DITA was first described to the world in the article *Introduction to the Darwin Information Typing Architecture* by Don Day, Michael Priestley, and Dave A. Schell, specific reference was made to Help in the opening definition: “and for using that content in delivery modes such as online help.” DITA was designed with Help systems in mind.

Indeed, from the release of OASIS DITA 1.0, the DITA Open Toolkit has supported various forms of Help output. However, Help is a somewhat amorphous term; it means different things to different people. Early adopters of DITA found that while Help documents in common formats like Microsoft HTML Help, HTML and Eclipse Help could be created, some of the accustomed features were absent. Popups couldn't be easily implemented, window layouts couldn't be easily defined, and a tri-pane HTML output seemed impossible. Most importantly, context-sensitivity for most Help formats didn't seem to be supported.

The OASIS DITA Help Subcommittee was formed to address some of the perceived shortcomings of DITA for authoring Help content. The Subcommittee will recommend changes to the DITA standard to further improve functionality in time for DITA 1.3. In the interim, this *DITA Help Technologies Guide* provides explanations of the current best practice for using DITA for Help authoring, and makes practical recommendations that can be applied today.

Tony Self

HyperWrite Pty. Ltd.

Chairperson, OASIS DITA Help Subcommittee

Editorial Preface

This first version of the *DITA Help Technologies Guide* is a work in progress designed to support users, consultants, and vendors working with Help systems and OASIS DITA. Here are a few notes about the Guide to provide context:

- *Endorsement:* OASIS, the DITA Technical Committee, and the DITA TC Help Subcommittee endorse no open-source or commercial technologies supporting DITA. Please view the technologies in this Guide as equally appropriate possible solutions for the particular set of Help-related requirements that you face and not as a particular set of endorsed or recommended technologies or implementations.
- *Technologies reviewed:* The selection of tools and technologies in this first version of the Guide emerged mostly from a series of demos of DITA-related Help technologies done by and for members of the DITA Help Subcommittee. The Subcommittee continues to invite users, consultants, and tools vendors to participate in reviewing this current version of the Guide and in submitting topics for upcoming versions of the Guide. It is our goal to be inclusive (all submissions are welcome) and even-handed (no technologies are 'preferred' over others). If any reader perceives content that does not reflect the goals of inclusivity and even-handedness, contact the Subcommittee.
- *Update schedule:* We hope to update this Guide quarterly. If you, a colleague, your team, or your company has developed DITA-related implementations or technologies that you want to submit to the DITA Help Subcommittee to include in upcoming versions of the Guide, please contact a member of the Subcommittee (see below).
- *Editorial reviews:* The DITA Help Subcommittee works with individual authors in the development and final integration of submitted topics. Final editorial and production controls rests with the Subcommittee.
- *Technical reviews:* Whenever possible, Subcommittee members request that the implementations and tools profiled in the Guide be available for them to test informally. These informal technical reviews are designed primarily to provide contributing authors with feedback. The Subcommittee makes no guarantees that these implementations or tools work as profiled or that the Subcommittee would support these implementations or tools.
- *Feedback:* Please send feedback on the Guide to the following DITA Help Subcommittee members:
 - Tony Self, Chairperson (Tony.Self@hyepwrite.com)
 - Stan Doherty, Secretary (Stan@modularwriting.com)

March 2009

Stan Doherty

Editor, *DITA Help Technologies Guide*

Secretary, DITA TC Help Subcommittee

Member, DITA TC

DITA and User Assistance

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 delivery format for user assistance 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. For example, 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 transforms 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 organizations tend to develop their own individual, custom approaches, using in-house technical expertise to do so.

Moving beyond simple Help systems, however, is currently difficult, but not impossible. The DITA Technical Committee is developing some enhancements 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

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 and window definitions. These 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.

Contributors

The following people contributed articles to Version 1.0 of this *DITA Help Technologies Guide*.

Contributor	Corporate Affiliation	OASIS Affiliation(s)
Jeff Antley	IBM Corporation	Member, OASIS DITA Technical Committee Help Subcommittee
Stan Doherty	Individual	Secretary, OASIS DITA Technical Committee Help Subcommittee
Chris Goolsby	PTC Corporation / Arbortext	Member, OASIS DITA Technical Committee Help Subcommittee
Scott Prentice	Individual	Member, OASIS DITA Technical Committee Help Subcommittee
Dave Resch	Sybase Inc.	Member, OASIS DITA Technical Committee Help Subcommittee
Tony Self	HyperWrite Pty. Ltd.	Chairperson, OASIS DITA Technical Committee Help Subcommittee Member, OASIS DITA Adoption Committee
Joe Welinske	Individual	Member, OASIS DITA Technical Committee Help Subcommittee

Document History

DITA Help Technologies Guide Version 1.0 (March 2009)

Reviewer	Review Date(s)	Review Status	Approval Status
OASIS DITA Technical Committee Help Subcommittee	Ongoing	Complete	Yes
OASIS DITA Technical Committee	02/23/09	In progress	No decision
OASIS DITA Adoption Committee	02/23/09	In progress	No decision
DITA Help practitioners	Ongoing	In progress	N/A

Help Delivery Technologies

The Help formats discussed in this section can be created from applications or tools that create or process DITA-based content. The DITA Open Toolkit (DITA-OT) is considered one such tool and is listed along with other commercially available authoring and publishing tools.

Table 1: Help Technology Comparison

Help Technology	Platforms	Development Tools
<i>Adobe AIR-based Help</i>	Windows, Mac, Linux, Mobile	DITA-OT, Adobe RoboHelp, MadCap Flare
<i>Browser-based Help</i>	Windows, Mac, Linux, Unix, Mobile	DITA-OT, Adobe RoboHelp, JustSystems XMetaL, MadCap Flare, Omni Systems DITA2Go, Oxygen XML Editor, PTC DMP, Quadralay ePublisher, XMLMind DITA Converter
<i>MadCap DotNet Help</i>	Windows	MadCap Flare
<i>Eclipse Help</i>	Windows, Mac, Linux, Unix	DITA-OT, Omni Systems DITA2Go, Quadralay ePublisher, XMLMind DITA Converter
<i>IDPF ePub Deliverable</i>	Windows, Mac, Linux, Unix, Mobile	DITA-OT, Adobe RoboHelp, Quadralay ePublisher, XMLMind DITA Converter
<i>Java Help</i>	Windows, Mac, Linux, Unix	DITA-OT, Omni Systems DITA2Go, Quadralay ePublisher, XMLMind DITA Converter
<i>Microsoft HTML Help</i>	Windows	DITA-OT, Adobe RoboHelp, MadCap Flare, Omni Systems DITA2Go, PTC DMP, Quadralay ePublisher, XMLMind DITA Converter
<i>Microsoft Viewer 1.0 Help</i>	Windows	<<none yet?>>
<i>Microsoft WinHelp</i>	Windows	Quadralay ePublisher, Omni Systems DITA2Go
<i>Omni Systems OmniHelp</i>	Windows, Mac, Linux, Unix	Omni Systems DITA2Go
<i>Oracle Help</i>	Windows, Mac, Linux, Unix	Quadralay ePublisher, Omni Systems DITA2Go
<i>PDF Deliverable</i>	Windows, Mac, Linux, Unix, Mobile	DITA-OT (plus FO processor), Adobe FrameMaker, DITA Exchange, MadCap Blaze/Flare, PTC Arbortext Publishing Engine, Quadralay ePublisher, SDL XPP, XMLMind DITA Converter
<i>PTC Arbortext DMP Help</i>	Windows, Mac, Linux, Unix	PTC DMP
<i>Quadralay WebWorks Help</i>	Windows, Mac, Linux, Unix	Quadralay ePublisher
<i>UNIX Man Pages</i>	Linux, Unix	DITA-OT <<??>>
<i>Wiki</i>	Windows, Mac, Linux, Unix	Quadralay ePublisher

Adobe AIR-based Help

Adobe AIR-based Help (or "AIR Help") is any user assistance application that was developed from the Adobe AIR (Adobe Integrated Runtime) development platform. Help authoring tools and a DITA-OT plug-in offer pre-packaged AIR Help options which provide numerous features and customization options. You can also develop highly customized user assistance applications from scratch using the open source AIR and Flex development languages.

Adobe AIR allows the creation of cross-platform desktop applications using web technologies like HTML, JavaScript, Flex, and Flash. The same AIR application can be installed on Windows, Mac, and Linux systems once the AIR runtime has been installed (this is similar in concept to a Java application requiring the prior installation of the Java runtime). AIR Help is just an AIR application that is intended to be used for user assistance.

Most AIR Help systems make use of the embedded HTML browser control that is provided by the AIR runtime. Because of this, the HTML and JavaScript coding used by your Help system needs to be designed for only one browser. The embedded browser is based on WebKit, the open source browser project that is also used for the Safari browser. Developing HTML and JavaScript for a single browser can save considerable time in both development and testing.

One feature of AIR is the ability to automatically check for updates. If an AIR Help system is so enabled, the user is notified of an available update and it can be easily downloaded and installed. This makes it very easy to provide updated documentation and Help content as needed.

AIR applications have the ability to read and write to the local file system as well as to remote files on the Internet. AIR Help systems can also read and parse XML files as well as HTML. This means that your Help system can dynamically incorporate data from the web as well as communicating with the locally installed software application. It is even possible to develop an AIR application that can read and display content directly from DITA XML files if so desired.

Because an AIR application can read and write to the local file system, each "air" file must be digitally signed by the creator. For development purposes you can use a "self-signed" certificate, but before you ship an AIR Help file to your end users, you should purchase a certificate from a provider.

Tools and methods for creating AIR Help from DITA:

DITA Open Toolkit: DITA-OT: *Leximation AIR Help Plug-in* Leximation provides a plug-in for the DITA-OT which generates an AIR Help deliverable. This plug-in is intended to be used as a framework from which to create your own custom AIR Help system, or it can be used as delivered.

Adobe RoboHelp RoboHelp allows the importing of a DITA map and topics which can be exported as AIR Help. You can also import FrameMaker files (DITA-sourced or binary) into RoboHelp, which can then be used as the source for an AIR Help deliverable. Once the project and template files are set up, this import and build process can be automated from the command line.

MadCap Flare Flare allows the importing of a DITA map and topics which can be exported as AIR Help. Flare also imports many other formats which can be combined with the DITA files to create and AIR Help deliverable. Once the project and template files are set up, this import and build process can be automated from the command line.

Browser-based Help

Browser-based Help is essentially a collection of HTML-based content that is rendered in the end user's web browser. This type of Help delivery is generally considered to be cross-platform since it can be used on any platform or operating system that provides a web browser application. There are no specific criteria that define the features that should exist in browser-based Help; in order to assess this type of Help delivery option, you must review the features provided by each tool.

Browser-based Help is often called "web help" because the source of the Help system may exist remotely (on the web) or locally on the file system (or both). It is probably the most flexible type of Help delivery option, but may be the most time consuming from both a development and testing perspective, since you will need to design for multiple web browsers and operating systems.

We identify two types of browser-based Help, "generic" and "enhanced." Generic browser-based Help has no specific installation requirements, meaning that it should function equally well on all platforms in all browsers, with no additional tool support required for deployment. Enhanced browser-based Help also runs in a web browser, but provides additional

features that require a specific browser or platform, or the installation of additional technologies to support those features. In some cases the same tool will create both types of browser-based Help and will be listed twice.

Tools and methods for creating "generic" Browser-based Help from DITA:

DITA Open Toolkit: DITA-OT: Browser-based Help

The DITA-OT provides standard transformations to generate browser-based Help. At this time the default "XHTML" target provides a "Contents" file and topics which are designed to be wrapped up in a frameset, however this frameset must be created manually or by other means. There are additional OT plugins that will provide additional support and features.

Features: Contents, Index

Adobe RoboHelp

RoboHelp allows the importing of a DITA map and topics which can be exported as web help. You can also import FrameMaker files into RoboHelp, which can then be used as the source for this deliverable. The "generic" output options from RoboHelp are called "WebHelp" and "FlashHelp."

Features: Contents, Index, Search, Browse

JustSystems XMetaL

XMetaL provides a default transformation option which generates a web help output.

Features: Contents, Index, Search

MadCap Flare

Flare allows the importing of a DITA map and topics which can be exported as a format called "WebHelp."

Features: Contents, Index, Search, Browse

Omni Systems DITA2Go

DITA2Go exports a browser-based Help format called "OmniHelp" as one of its default options.

Features: Contents, Index, Search, Browse

#unique_31

Oxygen provides a default transformation option which generates a web help output.

Features: Contents, Search

Quadralay ePublisher

ePublisher exports from DITA, Word, and FrameMaker documents to a browser-independent format called "WebWorks Help."

Features: Contents, Index, Search, Browse

XMLMind DITA Converter

XMLmind DITA Converter provides a default transformation option which generates a web help output.

Features: Contents, Browse

Tools and methods for creating "enhanced" Browser-based Help from DITA:

Adobe RoboHelp

RoboHelp allows the importing of a DITA map and topics which can be exported as webhelp. You can also import FrameMaker files into RoboHelp, which can then be used as the source for this deliverable. The "enhanced" output options from RoboHelp are called "WebHelp Pro" and "FlashHelp Pro" which are server-based versions of their generic web help options.

Requirements: Apache web server, RoboHelp Server

Features: Contents, Index, Search, Browse, Usage tracking, Project merging, Increased speed

Tools and methods for creating "enhanced" Browser-based Help from DITA:

Eclipse Help

Eclipse Help is a browser-based Help format that requires the installation of Java and is installed on a web server.

Requirements: Web server, Java

Features: Contents, Index, Search, Browse, <<MORE?>>

MadCap Flare

Flare allows the importing of a DITA map and topics which can be exported as a format called "WebHelp Plus."

Requirements: IIS web server, Windows Indexing Service

Features: Contents, Index, Search, Browse, Faster search, Search of binary files (PDF, DOC, etc)

PTC Arbortext Digital Media Publisher

PTC DMP provides a default transformation option which generates a custom WebHelp output.

Requirements: Java-based server delivery or application integration.

Features: Contents, Index, Search, Filtering, Bookmarks, Comments, Multiple languages in single Help system

Quadralay ePublisher

ePublisher exports from DITA, Word, and FrameMaker documents to Java Help and Eclipse Help in addition to multiple Wiki formats as possible browser-based Help options.

Requirements: Web server, Java

Features: Contents, Index, Search, Browse, <<MORE?>>

MadCap DotNet Help

<<basic info about the dotnet technology>>

<<more info about the dotnet technology>>

<<DITA-specific issues>>

Tools and methods for creating DotNet Help from DITA:

MadCap Flare

Currently the only DITA-based tool that exports DotNet Help is Flare.

<<more?>>

<<references for more info>>

Eclipse Help

<<basic info about the eclipse help technology>>

<<more info about the eclipse help technology>>

<<DITA-specific issues>>

Tools and methods for creating Eclipse Help from DITA:

DITA Open Toolkit

The DITA-OT provides standard transformations to generate eclipse Help.

<<more?>>

<<references for more info>>

Omni Systems DITA2Go

Exports Eclipse Help as one of its default options.

<<more?>>

<<references for more info>>

Quadralay ePublisher

Quadralay ePublisher ...

<<more?>>

XMLMind DITA Converter

info...

<<more?>>

<<references for more info>>

Others??

Others ...

<<more?>>

<<references for more info>>

IDPF ePub Deliverable

An ePub (or more correctly "EPUB") is actually a collection of files wrapped up in a single file archive container (like a CHM). These files define the content, navigation and formatting of the deliverable. An ePub is viewed on a dedicated eBook device or with an eBook reader application. ePub is just one of a number of eBook formats, but it is the most widely accepted by reader applications and devices. Although ePub is not a native format for the Amazon Kindle, an open (non-DRM) ePub can be converted into the MOBI format which can be read on a Kindle.

The ePub specification is maintained by IDPF (International Digital Publishing Forum). ePub 2.0 Became an official standard in September 2007, superseding the older Open eBook standard from 1999 ("OEB"). ePub 2.0.1 was approved in May 2010 and is the current stable release. The ePub 3.0 first public draft was released in Feb 2011.

The ePub specification is a combination of the following specs:

Open Publication Structure (OPS)	Defines the standard for representing the content of electronic publications.
Open Packaging Format (OPF)	Defines the structure and semantics as well as the mechanism by which the various components of an OPS publication are related.
Open Container Format (OCF)	Defines the mechanism by which all components of an electronic publication are packaged into a single deliverable.

The ePub specification provides for a navigation component (a TOC), but no index. An index can be created as pages at the end of the book, but it's not an integral part of the specification. Content within the document can contain most of the objects that you see in HTML, tables, lists, images, as well as CSS-defined inline formatting. It is important to note that although the ePub specification for content is based on XHTML 1.1, there are many exceptions, so you cannot assume that whatever displays in a web browser will work in an eBook reader. In fact, you will find that no two eBook readers will render the same content in the same way, so in general it's best to keep the formatting and structure as simple as possible.

The underlying technologies in an ePub collection are XML, HTML, and CSS. The content may include DRM (digital rights management) or be open (DRM restricts the use on specific readers). When viewed in a reader, the content is reflowed to fit the constraints of the rendering device or application. This is similar to the way HTML displays in a web browser and unlike a PDF which retains its formatting as intended by the author. This format works well on a variety of screen sizes, but certain types of objects (tables in particular) may have trouble on very small screens (phones). Most ePub readers work with a screen-based "paged" concept, trying to emulate a book. This means that the content is reflowed and paged based on the available screen real estate and the current zoom scaling. When you change the font size in an ePub reader (effectively changing the zoom scaling), you'll typically see the page numbers update to accommodate the new layout. Because of this you can't refer to page numbers in an ePub since they will be different for each device and user settings.

Tools and methods for creating ePub deliverables from DITA:

DITA Open Toolkit: DITA-OT: DITA for Publishers Plug-in

DITA for Publishers generates both ePub and Kindle eBook formats from DITA content through the DITA-OT. This plugin also provides a number of specializations for common publishing components, as well transformation frameworks for converting styled Word documents to DITA and for generating InCopy and InDesign documents from DITA content.

Adobe RoboHelp

RoboHelp allows the importing of a DITA map and topics which can be exported as ePub. You can also import FrameMaker files (DITA-sourced or binary) into RoboHelp, which can then be used as the source for an ePub deliverable. Once the project and template files are set up, this import and build process can be automated from the command line.

Quadralay ePublisher

ePublisher generates ePub as one of its standard output formats from DITA source as well as from FrameMaker or Word.

XMLMind DITA Converter

XMLmind DITA Converter (ditac) generates ePub as one of its standard output formats from DITA source files.

Java Help

<<basic info about java help>>

<<more info about java help>>

<<DITA-specific issues>>

Tools and methods for creating Java Help from DITA:

DITA Open Toolkit

The DITA-OT provides standard transformations to generate java Help.

<<more?>>

<<references for more info>>

Omni Systems DITA2Go

Exports JavaHelp as one of its default options.

<<more?>>

<<references for more info>>

Quadralay ePublisher

Quadralay ePublisher ...

<<more?>>

XMLMind DITA Converter

info...

Tools and methods for creating Java Help from DITA:

<<more?>>

<<references for more info>>

Others??

Others ...

<<more?>>

<<references for more info>>

Microsoft HTML Help

(from wikipedia) Microsoft Compressed HTML Help (.CHM) - Based on compiled HTML and other data such as images and JavaScript. HTML Help 1.0 was released in 1997. In 2006, it was available from Microsoft as HTML Help 1.4.

<<more info about html help>>

<<DITA-specific issues>>

Tools and methods for creating HTML Help from DITA:

DITA Open Toolkit

The DITA-OT provides standard transformations to generate HTML Help.

<<more?>>

<<references for more info>>

Adobe RoboHelp

RoboHelp 8 allows the importing of a DITA map and topics which can be exported as HTML Help. You can also import FrameMaker files into RoboHelp, which can then be used as the source for this deliverable.

<<more?>>

<<references for more info>>

MadCap Flare

Flare 5 and 6 allows the importing of a DITA map and topics which can be exported as HTML Help.

<<more?>>

<<references for more info>>

Omni Systems DITA2Go

Exports HTML Help as one of its default options.

<<more?>>

<<references for more info>>

PTC DMP

PTC DMP ...

<<more?>>

<<references for more info>>

Quadralay ePublisher

ePublisher ...

<<more?>>

<<references for more info>>

XMLMind DITA Converter

info...

Tools and methods for creating HTML Help from DITA:

<<more?>>

<<references for more info>>

Others??

Others ...

<<more?>>

<<references for more info>>

Microsoft Viewer 1.0 Help

<<basic info about the ms viewer technology>>

<<more info about the ms viewer technology>>

<<DITA-specific issues>>

Tools and methods for creating MS Viewer Help from DITA:

DITA Open Toolkit

NOT YET .. RIGHT?

<<more>>

<<references for more info>>

Others?

Others...

<<more>>

<<references for more info>>

Microsoft WinHelp

(from wikipedia) Microsoft WinHelp (.HLP) - Based on the Rich Text Format, this was the industry standard for Windows 3.1 and Windows 95/NT. The popular Windows Help program (WinHlp32.exe) was included with all Windows operating systems from Windows 3.0 until the Windows XP operating system. However, the help engine is not included with Windows Vista and is only available as a download.

<<more info about the winhelp technology>>

<<DITA-specific issues>>

Tools and methods for creating WinHelp from DITA:

DITA Open Toolkit

DOES THE OT EXPORT WINHELP?? DON'T THINK SO.

<<more?>>

<<references for more info>>

Omni Systems DITA2Go

Exports WinHelp as one of its default options.

<<more?>>

<<references for more info>>

Tools and methods for creating WinHelp from DITA:

[Quadralay ePublisher](#)

Quadralay ePublisher ...

<<more?>>

Omni Systems OmniHelp

<<basic info about the omnihelp technology>>

<<more info about the omnihelp technology>>

<<DITA-specific issues>>

Tools and methods for creating OmniHelp from DITA:

Omni Systems DITA2Go

Currently the only DITA-based tool that exports OmniHelp is DITA2Go...

<<more>>

<<references for more info>>

Oracle Help

(from wikipedia) Two formats developed by the Oracle Corporation: Oracle Help for Java (OHJ) and Oracle Help for Web (OHW).

<<more info about oracle help>>

<<DITA-specific issues>>

Tools and methods for creating Oracle Help from DITA:

Omni Systems DITA2Go

Currently the only DITA-based tool that exports Oracle Help is DITA2Go...

<<more>>

<<references for more info>>

[Quadralay ePublisher](#)

Quadralay ePublisher ...

<<more?>>

PDF Deliverable

The most common use for PDF is to deliver an online or electronic version of a printed book. Because this format does not reflow or modify the content based on the window size or user's settings, the content creator can be ensured that what the user sees is exactly what was intended. It is due to this feature of PDF that makes it less likely to be used for online Help, which is typically not an online representation of a printed book.

In order to effectively use PDF as an online Help delivery format, the Help developer would probably set the "page size" to something small enough that the end user would be able to use it while interacting with the application being documented. PDF documents can contain hyperlinks, which are very important for effective online Help, and it provides a built-in search mechanism. A PDF that's used for interactive online viewing should also probably not make use of text columns, while a PDF that's intended for print may make effective use of columns.

Tools and methods for creating PDFs from DITA:

DITA Open Toolkit

The default PDF target in the DITA-OT uses XSL-FO and an installed FO rendering engine to generate PDFs from DITA. There are numerous FO processors available for this type of use.

- Apache FOP
- Renderex XEP
- Antenna House Formatter
- Lunasil Xinc
- Altsoft Xml2PDF

XSL-FO has the benefit of having a very low entry cost since the format itself and at least one FO processor (Apache FOP) are open source and freely available. Another significant benefit of XSL-FO is that it is designed to work with all written languages. On the other hand, XSL-FO development can be very expensive and time consuming and it does have some formatting and layout limitations. Many of these limitations are overcome with processor-defined extensions, but because these extensions are specific to each processor, your FO stylesheets may not be portable.

References to more information on FO processors:

- http://gilbane.com/gilbane_report.pl/94/XSLFO_Ready_for_Prime_Time.html
- <http://www.sagehill.net/docbookxsl/FOprocessors.html>
- http://en.wikipedia.org/wiki/XSL_Formatting_Objects

Adobe FrameMaker

FrameMaker natively exports PDFs and can open DITA files created by other authoring tools.

DITA Exchange

Converts DITA into OpenXML, then opens that in Word for formatting and publishing.

MadCap Blaze and MadCap Flare

Both products import DITA, FrameMaker, Word and other formats. They use a proprietary rendering engine to export to PDF. Provides a GUI for layout and formatting definition.

PTC Arbortext Digital Media Publisher

High-end PDF formatting and publishing engine.

Quadralay ePublisher

Quadralay ePublisher can be used to generate PDFs from DITA. ePublisher provides a graphical user interface for defining the layout and formatting of the PDF and uses an available FO processor (Apache FOP by default) as the rendering engine.

SDL XPP

High-end PDF formatting and publishing engine.

XMLMind DITA Converter

Uses the XMLmind XSL-FO Converter to create RTF, WordML, DOCX, or ODT files from DITA and XSL-FO to allow the export to PDF.

PTC Arbortext DMP Help

<<basic info about the dmp help technology>>

<<more info about the dmp help technology>>

<<DITA-specific issues>>

Tools and methods for creating DMP Help from DITA:

PTC DMP

PTC DMP ...

<<more>>

<<references for more info>>

Others??

Others ...

<<more>>

<<references for more info>>

Quadralay WebWorks Help

<<basic info about webworks help>>

<<more info about webworks help>>

<<DITA-specific issues>>

Tools and methods for creating WebWorks Help from DITA:

Quadralay ePublisher

ePublisher ...

<<more?>>

<<references for more info>>

UNIX Man Pages

<<basic info about the man page technology>>

<<more info about the man page technology>>

<<DITA-specific issues>>

Tools and methods for creating Man Pages from DITA:

DITA Open Toolkit

PEOPLE ARE DOING THIS .. HOW?

<<more>>

<<references for more info>>

Others?

Others...

<<more>>

<<references for more info>>

Wiki

<<basic info about wiki>>

<<more info about wiki>>

<<DITA-specific issues>>

Tools and methods for creating Wiki from DITA:

DITA Open Toolkit

The DITA-OT ... Wikis ??.

<<more?>>

<<references for more info>>

Quadralay ePublisher

exports Confluence, MediaWiki, MoinMoin

<<more?>>

<<references for more info>>

Others??

Others ...

<<more?>>

<<references for more info>>

Help Development 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 deliverables, you can still author in DITA and work with existing HATs.

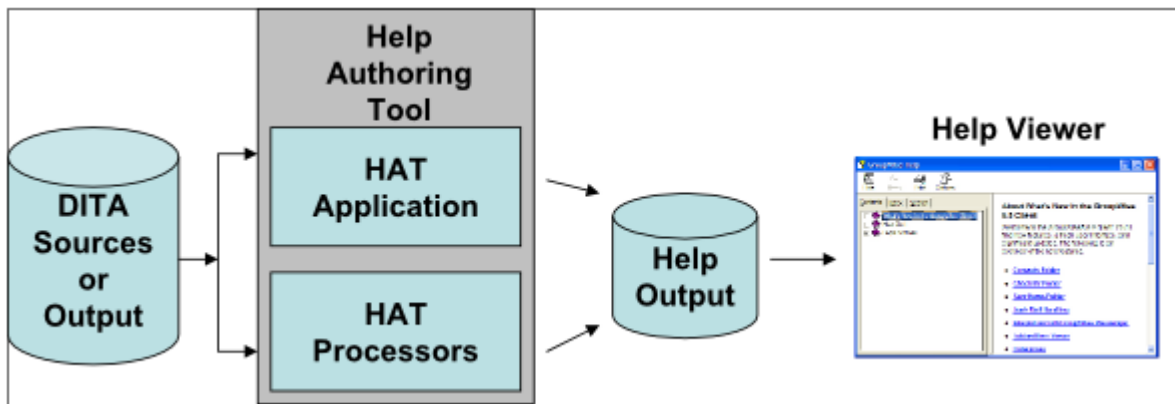


Figure 1: DITA Source to Existing HATs

The following topics in this section of the *DITA Help Technologies Guide* explain how to use integrate DITA source files or DITA output files with existing HATs.

- [#unique_41](#)

DITA Open Toolkit

<<info here>>

DITA-OT: PDF

<<info here>>

DITA-OT: Browser-based Help

HTMLSearch Plug-in

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`.

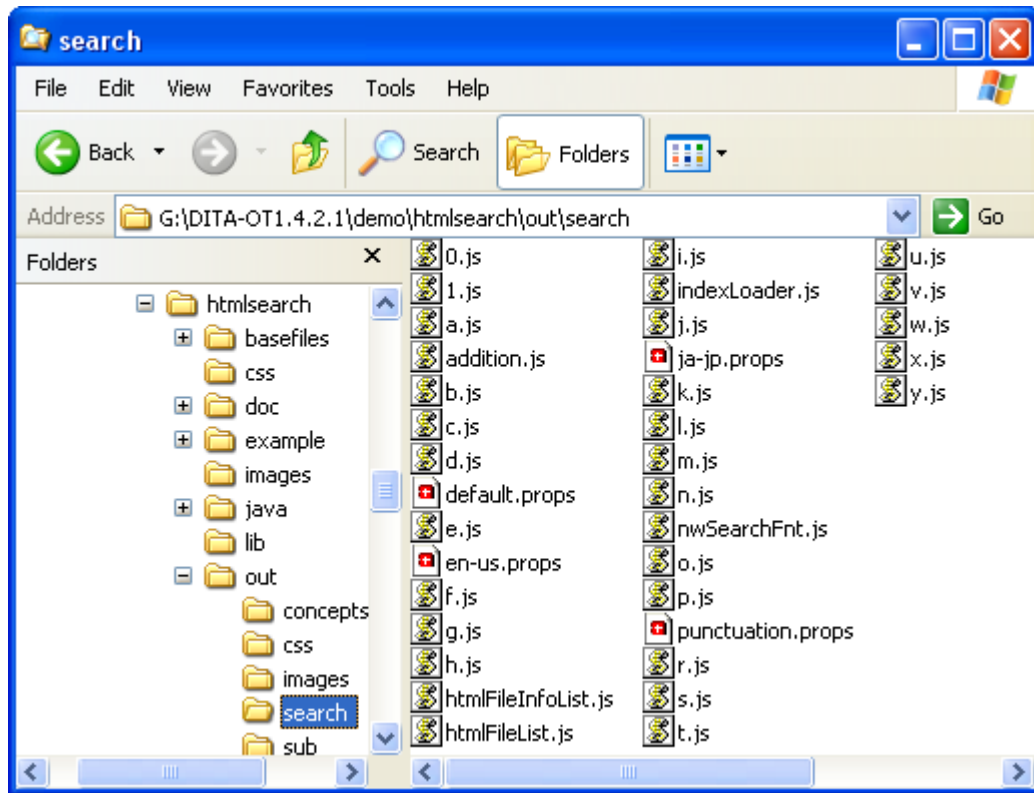


Figure 2: HTMLSearch JavaScript Output

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

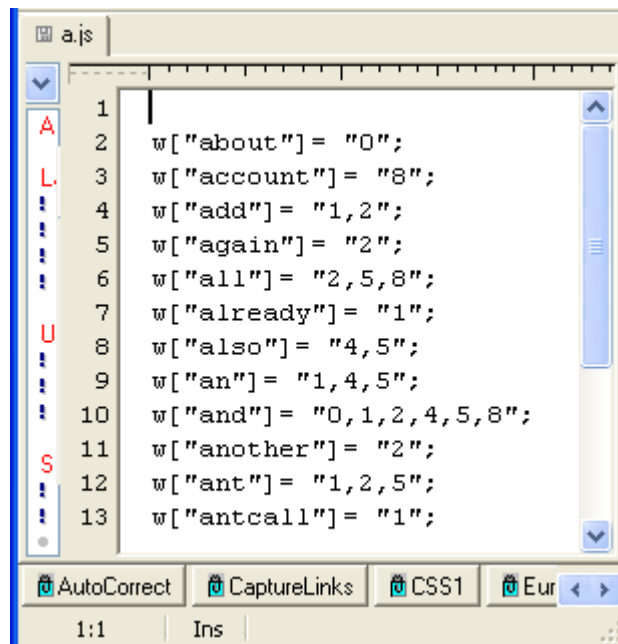
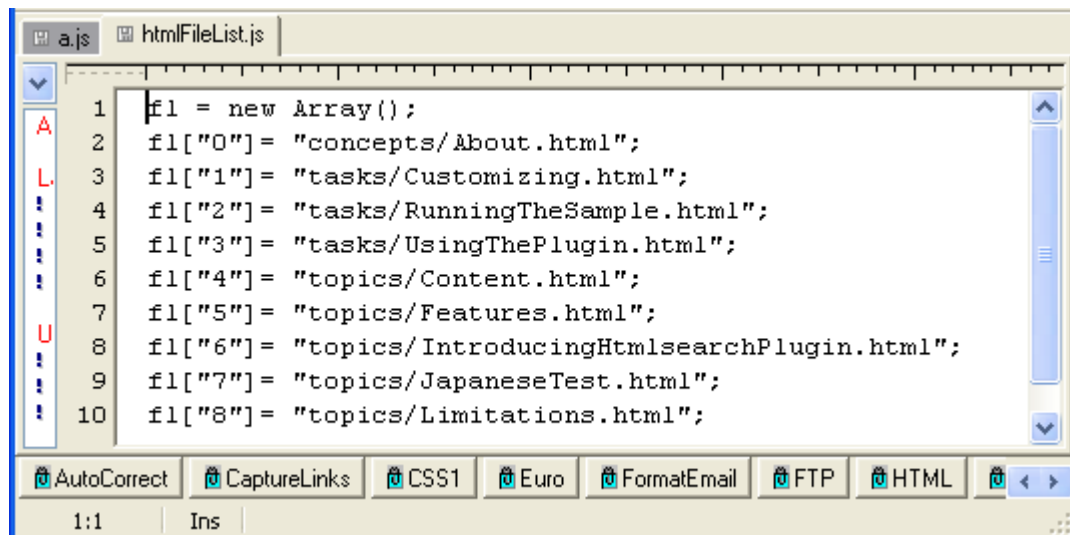


Figure 3: HTMLSearch JavaScript Search Array

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`.



```

1 fl = new Array();
2 fl["0"] = "concepts/About.html";
3 fl["1"] = "tasks/Customizing.html";
4 fl["2"] = "tasks/RunningTheSample.html";
5 fl["3"] = "tasks/UsingThePlugin.html";
6 fl["4"] = "topics/Content.html";
7 fl["5"] = "topics/Features.html";
8 fl["6"] = "topics/IntroducingHtmlsearchPlugin.html";
9 fl["7"] = "topics/JapaneseTest.html";
10 fl["8"] = "topics/Limitations.html";

```

Figure 4: HTMLSearch Keyword-to-Target-File Array

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.



Figure 5: HTMLSearch UI

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

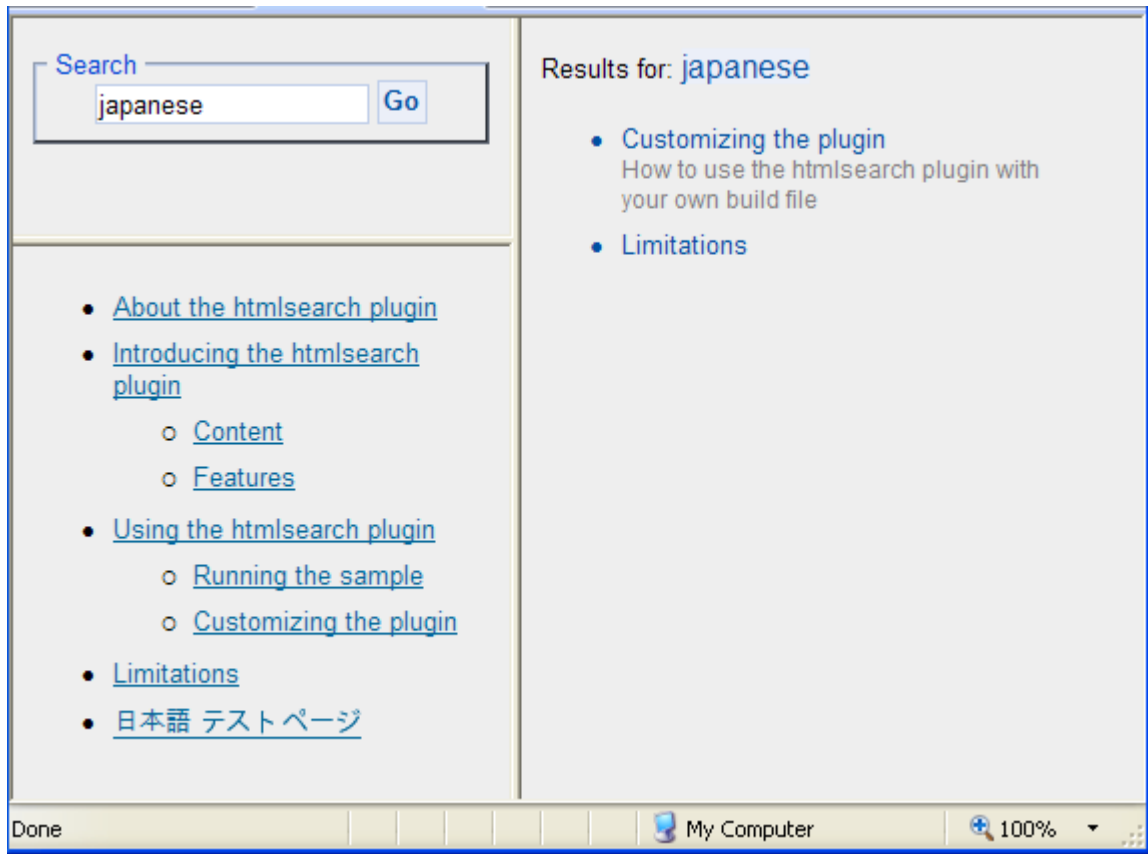


Figure 6: HTMLSearch UI

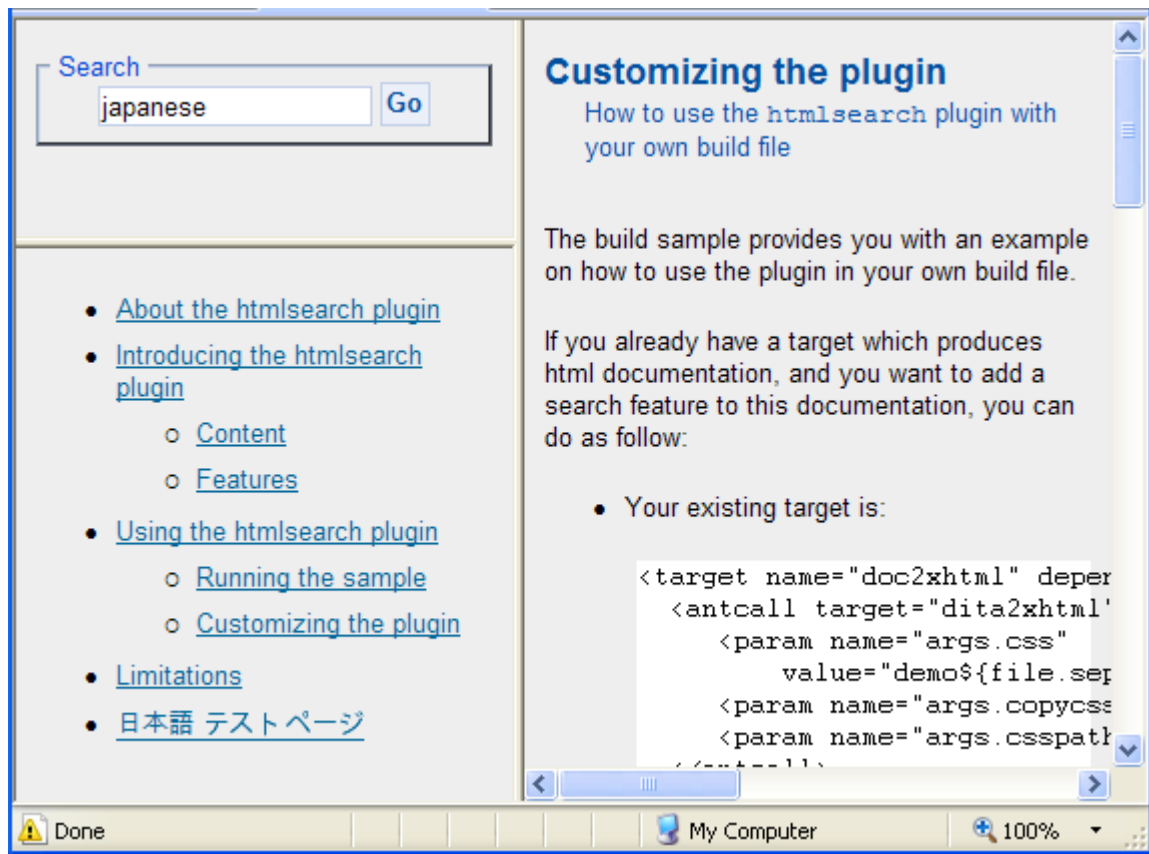


Figure 7: HTMLSearch UI

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.

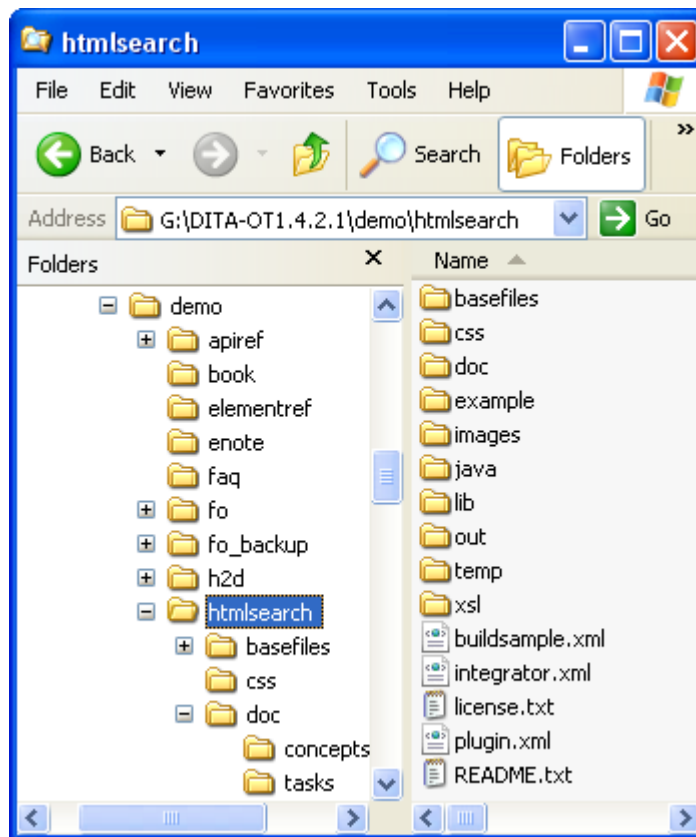


Figure 8: HTMLSearch Plug-in Installation

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 plug-in or plug-ins 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 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 DITA map navtitles:* If the navtitles in your DITA map 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+"\">"+tempTitle+"</a>"+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.

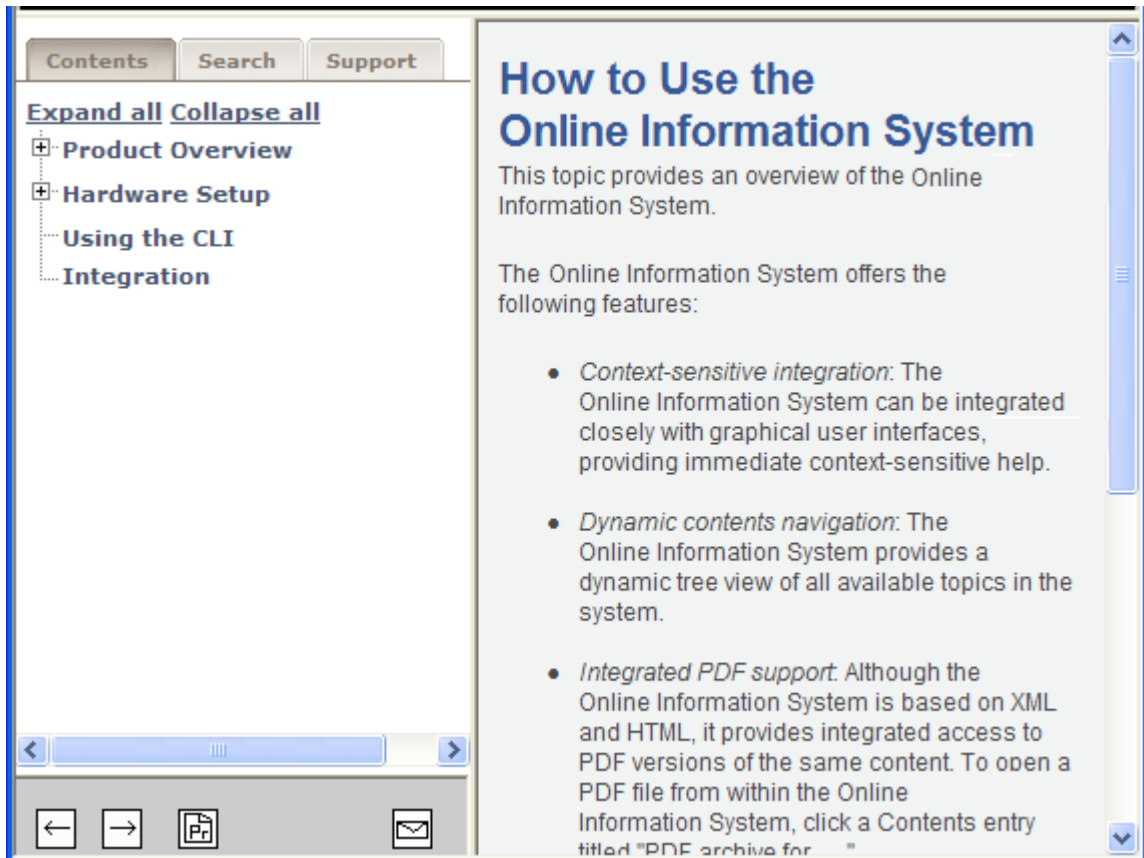


Figure 9: Integrated HTMLSearch

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

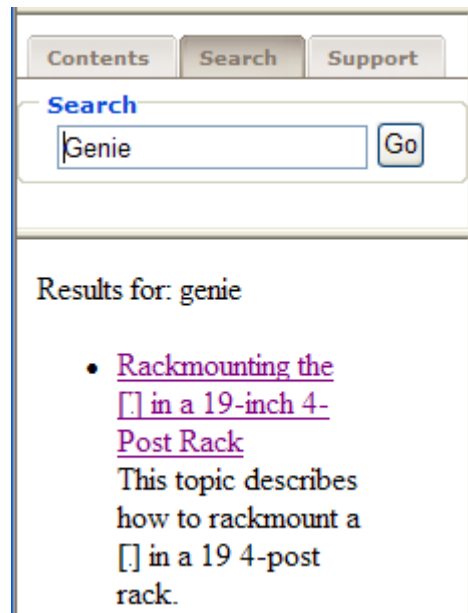


Figure 10: Integrated HTMLSearch

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

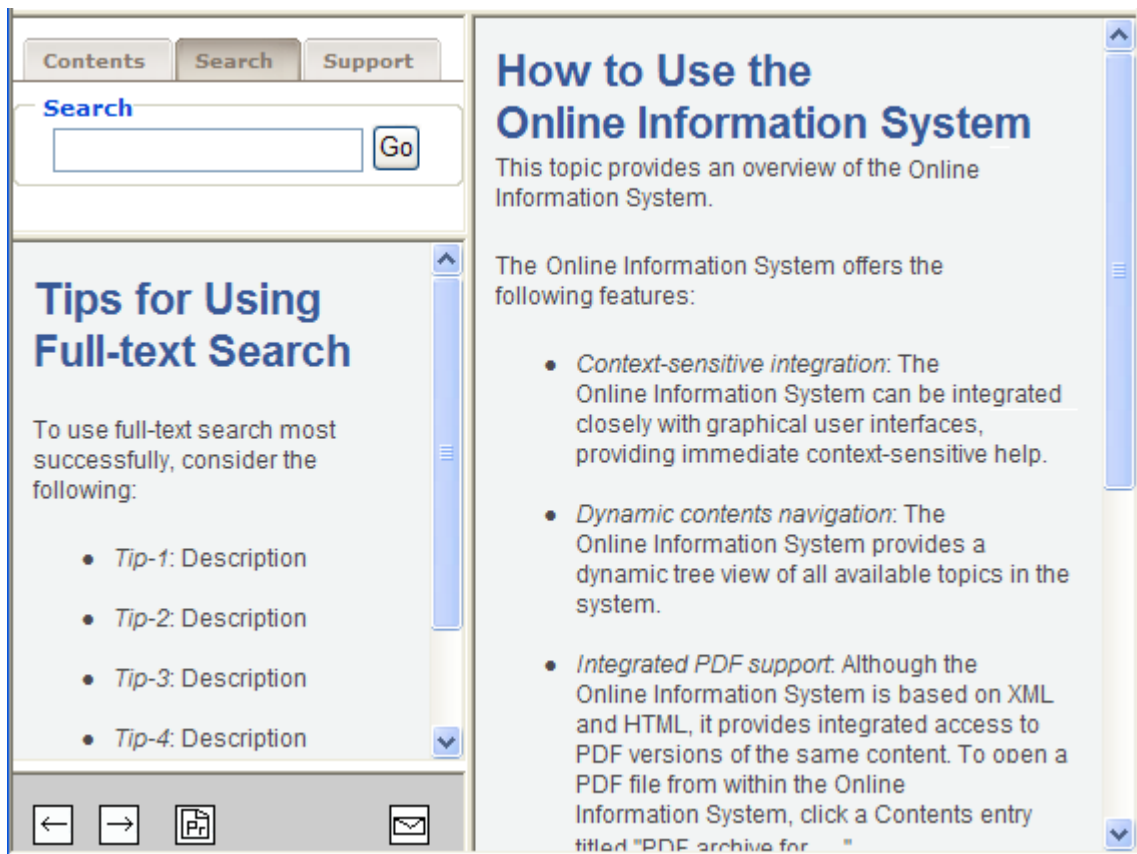


Figure 11: Integrated HTMLSearch

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

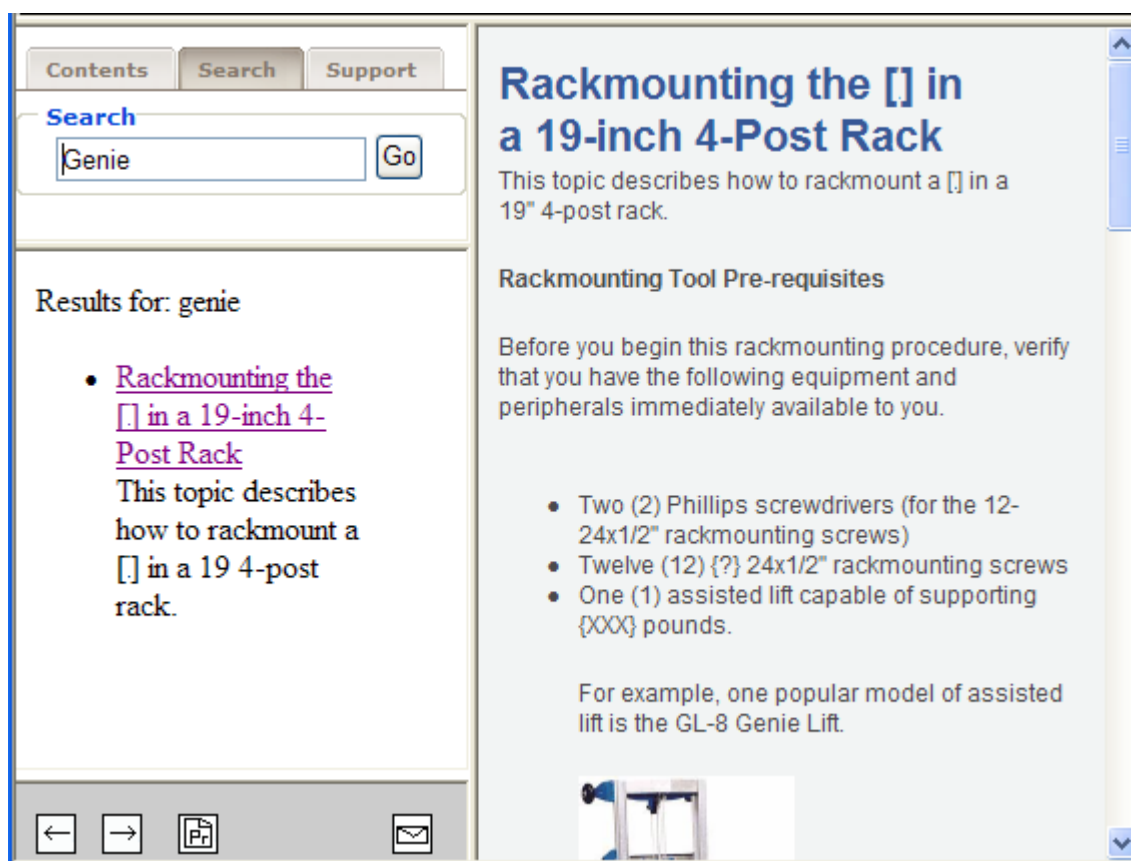


Figure 12: Integrated HTMLSearch

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.

TOCJS and TOCJSBIS Plug-ins

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.

Expand all Collapse all

- [-] DITA elements
- [-] DITAVAL elements
 - alt-text**
 - endflag
 - prop
 - revprop
 - startflag
 - style-conflict
 - val
- [-] Commonly referenced attribute
- [-] Domains
- [-] Appendix

alt-text

An element allowed inside either startflag or endflag to provide alternate text for an image, when the imageref attribute sets an image to be used for flagging. The default alternate text for revprop start of change is a localized translation of "Start of change". The default alternate text for revprop end of change is a localized translation of "End of change".

Contains

text data

Contained by

[startflag](#), [endflag](#)

Example

See the example in the [<val>](#) description.

Figure 13: OASIS DITA Technical Specification in tocjs

- *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 synchronizing 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/>

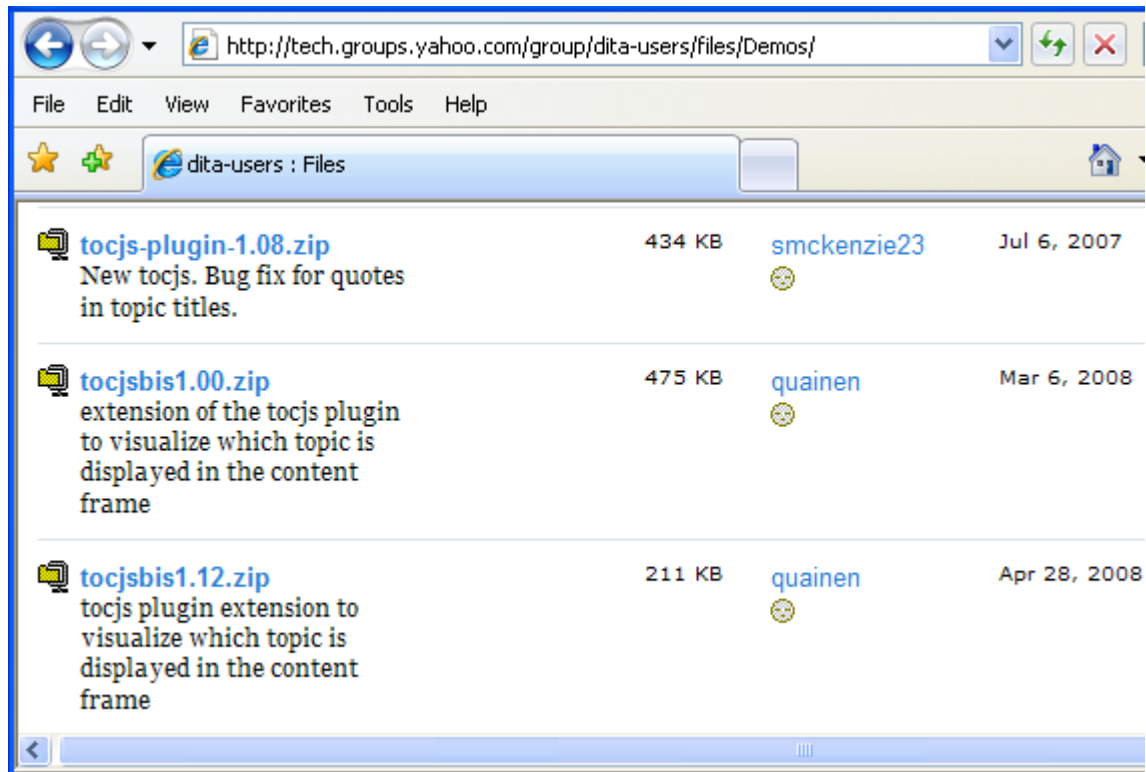


Figure 14: tocjs Download

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.

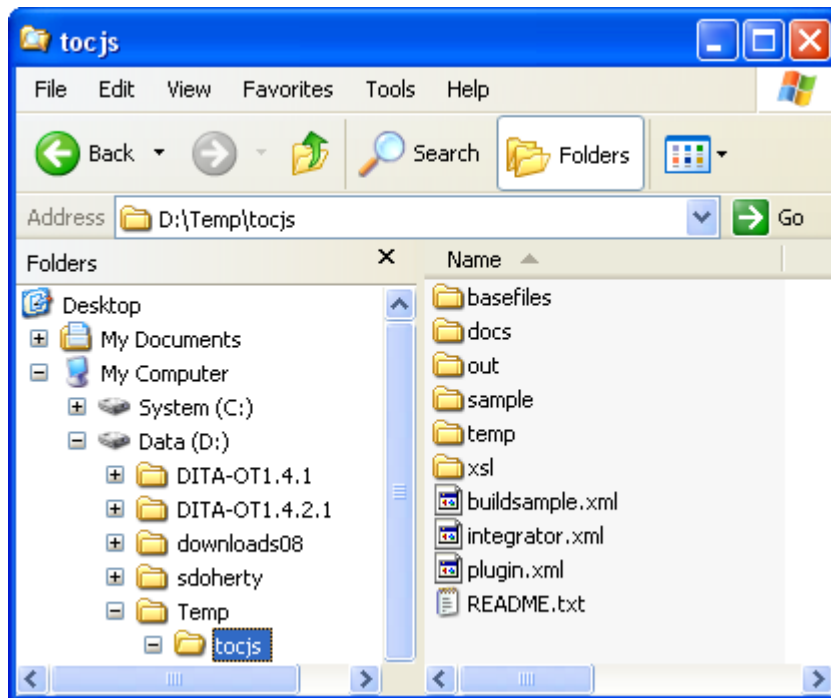


Figure 15: tocjs Installation

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

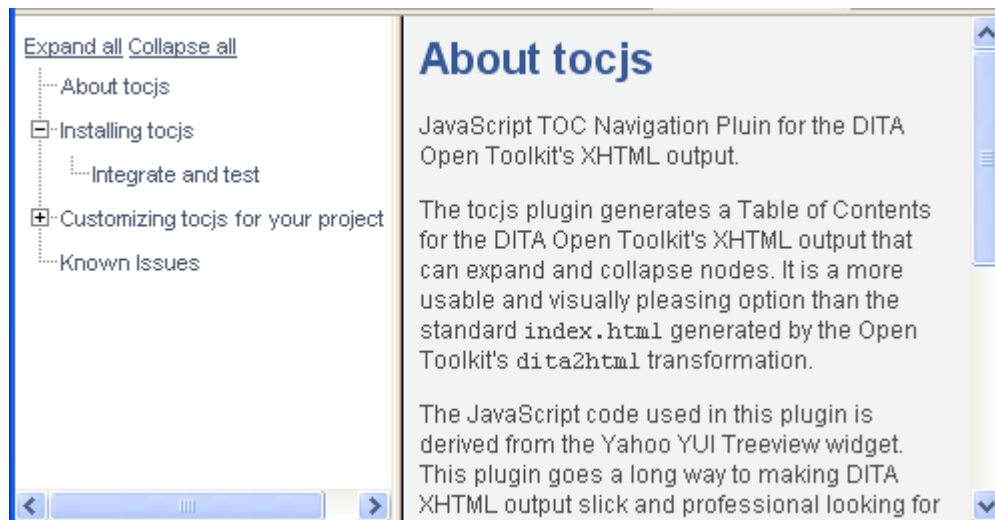
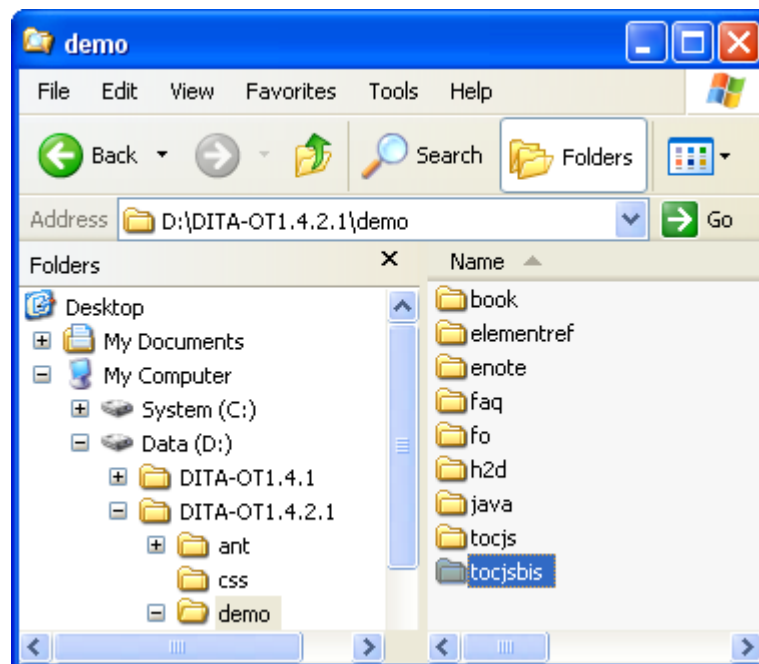


Figure 16: tocjs Documentation

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 plug-in or plug-ins 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/>

Yahoo! UI Library: TreeView

The YUI TreeView Control provides a rich, compact visual presentation of hierarchical node data. Support is provided for several common node types, for the association of custom metadata with each node, and for the dynamic loading of node data (via in-page data or via XMLHttpRequest using [Connection Manager](#)) to navigate large datasets with a small initial payload.



On This Page:

- [Getting Started](#)
- [Using TreeView](#)
- [Known Issues](#)
- [YUI on Mobile Devices](#)
- [Support & Community](#)
- [Filing Bugs and Feature Requests](#)

Quick Links:

- [Examples](#): Explore examples of the TreeView Control in action.

Figure 17: Yahoo UI Library

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 opportunities, 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.

SOPHOS WEB APPLIANCE HELP

About Your WS1000

Organizations typically expend considerable resources and effort preventing virus, worm and trojan infections from entering their networks via email. These same threats, as well as spyware, adware and phishing scams are increasingly infiltrating organizations' networks via web browsing.

Inappropriate web browsing by employees is also a significant legal liability and productivity concern for many organizations. The Sophos Web Appliance provides extensive URL categorization data that allows you to set acceptable web access policies for

Figure 18: Sample tocjs Documentation

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

Contents Search Support

Expand all Collapse all

- About tocjs
- Installing tocjs
- Customizing tocjs for your project
- Known Issues

Figure 19: Integrated tocjsbis

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.

DITA-OT: Eclipse Help

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, Macintosh 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 topics, and organizes the 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 plug-in ownership, or if the contents of the documentation plug-in will be globalized.

Plug-ins are named according to the Sun Java package naming guideline (for example, `org.eclipse.help`). Plug-ins that contain only documentation typically have `.doc` (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 run-time 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, DITA maps, 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 DITA map 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 run-time environment.

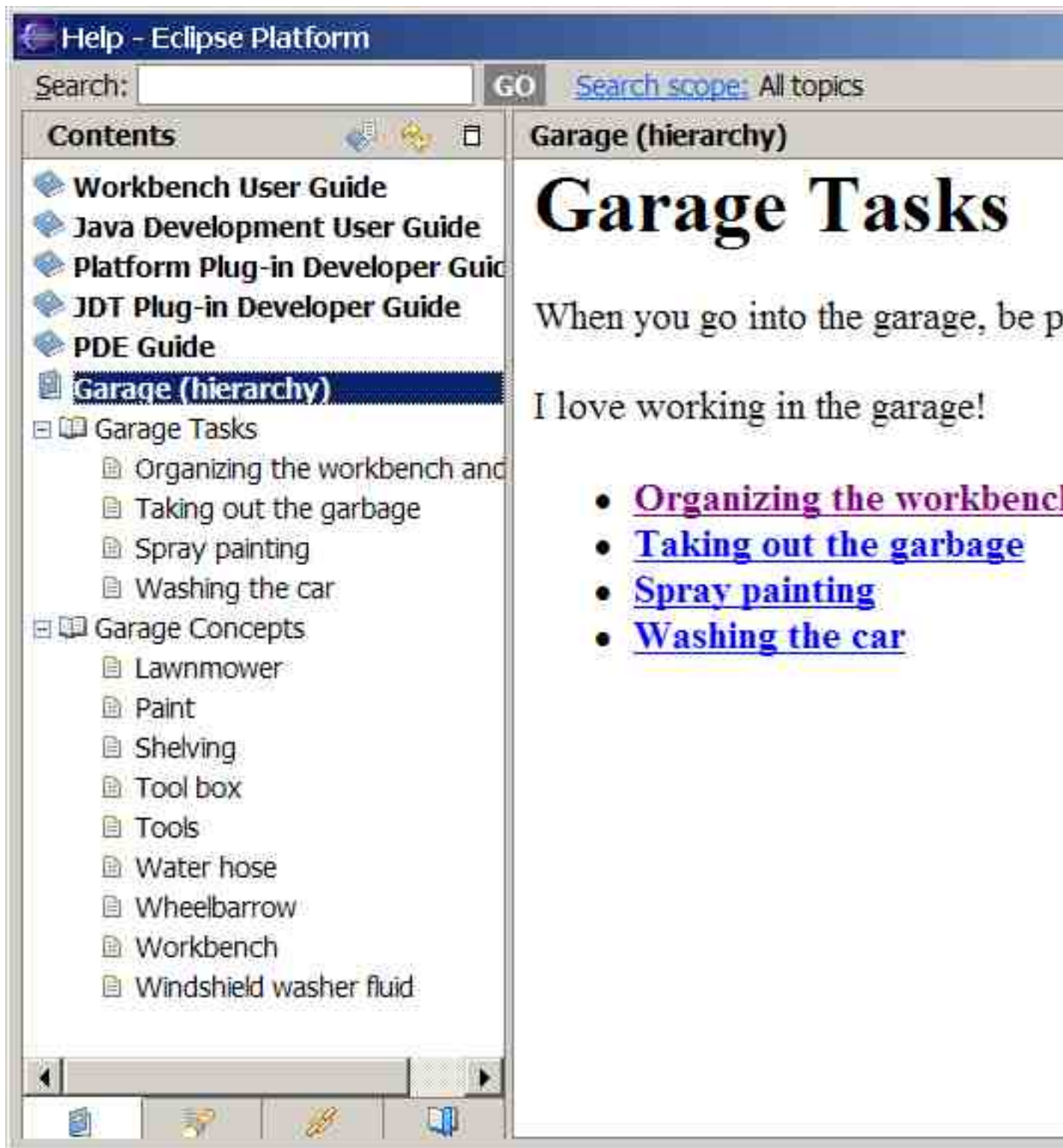


Figure 20: Eclipse Run-time Environment

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 for a standalone Eclipse Help implementation, consider DITA for your 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

IBM Corporation

OASIS DITA Help Subcommittee

CSHelp Plug-in

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 focuses 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 or in the Dynamic Help view. The latter option includes information and functionality in addition to the contents of the Context-Sensitive Help topic.

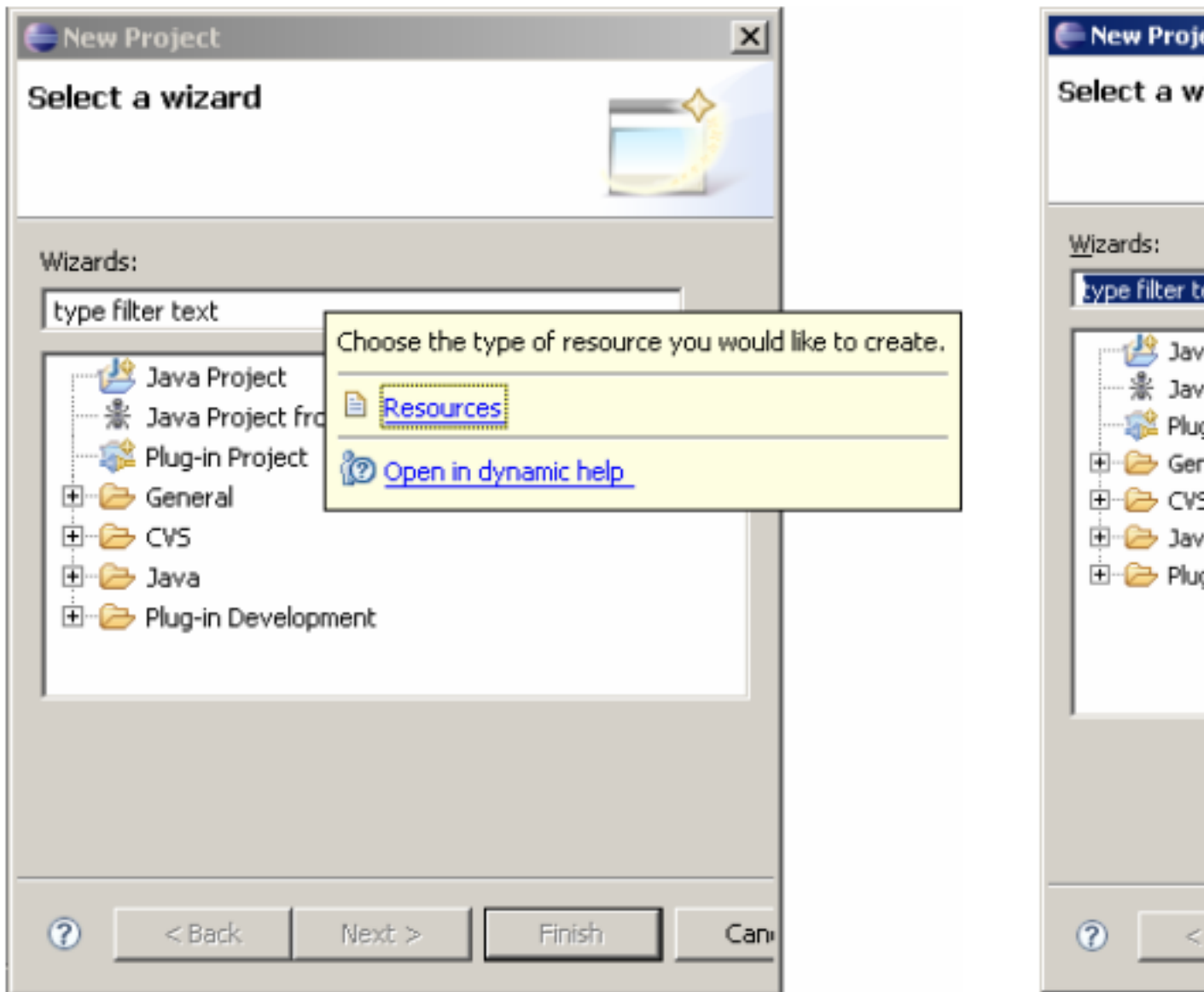


Figure 21: Eclipse User Interface and Help

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 plug-in 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	 

Figure 22: cshelp Plug-in Download

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

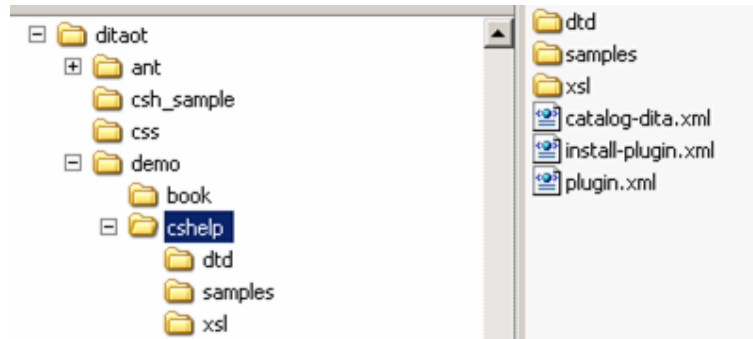


Figure 23: cshelp Installation

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 DITA map file inside the plug-in that points to each of them. Include any copyright information in the DITA map; 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 plug-ins. 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 DITA map 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:

```

<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.outtext" 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 Context-Sensitive Help in DITA allows you to maintain a consistent authoring and build environment and consistently formatted output.

Jeff Antley

IBM Corporation

OASIS DITA Help Subcommittee

Eclipse_CSH Plug-in for Dynamic Context-Sensitive Help

Eclipse plug-ins generated by the DITA-OT, with the `eclipse_csh` plug-in, provide dynamic context-sensitive Help for Eclipse-based applications that define abstract Help contexts and use the `ContextProviderDelegate` (supplied by `org.eclipse.datatools.help`).

Abstract Help contexts allow UI developers to isolate the Eclipse Help system's concrete Help context ID strings and Help search expressions from the UI implementation (Java code). That gives user assistance (UA) content developers more control over Help contexts and context-specific content, so they can provide more useful, and more precisely targeted, dynamic context-sensitive help.

Overview

The Eclipse Help system uses two key pieces of data to find context-specific UA content for dynamic context-sensitive Help:

- A Help context ID string, which associates a UI context with a particular chunk of context-specific content
- A context-specific Help search expression, which the Help system uses to find additional, related content in its online documentation "library"

For many developers, one of the main problems with Eclipse dynamic context-sensitive Help is the necessity to embed Help context ID strings and context-specific Help search expressions in the UI code. That can reduce code portability, and it forces recompiling whenever a Help context ID string or a Help search expression needs to be changed. It makes UI implementation more cumbersome, particularly in an agile development (or RAD) environment. Unless all UI contexts are explicitly specified, and frozen in advance, developers are tempted to delay implementing context-sensitive Help to avoid otherwise unnecessary recompiling and rebuilding.

By using the `org.eclipse.datatools.help.ContextProviderDelegate` (the DTP help-helper, provided by the Eclipse Data Tools Platform project), and defining abstract Help contexts in their code, UI developers are free to implement dynamic context-sensitive Help at any point in the UI development cycle.

By using the DITA-OT, with the `eclipse_csh` plug-in, UA content developers (writers) and information architects can define the associations of Help context IDs with UA content, using special context-sensitive Help markup in a standard DITA map that produces an Eclipse online documentation plug-in. Content developers are free to revise the UI-UA context associations and context-specific Help search expressions at any time, without impacting UI code.

Setup and configuration


The `eclipse_csh` plug-in is available from the DITA-OT project on sourceforge.net: [URL pending final release to open source]

Download the `eclipse_csh` zip file (`eclipse_csh_1.0.0.zip`), and extract all files and folders into the `plugins/` directory, in a previously installed DITA-OT.

Run the DITA-OT `integrator.xml` Ant buildfile to complete the plug-in installation.

Authoring

Any DITA map document that defines an Eclipse online documentation (doc) plug-in can be modified to also define a corresponding context-sensitive Help plug-in, by inserting the appropriate context-related markup.

 **Note:** The DITA map markup described in this section does not require any DITA specializations; it relies entirely on standard DITA map elements and attributes.

Identifying the UI plug-ins associated with Help contexts

DITA maps for context-sensitive Help plug-ins must contain a `<topicmeta>` element as the first child of the `<map>` element, and for each UI plug-in whose Help context IDs are identified in the map, that `<topicmeta>` element must contain one `<othermeta>` element that identifies the UI plug-in.

The `<othermeta>` element's name and content attribute values will be used to identify UI plug-ins in the `org.eclipse.help.contexts` extension, which is declared in the plug-in manifest (`plugin.xml` file) of the context-sensitive Help plug-in. For example:

```
<map id="org.eclipse.datatools.ui.doc">
  <topicmeta>
    ...
    <othermeta name="ui-plugin"
      content="org.eclipse.datatools.connectivity.ui"/>
    <othermeta name="ui-plugin"
      content="org.eclipse.datatools.connectivity.ui.dse"/>
  </topicmeta>
  ...
</map>
```

The name attribute value "ui-plugin" is a fixed, literal string. The content attribute value is the Eclipse plug-in ID of a UI plug-in.

Defining related topics associated with Help contexts

DITA maps for context-sensitive Help plug-ins contain `<resourceid>` elements, each of which identifies a concrete Help context ID associated with a DITA topic.

The `<resourceid>` element is a child of a `<topicmeta>` element. For example:

```
<topicmeta>
  <resourceid id="help_context_ID_string"/>
</topicmeta>
```

For each DITA topic associated with a Help context ID, the `<topicref>` element that points to that topic contains a `<topicmeta>` element (with a `<resourceid>` child element). This defines the association of a Help context ID with that topic. For example:

```
<topicref navtitle="label attribute in contexts.xml topic element"
  href="path/to/topic.xml">
  <topicmeta>
    <resourceid id="help_context_ID_string"/>
  </topicmeta>
</topicref>
```

Any DITA topic can be mapped to multiple Help context IDs by inserting as many `<resourceid>` child elements as necessary in the `<topicmeta>` element. For example:

```
<topicref navtitle="label attribute in contexts.xml topic element"
  href="path/to/topic.xml">
  <topicmeta>
    <resourceid id="help_context_ID_1"/>
    <resourceid id="help_context_ID_2"/>
    <resourceid id="help_context_ID_3"/>
  </topicmeta>
</topicref>
```

Other considerations for DITA maps that define context-sensitive Help plug-ins:

- Markup for context-to-topic mapping is needed only for the DITA topics that will be context-sensitive Help targets (i.e., Related Topics shown in the Help view).

- When a single DITA topic appears more than once in a map, only the first instance of a `<topicref>` that points to that topic needs the context-sensitive Help markup.
- Nested maps that contribute DITA topics related to Help contexts must include the same context-sensitive Help markup.

Context-specific Help content

Each `<topicmeta>` element that wraps a `<resourceid>` element may optionally contain one `<searchtitle>` element or one `<shortdesc>` element, or both (one of each), to provide context-specific Help content.

- The `<searchtitle>` element is used to supply the value of the title attribute on the `<context>` element in the Eclipse context XML file.
- The `<shortdesc>` element is used to supply the content of the `<description>` element in the Eclipse context XML file.

For example:

```
<topicref navtitle="label attribute in contexts.xml topic element"
  href="path/to/topic.xml">
  <topicmeta>
    <searchtitle>Optional text to override the help About \
    title.</searchtitle>
    <shortdesc>Text for context description.</shortdesc>
    <resourceid id="help_context_ID_string"/>
  </topicmeta>
</topicref>
```

- 👉 **Note:** The `<searchtitle>`, `<shortdesc>`, and `<resourceid>` elements must appear in the `<topicmeta>` element in the order shown above.

A DITA map that contains the example shown above produces the following content in an Eclipse context XML file:

```
<contexts>
  ...
  <context id="help_context_ID_string"
    title="Optional text to override the help About title.">
    <description>Text for context description.</description>
    <topic label="label attribute in contexts.xml topic element"
      href="PLUGINS_ROOT/doc_plugin_ID/path/to/topic.xml"/>
  </context>
  ...
</contexts>
```

where `doc_plugin_ID` is the value of the `id` attribute on the `<map>` element.

Integration

To provide dynamic context-sensitive Help, an Eclipse-based application must define the associations between its UI controls and Help contexts dynamically, by implementing methods of `org.eclipse.help.IContextProvider`. One of those methods (`getContext`) must return a concrete Help context ID string, which matches an `IContext` object contributed by an extension of `org.eclipse.help.contexts`, and defined in an Eclipse context XML file. Another method (`getSearchExpression`) must return a context-specific Help search expression, if the application requires more targeted search results than the default help search expression provides.

The DTP help-helper plug-in (`org.eclipse.datatools.Help`) provides a "help key" extension point (`org.eclipse.datatools.help.helpKeyProperties`), and supplies a context provider delegate implementation (`org.eclipse.datatools.help.ContextProviderDelegate`).

- The `helpKeyProperties` extension point allows any plug-in to contribute `ResourceBundle` properties files that define the mapping of abstract Help keys to concrete Help context IDs and Help search expressions.

- The `ContextProviderDelegate`, along with abstract Help keys, enables Help context abstraction for any UI control that implements methods of `IContextProvider`.

The `eclipse_csh` plug-in for DITA-OT provides the processing to generate context-sensitive Help plug-ins, which serve as companions to Eclipse online documentation plug-ins generated by DITA-OT (using its standard `dita2eclipsehelp` transtype). Indeed, the `dita2eclipse_csh` transtype target depends on the `dita2eclipsehelp` target to first generate an Eclipse online documentation plug-in.

Eclipse plug-ins produced by `eclipse_csh` handle the mapping of Help contexts to context-specific Help content, and provide support for the DTP help-helper infrastructure by contributing:

- Eclipse context XML files, which supply the context-specific Help content for each Help context, and point to other Help topic contributions directly related to the Help context
- Java properties files, which define key-value pairs that map abstract Help contexts (`helpKey` constants) to concrete Help context IDs and context-specific Help search expressions

Eclipse plug-ins produced by `eclipse_csh` do not contribute (or contain) any topic-based UA content. Their context XML files refer to topics contributed by their companion documentation plug-ins.

Team collaboration

Successful delivery of Eclipse dynamic context-sensitive Help requires close coordination of UI components and UA components. It imposes responsibilities on both Development teams and Documentation teams, and it requires ongoing collaboration.

Development teams are primarily responsible to:


- Implement the Eclipse classes and methods necessary to enable dynamic context-sensitive Help for all appropriate UI controls.
- Implement interface classes that declare `helpKey` constants for each UI plug-in.
- Provide lists of the `helpKey` constants to Documentation teams in a timely manner.
- Test the context-sensitive Help UI implementation, with context-sensitive Help UA plug-ins and online documentation plug-ins provided by Documentation teams.

Development teams should provide lists of `helpKey` constants in the form of Java source files for the `helpKey` constants interface classes. Java source files provided as `helpKey` lists must include appropriate Help context comments to provide information about each associated UI control.

Documentation teams are primarily responsible to:

- Develop context-specific Help content and context-specific Help search expressions for appropriate UI Help contexts.
- Define the concrete Help context ID strings that associate abstract Help contexts with context-specific Help content.
- Create the Java properties files that define the mapping of abstract Help contexts to concrete Help context IDs.
- Create the context-sensitive Help UA plug-ins that contribute Eclipse context XML files and the Java properties files.
- Test the context-sensitive Help UA plug-ins and online documentation plug-ins, with UI components provided by Development teams.

Documentation teams should rely on the Java source files (for the `helpKey` constants interface classes) as the original and definitive sources of all `helpKey` constant strings.

 **Note:** Creating the Java properties files can be somewhat automated (e.g., by processing the Java source files with a simple PERL script).


Workflow

The following summarizes the Documentation team workflow to create Eclipse context-sensitive Help plug-ins:

1. Get the `helpKey` list (provided as a Java source file) from the UI Development team for each UI plug-in.
2. Analyze the `helpKey` list and associated UI controls to define the Help contexts.


The Documentation team must determine whether:

- The helpKey constants alone are sufficient to identify actual Help contexts, and thus, a helpKey constant could be mapped directly to a concrete Help context ID, with the same string value as the helpKey constant.
- Distinct Help context ID strings must be defined to combine groups of helpKey constants into common Help contexts.

 **Tip:** It may be preferable to combine Help contexts in the helpKey properties file, instead of defining the mapping for multiple Help contexts to a single topic in a DITA map. This is a judgment call for the Documentation team responsible for maintaining DITA maps.

3. Analyze the Help contexts and existing (or planned) Help topics to define context-specific Help search expressions.
4. Create helpKey properties files, based on the content of each Java source file.

- Define the mapping of helpKey constants to concrete Help context IDs and context-specific Help search expressions, based on results of the Help context analysis and Help topic (content) analysis.
- Save the helpKey properties files in source control, as appropriate.

 **Tip:** The helpKey properties files are flat ASCII text files, so authors (or IAs) responsible for defining the Help context IDs and context-specific Help search expressions should use a suitable ASCII text editor to create and edit those files.

5. Modify existing DITA maps (if used to produce Eclipse online documentation plug-ins) to add the markup for context-sensitive help.
6. Build Eclipse plug-ins, using the DITA-OT with the eclipse_csh plug-in, and test the Eclipse plug-ins with UI components provided by Development teams.

Summary

Help context abstraction is a technique to simplify the handling of Help context IDs and Help search expressions in the UI code, by abstracting them to "help keys."

Help context abstraction provides the following benefits:

- Development teams are free to associate new Help contexts with UI controls, without necessitating that corresponding Help context IDs or Help search expressions exist.
- Documentation teams are free to define UA Help contexts, modify Help context IDs, and control the mapping from abstract Help contexts to concrete Help context IDs, without necessitating any change in the UI code.
- Documentation teams are free to define and modify context-specific Help search expressions, and the mapping from abstract Help contexts to Help search expressions, without necessitating any change in the UI code.

This separation of responsibilities enables the project team to provide higher quality, and more precisely targeted, dynamic context-sensitive help.

For more information about the DTP help-helper, see:

http://www.eclipse.org/datatools/doc/20080310_DTP_Help-Helper.pdf.

Dave Resch

Sybase Inc.

OASIS DITA Help Subcommittee

DITA-OT: DITA for Publishers Plug-in

<<need info here>>

DITA-OT: Microsoft HTML Help

One of the standard output formats of the DITA Open Toolkit publishing tools is compiled Microsoft HTML Help, or *CHM files*. This output format is selected through the transformation type of `htmlhelp`.

The topics within Microsoft HTML Help documents are standard HTML files, and as such can be formatted through cascading style sheets (CSS). The DITA Open Toolkit publishing features allow the name of a CSS file to be passed when the DITA source is processed. The nominated CSS file will be attached to each topic in the output, thus controlling the look-and-feel of the output.

Additionally, HTML code to be inserted into the head section of the output topics, and at the top and bottom of the body section, can also be nominated when the DITA source is processed. This feature allows branding and standard text and navigation blocks to be added to output topics, and also opens up the opportunity for JavaScript code to be added to output topics to provide additional scripting functionality.

Context-Sensitive Help Using the Enhanced HTML (`htmlhelp2`) Plug-In

Overview

The process of generating Microsoft HTML Help (CHM) output from DITA content using the DITA Open Toolkit does not provide for storage of Context-Sensitive Help identifiers in the topics or in the DITA map. Context identifiers have to be created and managed outside the DITA authoring environment. The `htmlhelp2` plug-in created by Deborah Pickett extends the DITA OT functionality to allow context-identifiers to be generated from values stored in the `resourceid` attribute of the DITA map's `topicref`.

Setup and configuration

To install the plug-in, download the plug-in ZIP file from [Yahoo! DITA Users Group](#), extract the files into the `\plugins` directory in the DITA-OT directory, and run the integrator task. The `htmlhelp2` transtype will then be available as an Ant build output.

The `htmlhelp2` functionality is built into the `ditaplus` plug-in, also created by Deborah Pickett.

Authoring

The plug-in uses the `resourceid` element within the `topicmeta` element in the `topicref` in the DITA map as the basis for building the map and alias files required for HTML Help Context-Sensitive Help. The `resourceid` element doesn't contain data, and has two attributes: `id` and `appname`. The `id` attribute can hold only one value, whereas the context-hooks in HTML Help require a string and a numeric identifier. It is expected you will enter a numeric value into the `id` attribute of the `resourceid`, and the plug-in will derive from that number a string value.

For example, an `id` of 3254 will be transformed to a map file entry of:

```
#define identity_3524 3524
```

In other words, the `resourceid`'s `id` attribute is used as the context number, and used to create the context string by prefixing the number with the string `identity_`.

In addition to entering a context number in the `id` attribute, you must also enter an `appname` attribute of `WindowsHelpId` in the `resourceid` element in the `topicref` in the DITA map. The `WindowsHelpId` value identifies the particular `resourceid` as being used for Context-Sensitive Help, thus differentiating it from `resourceid` elements used for other purposes.

You can have more than one `resourceid` element if you need several different help IDs to be pointed to the one topic.

How It Works

In Microsoft HTML Help, map (.h or .map) files correlate context strings to context numbers, and alias (.ali) files correlate context numbers to topic file names.

The `htmlhelp2` transformation extracts from all topics in the DITA map any `resourceid` elements where the `appname` attribute is `WindowsHelpId`. The map and alias files generated by `htmlhelp2` are named `[map_file_name].map` and `[map_file_name].ali` respectively.

The plug-in has another useful feature. When it generates the HTML Help project (.hhp) file, it adds window definitions named `default`, `global_Standard` and `global_Dialog`.

The latter window definition will displays as a single pane in the right third of the window.

Example

The following markup:

```
<map>
<title>Cat flossing </title>
<topicref href="abc.xml">
  <topicmeta>
    <resourceid appname='WindowsHelpId' id='3524' />
  </topicmeta>
</topicref>
...
</map>
```

will produce the following map and alias file lines:

- map:
#define identity_3524 3524
- alias:
identity_3524 abc.html

Summary

The `htmlhelp2` plug-in provides a simple method for generating HTML Help context map and alias files from values entered in the `resourceid` element in the DITA map's `topicref`.

Tony Self

HyperWrite Pty. Ltd.

Chairperson, OASIS DITA Help Subcommittee

Context-Sensitive HTML Help Using the The DITA Open Toolkit

We would be remiss not to call attention to the original HTML Help transform that has been bundled with the DITA Open Toolkit for several years.

Overview


Output from the DITA Open Toolkit HTML Help plug-in serves as input to the Microsoft HTML Help Compiler. If you choose to build your DITA sources on the same Windows system on which you have installed a copy of the Microsoft HTML Help Software Development Kit, you will be able to generate a ready-to-deploy MS HTML Help .chm library from your DITA-OT ant script. Otherwise, you can copy output from the DITA Open Toolkit HTML Help transform on your DITA-OT build system to some other Windows system on which the HTML Help SDK has been installed.

Setup and configuration

Once you have installed and configured the DITA Open Toolkit, you have the HTML Help transform installed by default and ready to use. See the *DITA Open Toolkit User's Guide* for detailed information about working with transforms and about specifying the HTML Help transform specifically as a processing target (dita2htmlhelp).

To download the latest Microsoft HTML Help 1.4 SDK, go to the following MSDN web site. It is a free download.

<http://msdn.microsoft.com/en-us/library/ms669985.aspx>

 **Note:** If you run the test build script that is bundled with the Open Toolkit, `sample_all.xml`, you can verify that the HTML Help transform is installed and configured correctly by looking in the `ant/out/samples/htmlhelp` subdirectory.

Authoring

To generate ready-to-deploy `.chm` libraries from most any DITA source collection, you do not really have to customize or rework your DITA source files.

DITA source	MS HTML Help output
DITA map entries	Contents tab entries
index entries	Index tab entries
running text	full-text search index

Integration

Truth be known, there are also limitations to both this DITA-OT transform and to the MS HTML Help run-time environment. Factor these into your planning.

- *Context-sensitive mapping:* The DITA-OT HTML Help transform is not designed to process context-sensitive mappings between DITA source topics and the context IDs associated with those target topics. To feed the MS HTML Help compiler with the appropriate context-sensitive ID maps and/or alias files, consider developing a custom shell for the Open Toolkit or building your HTML Help output with WinANT. See [#unique_53](#) for more information about the latter.
- *HTML support in MS HTML Help:* The Microsoft HTML Help application displays HTML topics in a captive HTML viewer. This viewer supports most, but not all, of the HTML output that you can generate from the DITA-OT XHTML transform. If, for example, you are working with SVG images or some interactive JavaScript libraries, test them with output from the HTML Help transform before deploying them.

Output

The `.chm` library produced by the combination of the DITA-OT HTML Help transform and the MS HTML Help compiler is very functional.

Here's what output from the DITA-OT test collection looks like.

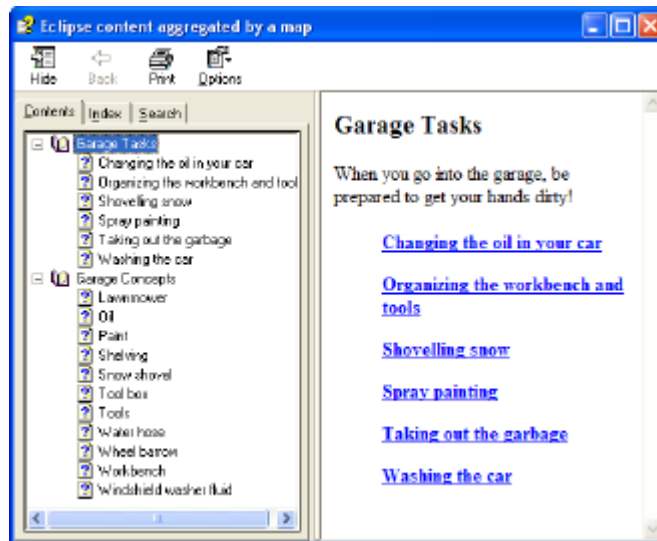


Figure 24: Garage Tasks Contents Tab in HTML Help

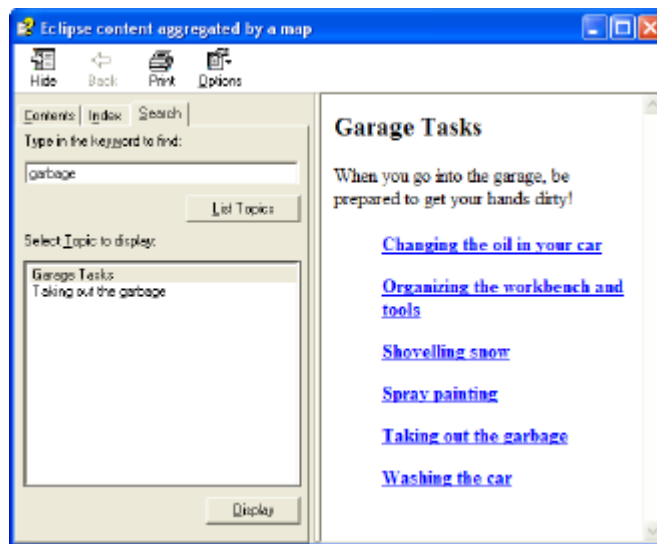



Figure 25: Garage Tasks Search Tab in HTML Help

 **Note:** The DITA-OT test collection source files contain no index entries, so no index entries are generated in MS HTML Help.

Summary

If you have requirements to deploy a general-purpose Help collection to Windows customers *without* context-sensitive integration, consider using the DITA-OT HTML Help transform. It is functional, reliable and fast. Once all your content in is .chm format, you can convert it to other distribution formats with second-tier conversion tools such as [chm2web](#). If you have requirements to integrate .chm output with Windows applications as Context-sensitive Help, you'll need to invest some time managing maps and alias files outside the DITA environment and then integrating them with .chm output using WinAnt or other custom processing environments.

DITA-OT: Leximation AIR Help Plug-in

The Leximation AIR Help DITA-OT plug-in is not yet publicly available; to request a Beta version of this plug-in, please visit www.leximation.com/airhelp.

Overview

“AIR Help” is a term that refers to any user assistance application that has been developed using Adobe’s AIR (Adobe Integrated Runtime) technology and does not specifically define appearance or functionality. Applications developed using AIR can be installed on the Mac, Windows, and Linux operating systems, and are highly customizable, making it an ideal technology from which to develop an online Help deliverable.

Currently there are two commercially available products for generating AIR Help, Adobe’s “RoboHelp 8” and MadCap Software’s “Flare 4.” Adobe also offers a utility called “RoboHelp Packager for AIR” which allows you to generate an AIR Help file from a RoboHelp WebHelp project (for use with older versions of RoboHelp). RoboHelp 8 allows you to import DITA files which can then be exported to an AIR Help package.

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 Leximation AIR Help DITA-OT plug-in 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 AIR Help interface to add features or to modify the appearance as needed.

Setup and configuration

A new feature in the DITA-OT 1.5 allows for external hooks into the indexing pipeline making it possible to generate the alternate format index required by this plug-in (in earlier versions a recompile of `dost.jar` was required).

In order to compile and package up an AIR file, you must have the Flex and AIR SDKs installed (both are freely available from Adobe). If you have Flex Builder 3 installed, both of these SDKs are included in the installation.

To install this plug-in:

1. Extract the ZIP archive into the `DITA-OT/demo` directory to create an `lmi-airhelp` directory which contains all of the necessary files.
2. If you have Flex Builder 3 installed, continue to the next step, otherwise download and install the following:
 - Adobe Flex 3 SDK (<http://www.adobe.com/products/flex/flexdownloads/>)
 - Adobe AIR 1.5 SDK (<http://www.adobe.com/products/air/tools/sdk/>)
3. Locate the `lmi-airhelp/integrator.xml` file. Modify the property values so they are correct for your system. In particular, the `FLEX.SDK_HOME` and `AIR.SDK_HOME` properties, but possibly others. Search for “AIRHELP PROPERTIES.”
4. Add the `lmi-airhelp.jar` file to the `CLASSPATH`. If you're using the `startcmd.bat` file to set up the environment, add `%DITA_DIR%demo\lmi-airhelp\lmi-airhelp.jar;` to the `CLASSPATH` declaration:


```
set CLASSPATH=%DITA_DIR%demo\lmi-airhelp\lmi-airhelp.jar; ...
```
5. Integrate the plugin with the OT. Run the integration command:


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

After installation, you can test the install by copying the `sample_airhelp.xml` file into the `DITA-OT/ant/` directory. Run the `startcmd.bat` file and enter the following:

```
ant -f ant/sample_airhelp.xml
```

Assuming the build is successful, you should end up with a `DITA-OT/ant/out/airhelp/` directory. In that directory you'll see an `airhelp-test.bat` file. Run the batch file to launch the new AIR Help file. Test the index and search to make sure that all's well.

In order to build an AIR file that can be sent to others, run the `airhelp-packager.bat` file. The packager batch file assumes that you have set up the digital certificate information properly in the `integrator.xml` file. For information on creating a self-signed certificate, see http://help.adobe.com/en_US/AIR/1.5/devappshtml/WS5b3ccc516d4fbf351e63e3d118666ade46-7f74.html

The plug-in creates the following files:

- TOC file (`<mapname>.ahtoc`)
- Keyword index file (`<mapname>.ahix`). Displays on the Index tab and is built from available `indexterm` elements.
- “Local” full text search index (`_searchIndex.xml`). Displays on the Search tab and is created by parsing the generated HTML files.
- The AIR Help project file (`airhelp.ahp`)
- The AIR application file (`airhelp-app.xml`)
- Compiled SWF (`airhelp.swf`)
- Batch files for testing and packaging of the AIR file (`airhelp-test.bat` and `airhelp-packager.bat`)

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

- `/map/topicmeta/copyright` or `/bookmap/bookmeta/bookrights`
- `/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)

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

No specific authoring requirements are needed.

Integration

Because an AIR Help file is actually a desktop application, you launch the Help system by making a system call to run the executable. If no command-line parameters are provided, the AIR Help file will open with the default topic displayed.

In order to launch the Help system on a specific topic, use the following command-line syntax:

```
<path/app_executable> <target_topic>
```

For example, the following call will display the “Image” topic in the DITA Reference AIR Help file:

```
"c:\program files\ditaref\ditaref.exe" image.html
```

To launch the Help system with the search panel selected and pre-populated with a query, use the following syntax:

```
<path/app_executable> "search:using conrefs"
```

To launch the Help system with an external URL (website) displayed, use the following syntax:

```
<path/app_executable> http://google.com
```

Additional command-line parameters can be added as needed.

Because only one instance of each AIR Help application can exist at one time, if you make a system call to the application when one is already running, the result of the new call replaces the running instance.

Output

This plug-in uses the HTML generated by the dita2html target, and only controls the interface features of the “container” application.

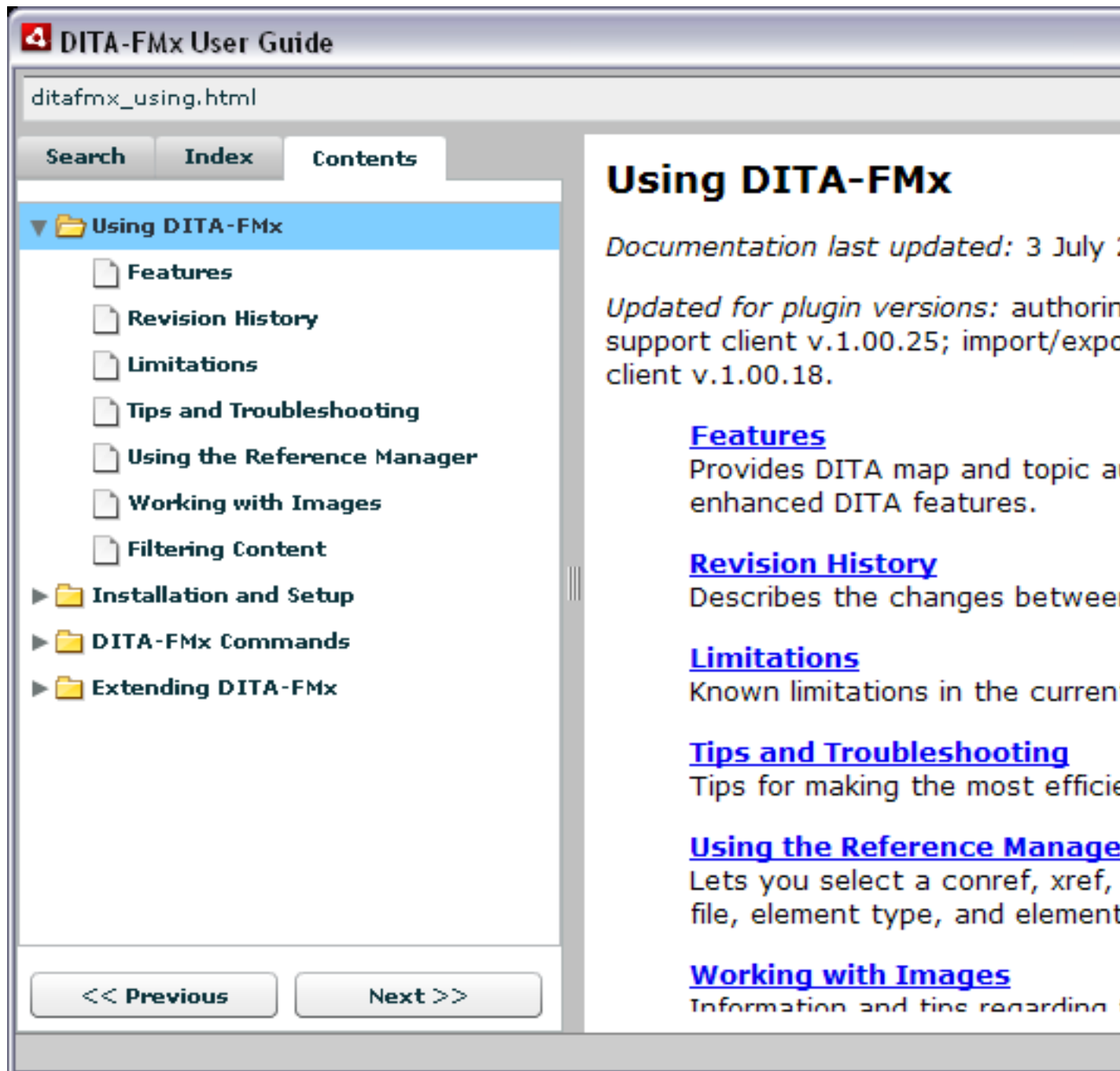


Figure 26: Example AIR Help application window

Summary

Because AIR Help is very new and still under development, it may be best for some to wait for further development to be completed. However, because of its cross-platform support and reduced testing effort, you may find that it is worth closer investigation as a potential Help delivery format.

Scott Prentice

Individual

OASIS DITA Help Subcommittee

Adobe FrameMaker

<<info here>>

<<PDF PUBLISHING>>

Adobe RoboHelp

<<info here>>

Converting DITA Content to *WebHelp* using RoboHelp

The DITA Open Toolkit provides only rudimentary support for tri-pane, HTML-based documents. Help Authoring Tools such as RoboHelp have long set the benchmark for feature-rich, HTML-based documents. Adobe describes the RoboHelp tri-pane HTML output as *WebHelp*.

1. Generate HTML Help output from your DITA content using the DITA Open Toolkit.
2. Start RoboHelp.
3. From the **File** menu, choose **New Project**.
The **New Project** window displays.
4. Select the **Import** tab, select **Microsoft HTML Help Project (*.hhp)**, and click **OK**.
The **Select HTML Help Project** dialog box displays.
5. Browse to the directory into which the DITA Open Toolkit generated your DITA document's output, select the *project (.hhp)* file, and click **Open**.
The HTML Help project files will be imported into a new RoboHelp project.
6. Generate the WebHelp output.

RoboHelp will generate the WebHelp output into the `/!SSL!/WebHelp` subdirectory of the project path. The default file name to launch the WebHelp document takes its name from the project file name.

Tony Self

HyperWrite Pty. Ltd.

Chairperson, OASIS DITA Help Subcommittee

DITA Exchange

<<info here>>

JustSystems XMetaL

<<info here>>

<< NICE WEBHELP OUTPUT >>

MadCap Blaze

<<info here>>

<< PDF PUBLISHING OPTION >>

MadCap Flare

<<info here>>

Omni Systems DITA2Go

<<info here>>

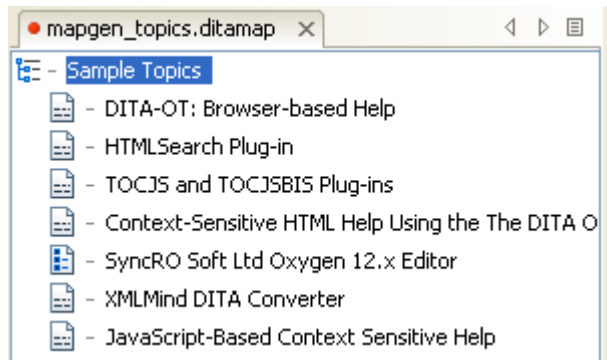
SyncRO Soft Ltd Oxygen 12.x Editor

SyncRO Soft Ltd bundles a copy of the DITA Open Toolkit with its Oxygen XML Editor 12.x, allowing you to transform content in your DITA projects into any of the following Help output formats.

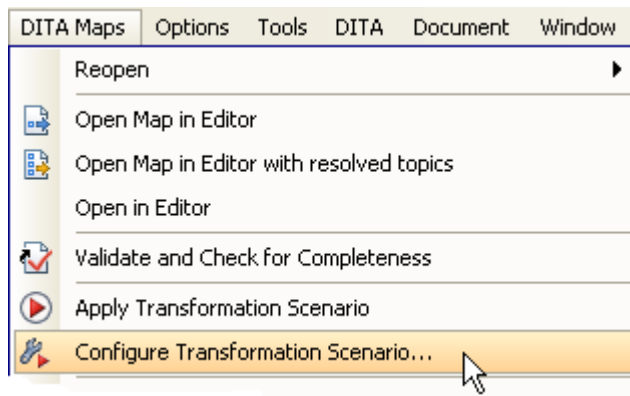
Output Format	Description
EclipseHelp	SyncRO provides basically the same transformation options into EclipseHelp as the DITA Open Toolkit 1.4 and above.
Generic XHTML	The generic XHTML output supports the same transformation switches and options as those in the DITA Open Toolkit 1.4 and above.
JavaHelp	The JavaHelp output from Oxygen is equivalent to that in the DITA Open Toolkit 1.4 and above.
Microsoft HTMLHelp	The compiled XHTML output from this MS HTMLHelp transformation (*.chm) is equivalent to what the DITA Open Toolkit 1.4 and above provides. It does not integrate the additional functionality available in the CSHelp plug-in.
WebHelp	The WebHelp output from Oxygen actually generates a classical tripane Help package that builds upon basic XHTML output. This output format is not available currently in the DITA Open Toolkit.

To generate one or more of these Help output types from within the Oxygen 12.x XML Editor, do the following.

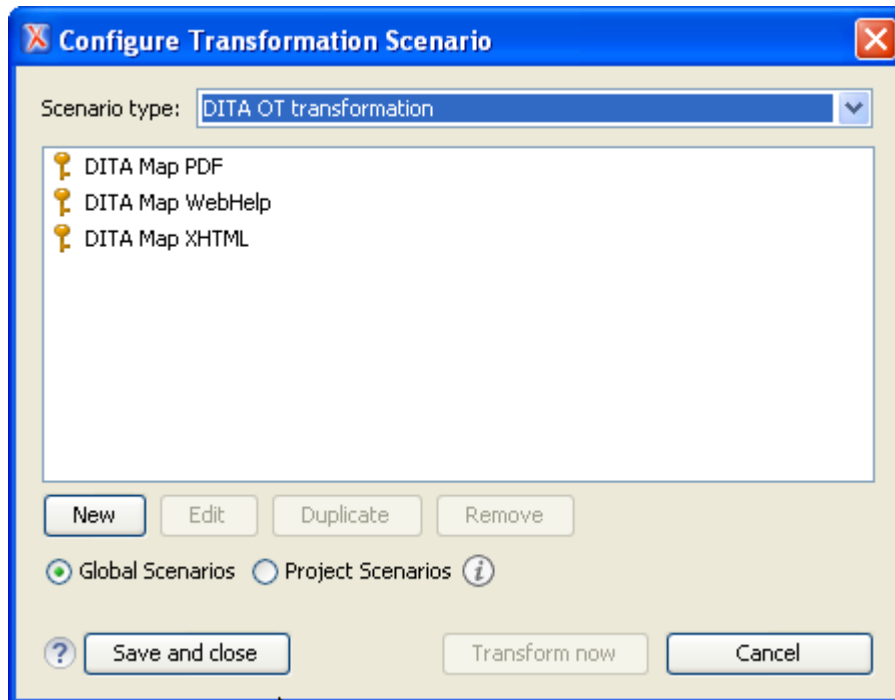
1. In Oxygen, open the top-level DITA map document that you want to transform into one or more Help output formats. For example, if you were to open a DITA map named `mapgen_topics.ditamap`, Oxygen would display it in the DITA Maps Manager pane.



- From main menu in Oxygen, choose **DITA Maps -- Configure Transformation Scenario**.

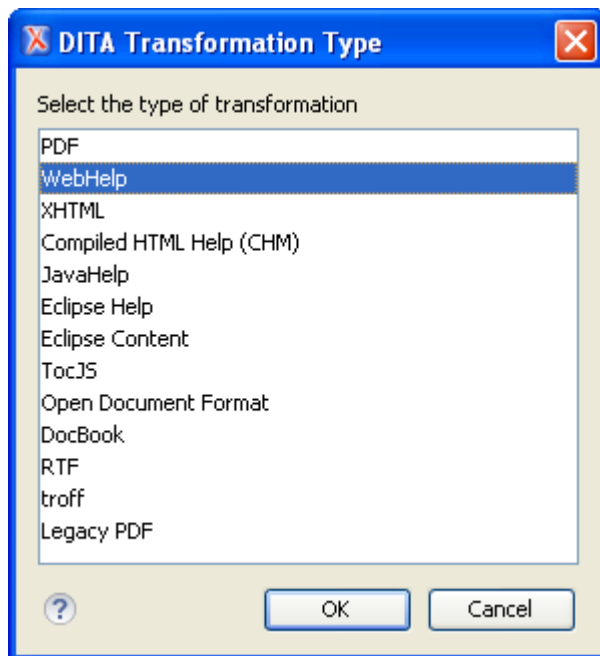


Oxygen displays a dialog box similar to the following.



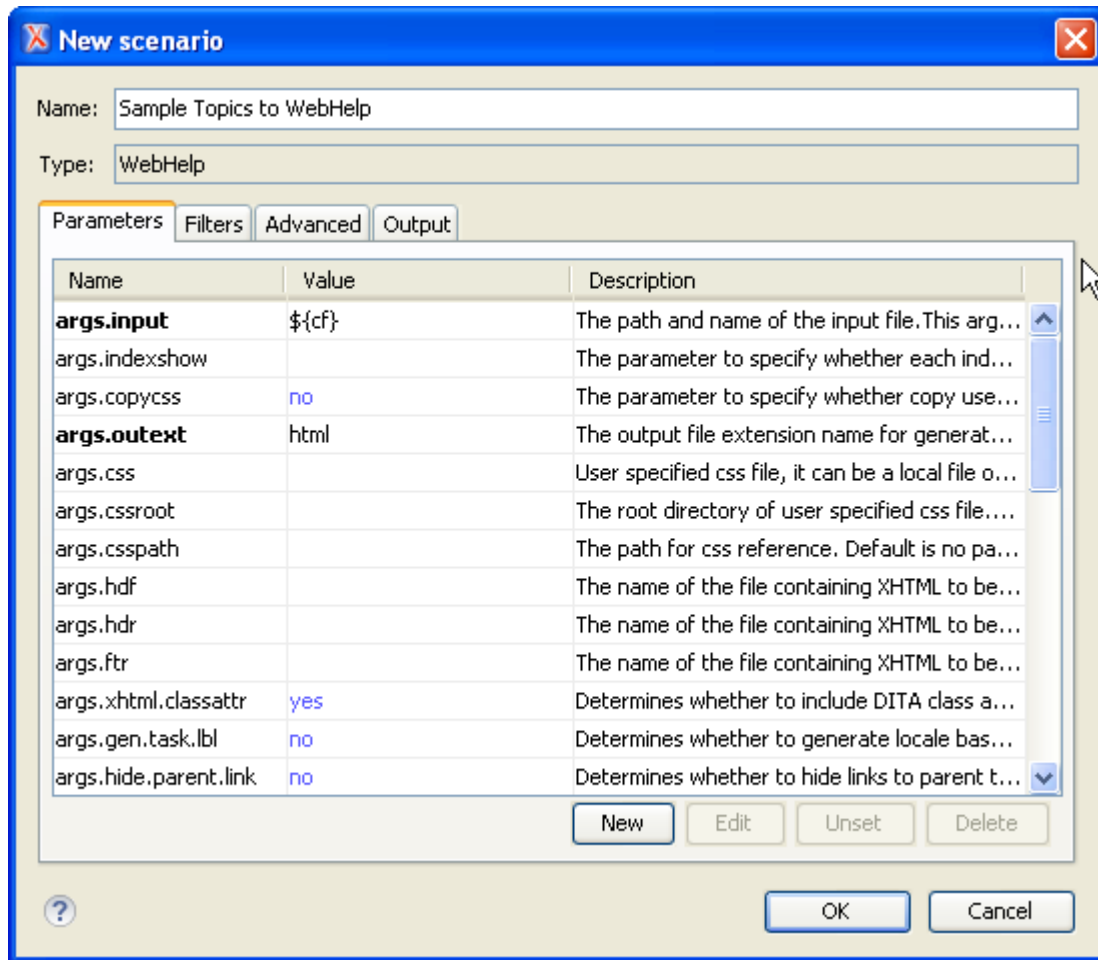
3. Click the **New** button to create a map-specific transformation profile for the DITA map document currently in the editor.

Oxygen displays a dialog box similar to the following.



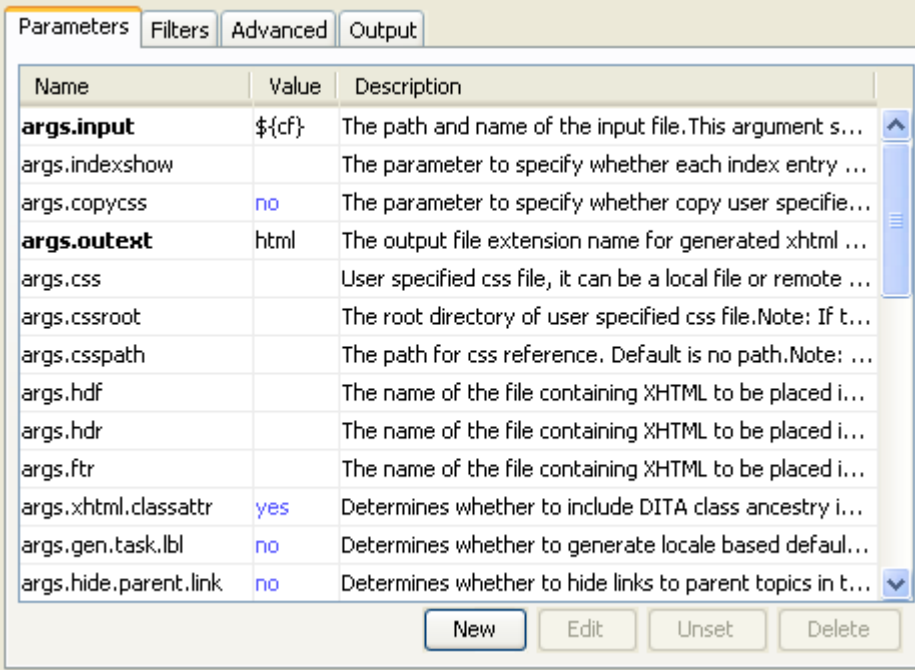
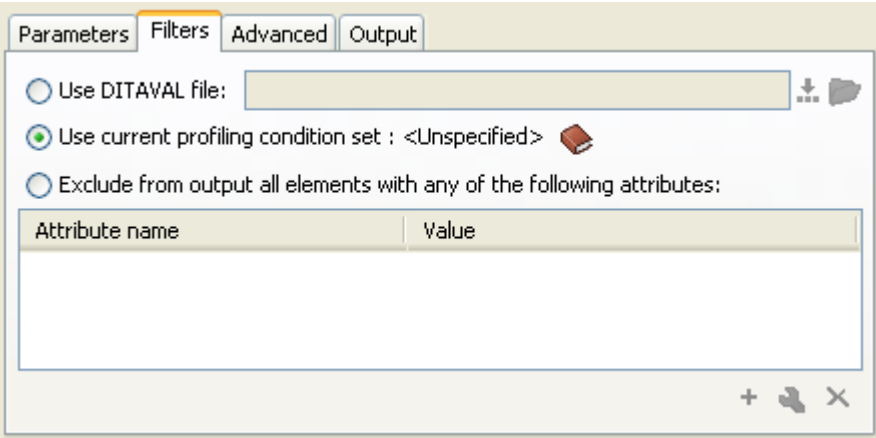
4. Choose a Help transformation type, for example `WebHelp`, and click **OK**.

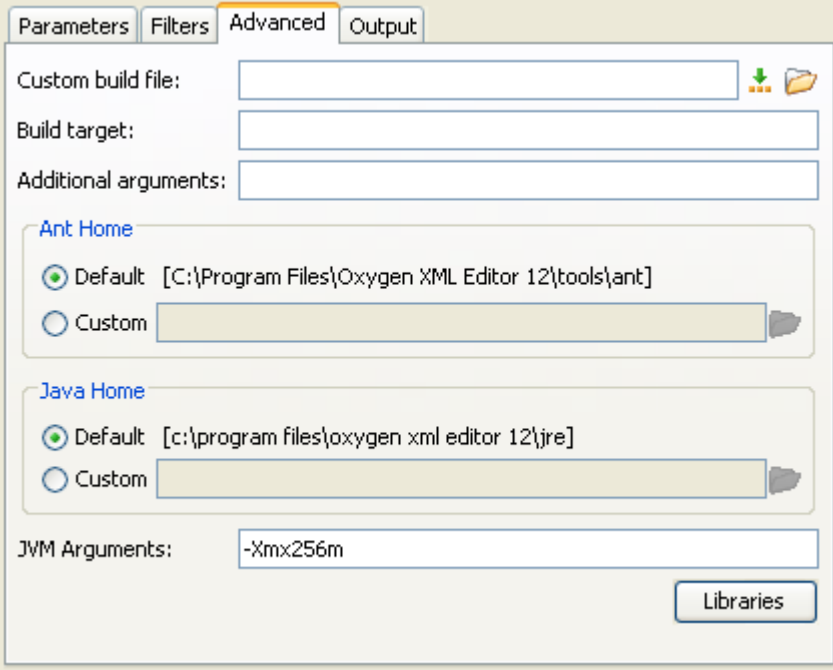


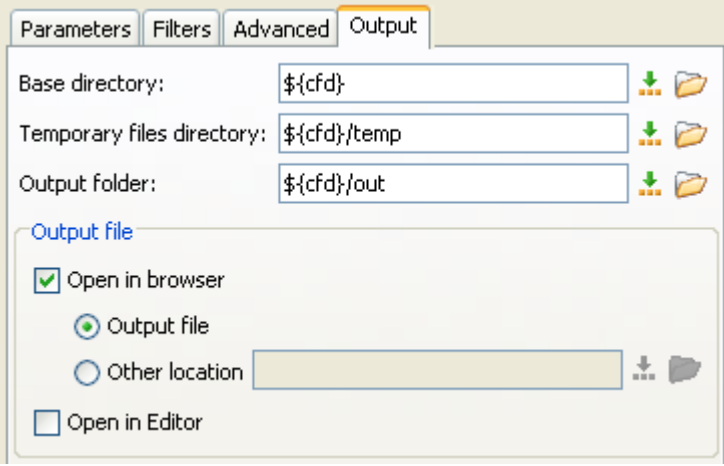








Oxygen displays a dialog box similar to the following.



- Specify a name for this transformation profile, for example `Sample Topics to WebHelp`, and (optionally) set other flags in the tabbed settings panes.

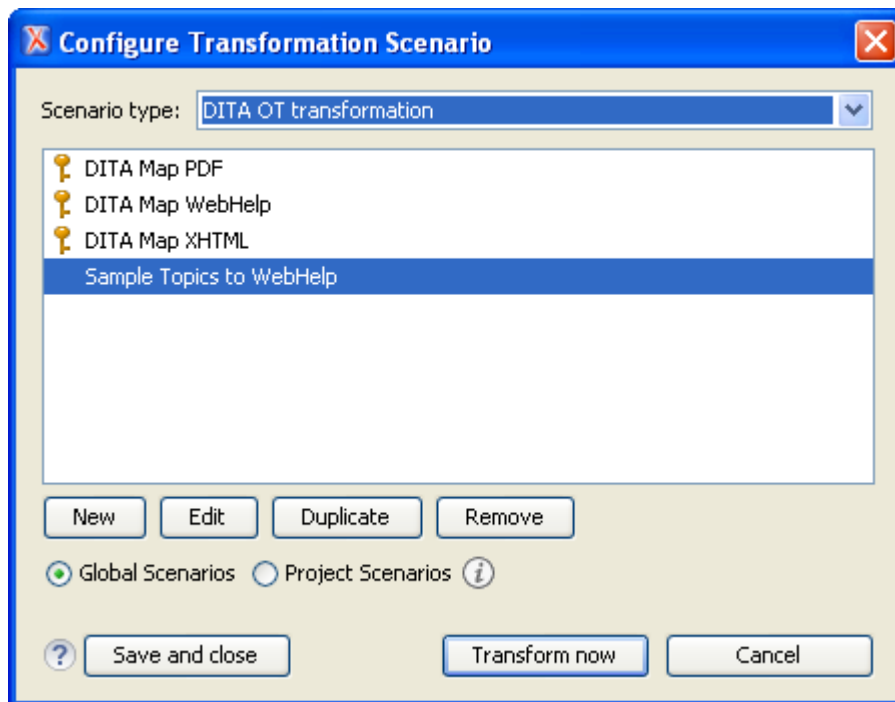
Tab	Options
Parameters	Specifies DITA Open Toolkit ant parameters for this transformation type.

Tab	Options																																										
	 <p>Parameters Filters Advanced Output</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>args.input</td> <td><code>\${cf}</code></td> <td>The path and name of the input file. This argument s...</td> </tr> <tr> <td>args.indexshow</td> <td></td> <td>The parameter to specify whether each index entry ...</td> </tr> <tr> <td>args.copycss</td> <td>no</td> <td>The parameter to specify whether copy user specifie...</td> </tr> <tr> <td>args.outtext</td> <td>html</td> <td>The output file extension name for generated xhtml ...</td> </tr> <tr> <td>args.css</td> <td></td> <td>User specified css file, it can be a local file or remote ...</td> </tr> <tr> <td>args.cssroot</td> <td></td> <td>The root directory of user specified css file. Note: If t...</td> </tr> <tr> <td>args.csspath</td> <td></td> <td>The path for css reference. Default is no path. Note: ...</td> </tr> <tr> <td>args.hdf</td> <td></td> <td>The name of the file containing XHTML to be placed i...</td> </tr> <tr> <td>args.hdr</td> <td></td> <td>The name of the file containing XHTML to be placed i...</td> </tr> <tr> <td>args.ftr</td> <td></td> <td>The name of the file containing XHTML to be placed i...</td> </tr> <tr> <td>args.xhtml.classattr</td> <td>yes</td> <td>Determines whether to include DITA class ancestry i...</td> </tr> <tr> <td>args.gen.task.lbl</td> <td>no</td> <td>Determines whether to generate locale based default...</td> </tr> <tr> <td>args.hide.parent.link</td> <td>no</td> <td>Determines whether to hide links to parent topics in t...</td> </tr> </tbody> </table> <p>New Edit Unset Delete</p>	Name	Value	Description	args.input	<code>\${cf}</code>	The path and name of the input file. This argument s...	args.indexshow		The parameter to specify whether each index entry ...	args.copycss	no	The parameter to specify whether copy user specifie...	args.outtext	html	The output file extension name for generated xhtml ...	args.css		User specified css file, it can be a local file or remote ...	args.cssroot		The root directory of user specified css file. Note: If t...	args.csspath		The path for css reference. Default is no path. Note: ...	args.hdf		The name of the file containing XHTML to be placed i...	args.hdr		The name of the file containing XHTML to be placed i...	args.ftr		The name of the file containing XHTML to be placed i...	args.xhtml.classattr	yes	Determines whether to include DITA class ancestry i...	args.gen.task.lbl	no	Determines whether to generate locale based default...	args.hide.parent.link	no	Determines whether to hide links to parent topics in t...
Name	Value	Description																																									
args.input	<code>\${cf}</code>	The path and name of the input file. This argument s...																																									
args.indexshow		The parameter to specify whether each index entry ...																																									
args.copycss	no	The parameter to specify whether copy user specifie...																																									
args.outtext	html	The output file extension name for generated xhtml ...																																									
args.css		User specified css file, it can be a local file or remote ...																																									
args.cssroot		The root directory of user specified css file. Note: If t...																																									
args.csspath		The path for css reference. Default is no path. Note: ...																																									
args.hdf		The name of the file containing XHTML to be placed i...																																									
args.hdr		The name of the file containing XHTML to be placed i...																																									
args.ftr		The name of the file containing XHTML to be placed i...																																									
args.xhtml.classattr	yes	Determines whether to include DITA class ancestry i...																																									
args.gen.task.lbl	no	Determines whether to generate locale based default...																																									
args.hide.parent.link	no	Determines whether to hide links to parent topics in t...																																									
Filters	<p>Specifies filtering options compatible with the DITA Open Toolkit (DITAVAL) or specific to the Oxygen Editor.</p>  <p>Parameters Filters Advanced Output</p> <p><input type="radio"/> Use DITAVAL file: <input type="text"/></p> <p><input checked="" type="radio"/> Use current profiling condition set : <Unspecified></p> <p><input type="radio"/> Exclude from output all elements with any of the following attributes:</p> <table border="1"> <thead> <tr> <th>Attribute name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td> </td> <td> </td> </tr> </tbody> </table> <p>+ [icon] X</p>	Attribute name	Value																																								
Attribute name	Value																																										
Advanced	<p>Specifies transformation-specific options and alternate locations for ant and JAVA binary files.</p>																																										

Tab	Options
	 <p>Parameters Filters Advanced Output</p> <p>Custom build file: <input type="text"/>  </p> <p>Build target: <input type="text"/></p> <p>Additional arguments: <input type="text"/></p> <p>Ant Home</p> <p><input checked="" type="radio"/> Default [C:\Program Files\Oxygen XML Editor 12\tools\ant]</p> <p><input type="radio"/> Custom <input type="text"/></p> <p>Java Home</p> <p><input checked="" type="radio"/> Default [c:\program files\oxygen xml editor 12\jre]</p> <p><input type="radio"/> Custom <input type="text"/></p> <p>JVM Arguments: <input type="text" value="-Xmx256m"/></p> <p><input type="button" value="Libraries"/></p>
Output	<p>Specifies output display, naming, and processing options.</p>  <p>Parameters Filters Advanced Output</p> <p>Base directory: <input type="text" value="\${cfid}"/>  </p> <p>Temporary files directory: <input type="text" value="\${cfid}/temp"/>  </p> <p>Output folder: <input type="text" value="\${cfid}/out"/>  </p> <p>Output file</p> <p><input checked="" type="checkbox"/> Open in browser</p> <p><input checked="" type="radio"/> Output file</p> <p><input type="radio"/> Other location <input type="text"/>  </p> <p><input type="checkbox"/> Open in Editor</p>

- Click **OK** when you have completed editing the profile.

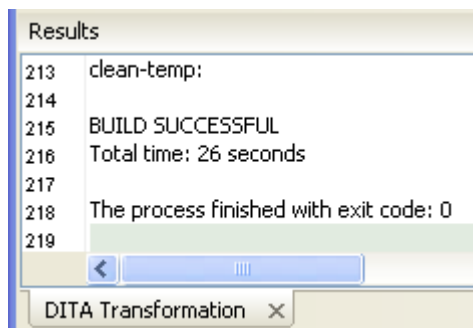
Oxygen adds this new profile to the list of available transformation profiles.



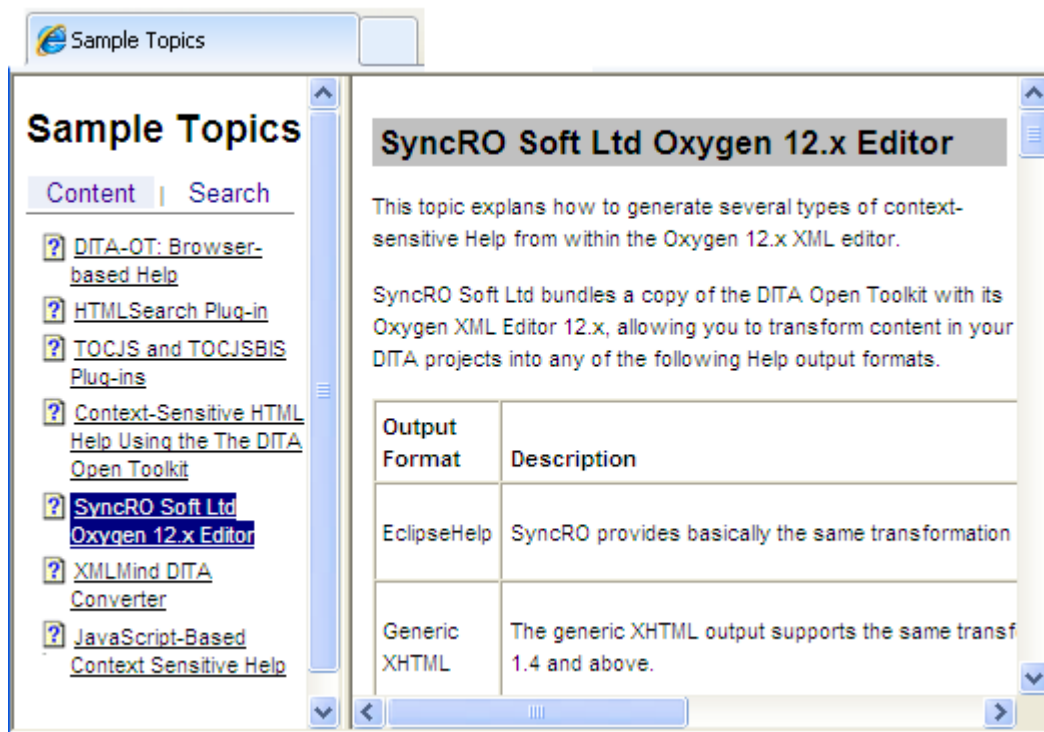
- Choose the new profile name and click **Transform now** to transform the specified DITA map into the target Help output type.

Note: Once you have created a transformation profile, you do not need to have the specified DITA map open in your editor. Oxygen automatically opens the target DITA map and transforms it per the profile.

Oxygen begins the transformation and displays build processing information in a separate pane.



For HTML-based transformations, Oxygen typically launches a browser and displays the transformed output in it.



For more information about configuring Oxygen to generate other types of output, see the Oxygen Help system.

PTC Arbortext Digital Media Publisher

Arbortext Digital Media Publisher (DMP) has been available as a standalone product for some time, but in the upcoming release DMP provides a DITA-based integration with Arbortext Editor that enables you to create a Help system or other online information system directly from a DITA map.

Overview

DMP enables you to take a set of documents and compose them to various types of output. The documents can be of several different types including DITA and other types of XML documents, HTML, PDF, Microsoft Word, Microsoft PowerPoint, Microsoft Excel, Javadoc, or just text documents. Once you have compiled your set of content, you can use DMP to compose that content to the following types of output:

- Help systems
- Standalone knowledge bases or information systems
- Web applications

DMP is a browser-based application and uses either an embedded browser (based on Internet Explorer) or the native browser installed on a system to serve up the content. DMP is a cross-platform application and runs on Windows, Linux, and some types of UNIX. DMP also supports multiple languages in the same image.

PTC offers more information on DMP at the [PTC web site](#).

Setup and configuration

DMP can be purchased from PTC, either as a standalone product or bundled with Arbortext Publishing Engine. DMP comes with its own installation program. To create a DMP image directly from a DITA map also requires Arbortext Editor.

You can configure a DMP image in the following ways:

- DMP as a standalone product

DMP has some Java property files that you use to set up the application properties and a content configuration XML file that you use to identify both the content sets to go in the application and the structure of the table of contents. You can also apply metadata to the table of contents to filter the content displayed in the image based on any criteria you define. Once you have completed setting up your configuration files and content sets, you can run a DMP build to create the DMP image.

- DMP in conjunction with Arbortext Editor

You can use Arbortext Editor to edit a specialized DITA map called a “DMP Map” to construct the table of contents and reference the associated content. In this case, you can create a DMP image directly from the map. A DMP Map supports all of the configuration options available in the standalone DMP configuration.

Following is an example of a DMP Map:

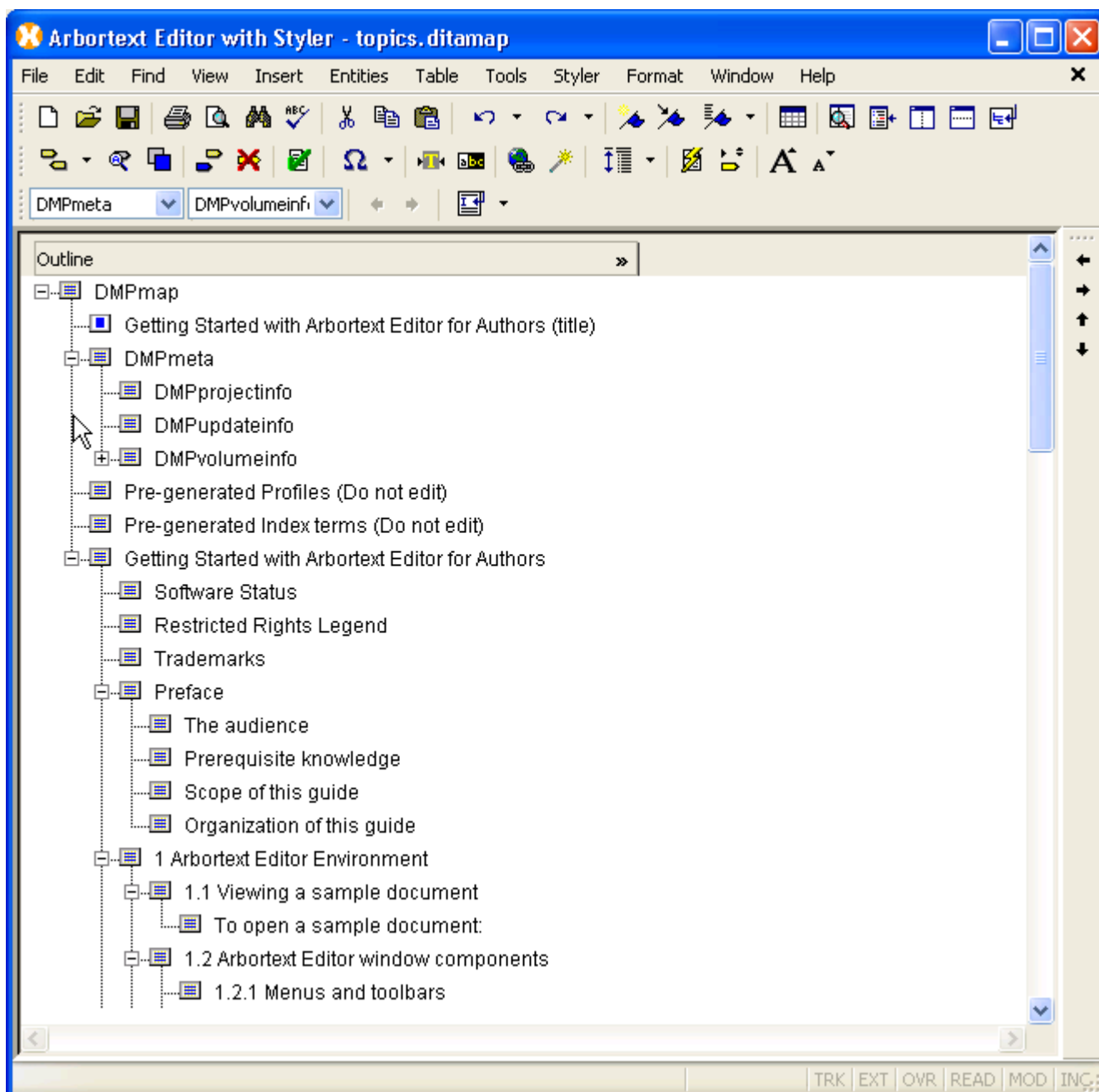


Figure 27: Sample DMP Map

Authoring

Authors will need to learn about the specialized elements and attributes in a DMP Map to use the map. However, for authoring regular DITA topics and maps, nothing special is required.

Integration

DMP provides both a Java API and an XML RPC API. The same features are available in both of these APIs. The API provides several capabilities, including displaying an individual topic based on an ID and searching through HTML metadata for strings or IDs to find the appropriate topic to display. Application developers can use the DMP API to integrate a DMP image with their software for use as online Help.

All PTC documentation groups will be using DMP images for their products' online Help.

Output

A DMP image is highly customizable. All of the images/icons, colors, text, and features of a DMP image can be modified or removed during configuration. The default image contains the following features:

- Table of Contents tab — Provides a table of contents for the content in the image.
- Search tab — Contains search results, arranged either by relevance or position in the table of contents.
- Index Terms tab — Contains a universal index for the content in the image.
- Bookmark tab — Contains personal bookmarks and comments for individual topics.
- Configuration tab — Enables the Help Center image to be customized in various ways (for example, switching to a different language or filtering the content based on various metadata criteria).
- Search field — Enables you to enter search terms with full Boolean, wildcard, and other standard search support.
- Advanced search — Enables you to filter search results based on various criteria.
- Toolbar — Provides access to various DMP features.
- View frame — Displays the current topic.

Following is a DMP image delivered in the embedded browser for use as online Help or a standalone knowledge base:

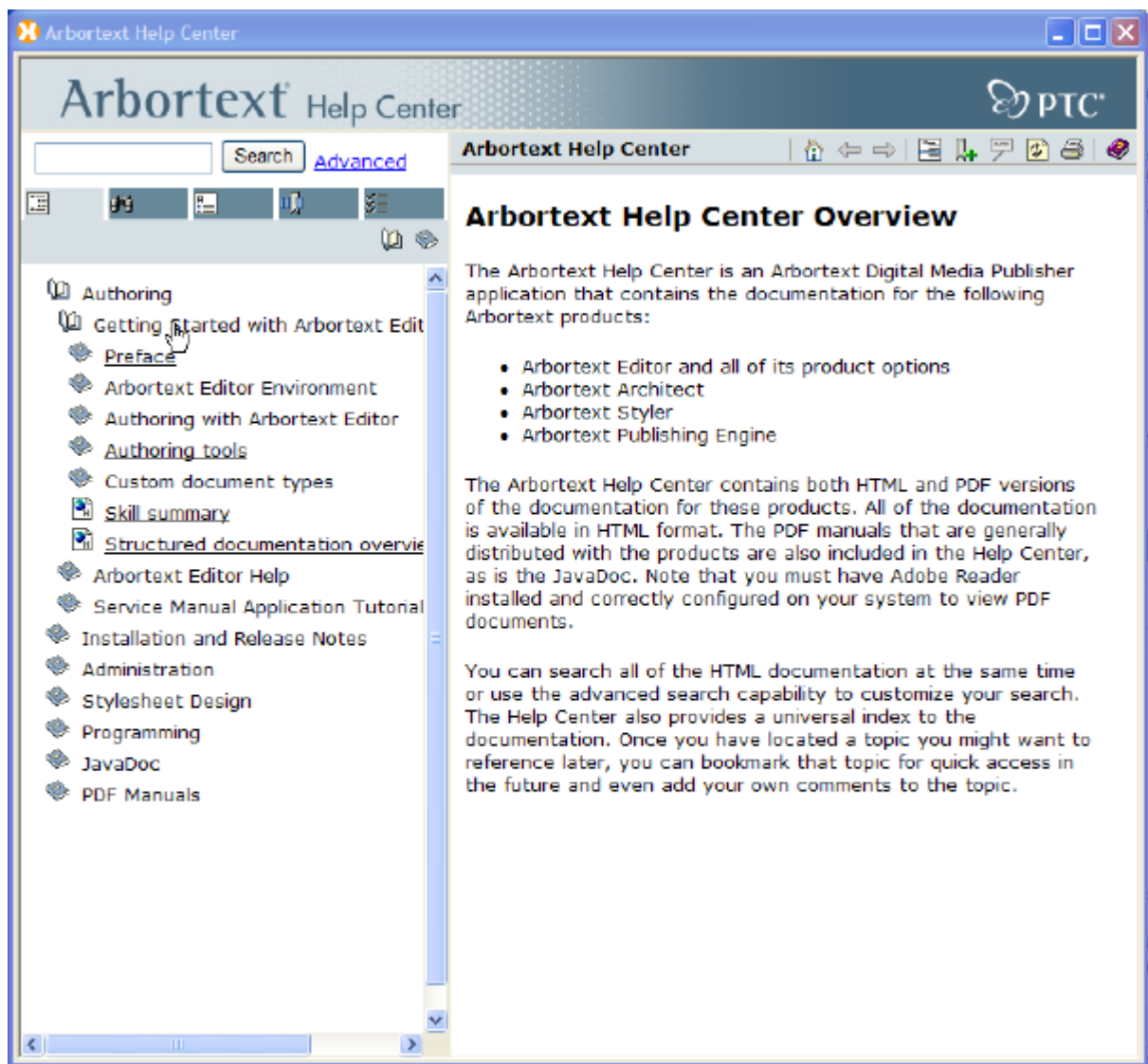


Figure 28: DMP Browser-based Help

Following is the same image delivered as a web application:

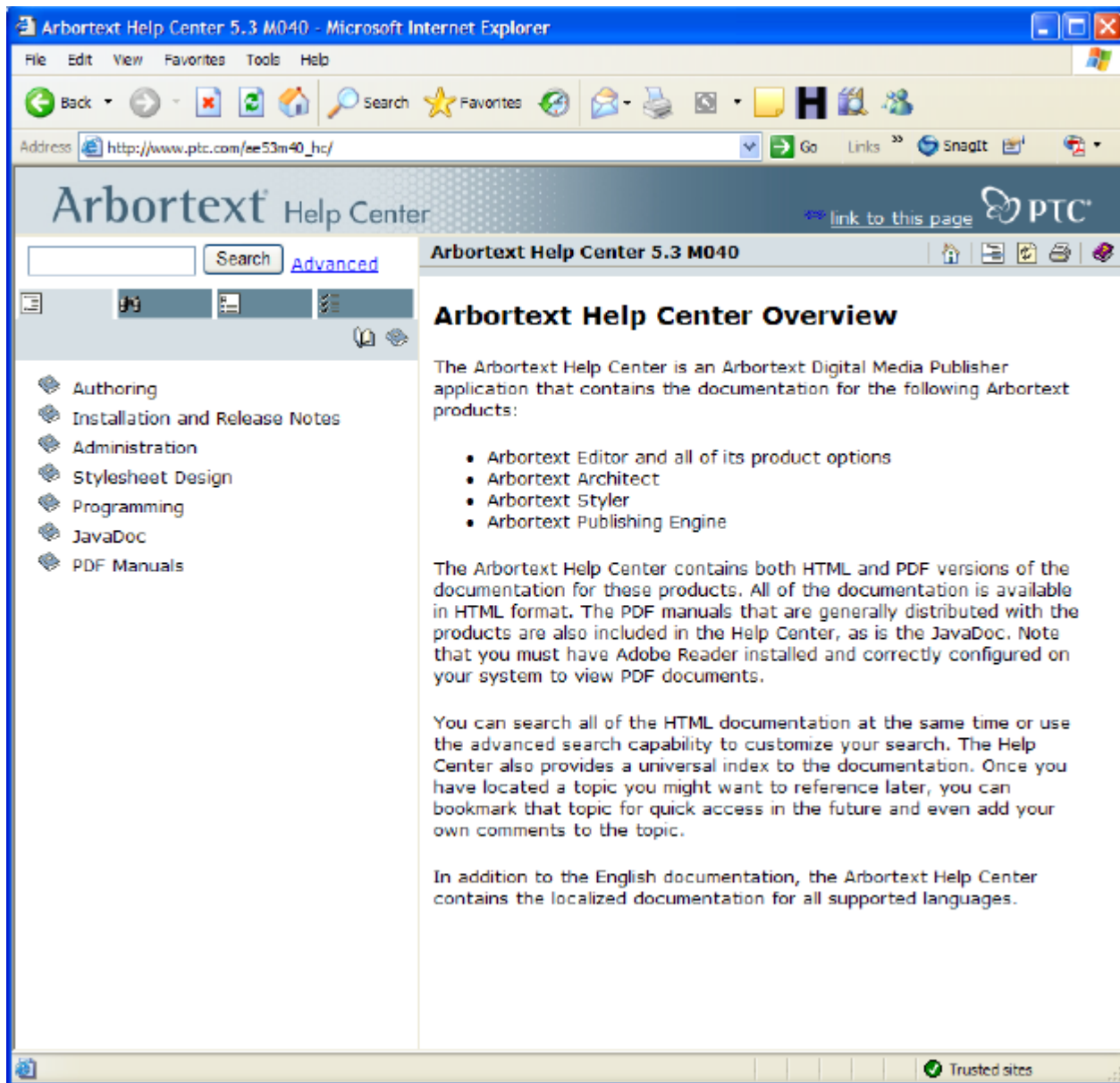


Figure 29: DMP Help as a Web Application

The main difference between the two images is that the bookmark and commenting feature in the standalone image is replaced by the web browser's favorites or bookmarks capability. For either case, displaying documents other than HTML leverages browser plug-ins. For example, PDF documents are displayed in the View frame in the Adobe Reader plug-in.

Summary

Arbortext Digital Media Publisher enables you to go directly from a DITA map to a composed image. It is a highly flexible product that can be customized as needed. The DMP API enables integration with other products for use as online Help or a knowledge base.

Chris Goolsby

PTC Corporation / Arbortext

OASIS DITA Help Subcommittee

Quadralay ePublisher

<<info here>>

<<from Ben Alums>>

Help formats:

- Eclipse Help
- Microsoft HTML Help 1.x
- Oracle Help
- Sun JavaHelp 1.1.3
- Sun JavaHelp 2.0
- WebWorks Help 5.0
- WinHelp

print:

- PDF
- PDF - XSL-FO

eBook formats:

- eBook - ePUB 2.0
- Microsoft Reader
- Palm Reader

Wiki formats:

- Wiki - Confluence
- Wiki - MediaWiki
- Wiki - MoinMoin

web formats:

- Dynamic HTML
- Simple HTML
- XML+XSL

SDL XPP

<<info here>>

WinANT Echidna

When you have DITA documents you want to publish, you can use WinANT Echidna to automate the processing of the output.

WinANT Echidna is a Windows application used in conjunction with ANT and the DITA Open Toolkit to process DITA output.

WinANT Echidna is an open-source application you can download from the Internet. It is used for the sole purpose of processing DITA documents. WinANT Echidna simplifies the publishing process by acting as a Windows interface to ANT.

WinANT Echidna is an open-source application which is freely available to all internet users.

1. In your web browser, navigate to <http://sourceforge.net/projects/winant-echidna>.

2. Click Download Now.

A file download dialog box opens.

3. Click Save.

The

Save As

dialog box opens.

4. Select the directory on your computer where you want to save the file, and click Save.

The location where you save the installation program can be different to where you want to install the WinANT Echidna software. You can specify where to install the software when the installation program runs.

WinANT Options Supporting HTML-Based Output

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 of 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 2008 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 drop-down 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.

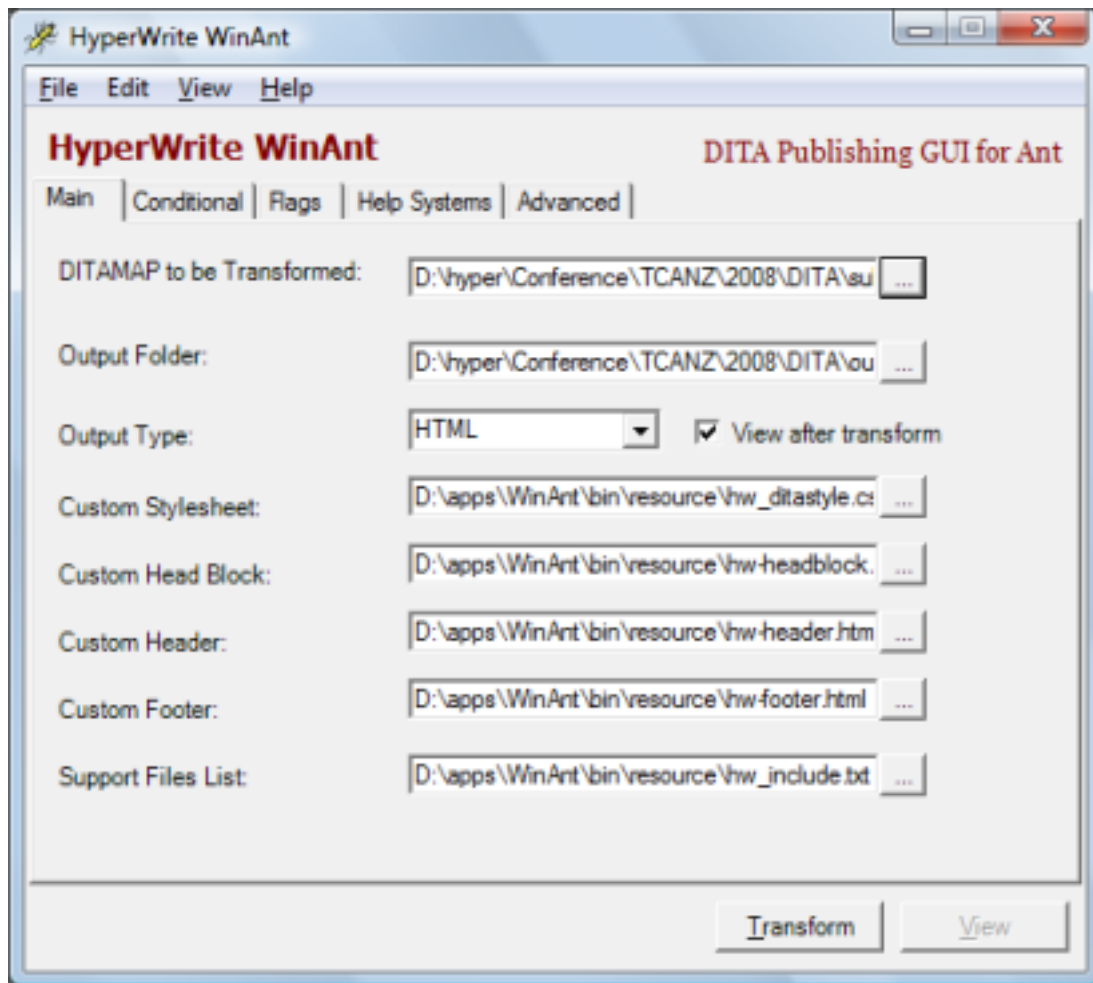


Figure 30: WinANT 1.6 - Main tab

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, and requires *.Net Framework 2.0*.

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).
- Some limited 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.
- 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 DITA map file through WinANT.

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

- HTML
- (Microsoft) HTML Help
- HTML Help 2 (plug-in)
- PDF
- PDF2
- Eclipse Help
- DocBook
- Word
- HTML - tocjs (plug-in)
- XHTML with Search (plug-in)
- HTML - tocjsbis (plug-in)
- 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 you 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* or *HTML Help 2*.

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.

Tony Self

HyperWrite Pty. Ltd.

Chairperson, OASIS DITA Help Subcommittee

WinANT Options Supporting Microsoft HTML Help

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 2008 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 drop-down 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.

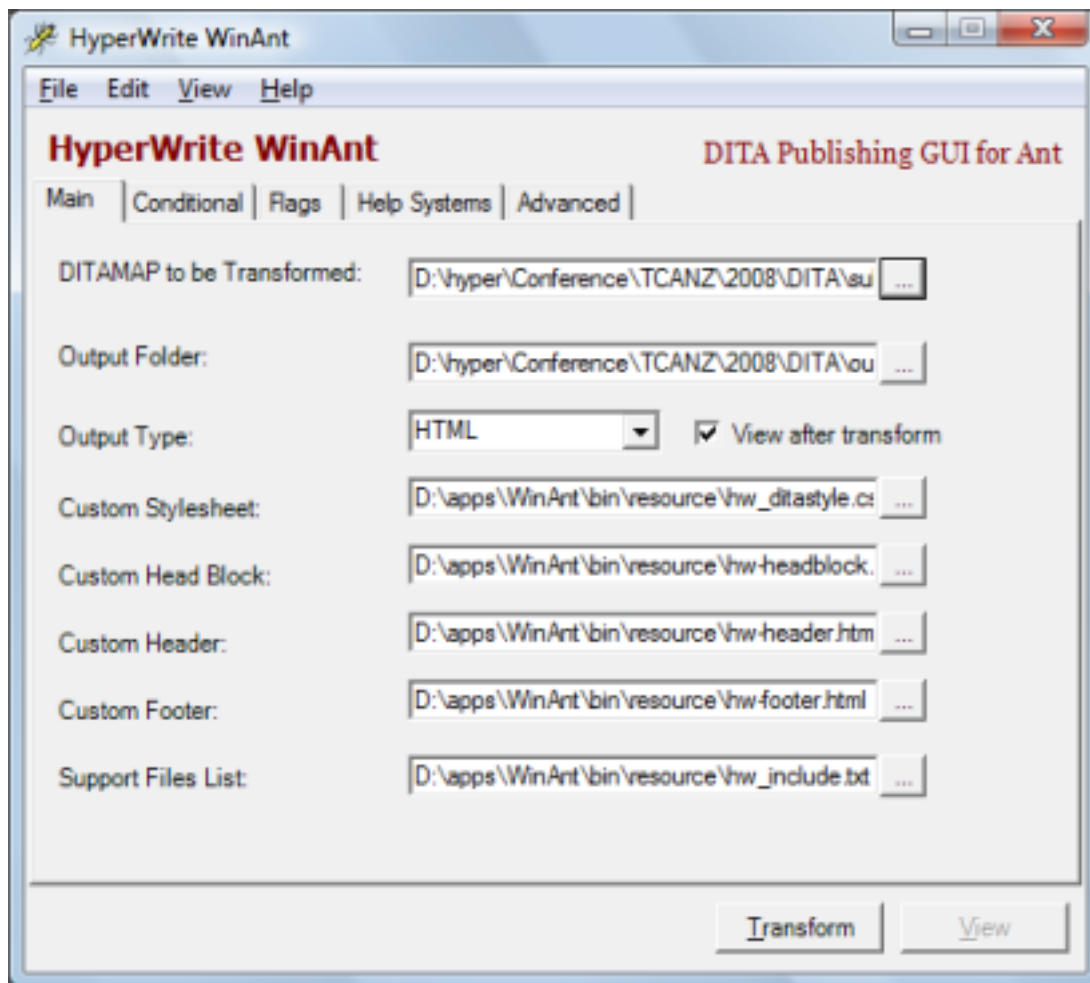


Figure 31: WinANT 1.6 - Main tab

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, and requires *.Net Framework 2.0*.

To enable the incorporation of map and alias files in HTML Help output, you have to manually edit the standard build_dita2htmlhelp.xml DITA OT, by inserting after the line:

```
<param name="OUTTEXT" expression=">${out.ext}" if="out.ext" />
```

the two lines:

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

Authoring

Nominating the map and alias files in the HTML Help project file effectively *externalizes* 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 **Help Systems** tab. These fields are only active if the **Output Type** field on the **Main** tab is set to *HTML Help* or *HTML Help 2*.

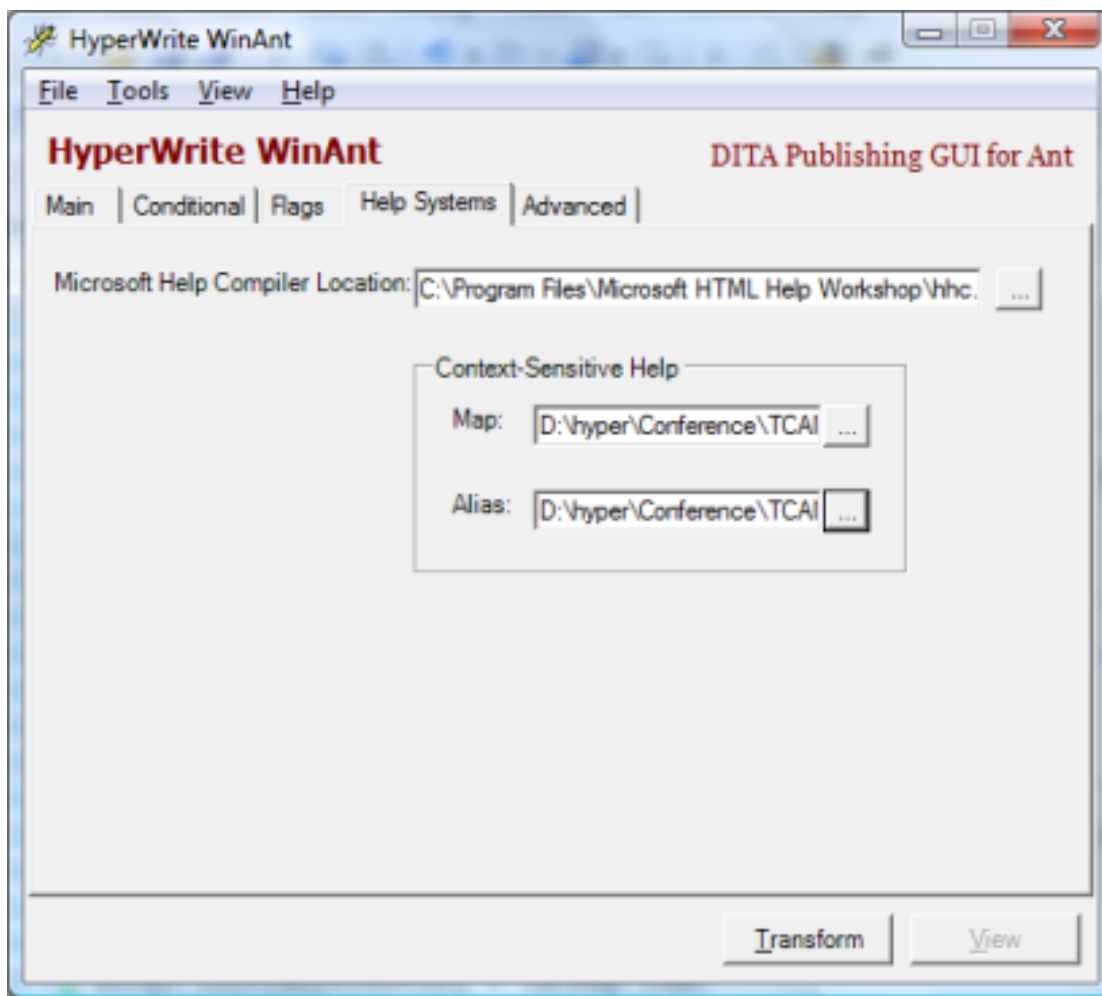


Figure 32: WinANT 1.6 - Help Systems tab

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.

Tony Self

HyperWrite Pty. Ltd.

Chairperson, OASIS DITA Help Subcommittee

XMLMind DITA Converter

<<info here>>

Help Development Techniques

Developing Custom DITA-based Help Systems

If your company has developed a custom Help viewer or custom run-time environment, it is unlikely that the default output from an existing DITA plug-in will be what you need. This section of the *DITA Help Technologies Guide* explains how to use DITA plug-ins and other tools 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.

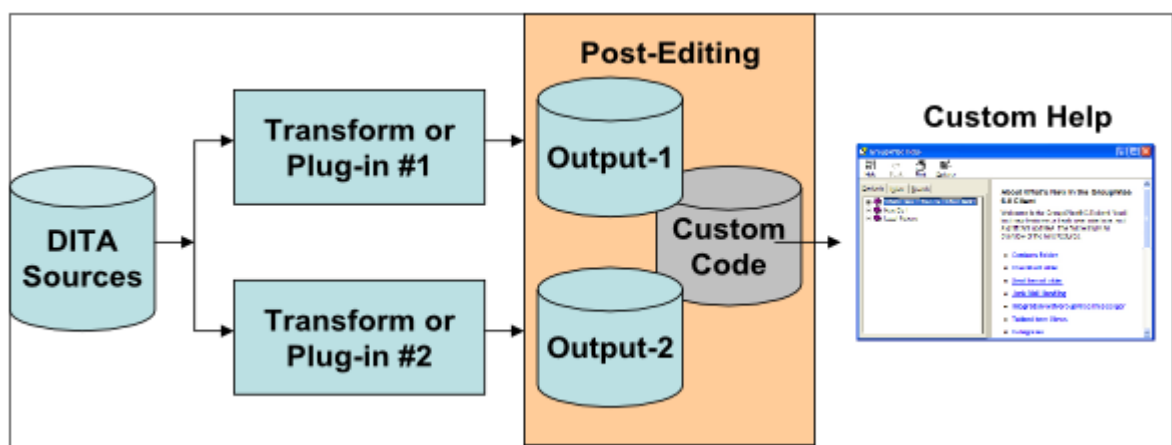


Figure 33: DITA Source to Customized Help Environments

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.

- [#unique_63](#)
- [#unique_64](#)
- [#unique_65](#)
- [#unique_66](#)
- [#unique_67](#)
- [#unique_68](#)
- [#unique_53](#)

Developing DITA-based Help for Existing Help Environments

If your company delivers 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.

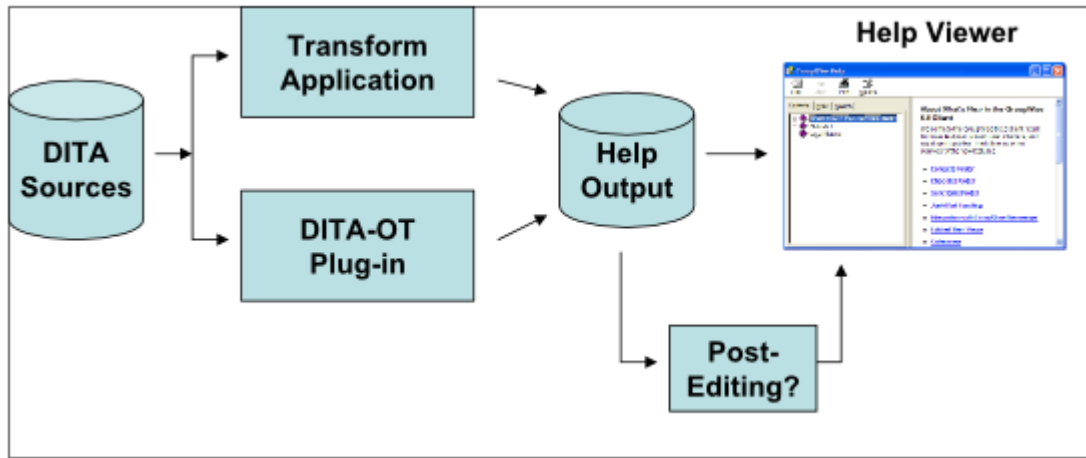


Figure 34: DITA Source to Existing Help Environments

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

- [#unique_69](#)
- [#unique_70](#)
- [#unique_71](#)
- [#unique_72](#)
- [#unique_73](#)
- [#unique_74](#)

DHTML Effects in HTML Generated from DITA

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, drop-down text and popup text, are implemented using Dynamic HTML.

Overview

The DITA Open Toolkit HTML transforms do not provide for layering effects. However, some changes to the XSL-T files, and the use of outputclass metadata 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 this technique, a note element is encoded to output as drop-down text, where the note label is used to toggle the display of the note text. The note element is simply marked up with a distinct outputclass 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 this technique, note how the following lines change the way that the XSL-T file processes the note element:

```
<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){
      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. See <http://www.hyperwrite.com/Articles/showarticle.aspx?id=71>.

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.

Tony Self

HyperWrite Pty. Ltd.

Chairperson, OASIS DITA Help Subcommittee

Dynamic Rendering of DITA into XHTML

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 regenerated, 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 (<http://www.hyperwrite.com/Articles/default.aspx>) 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 minimize 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, provide greater support for complex content structures.

Tony Self


HyperWrite Pty. Ltd.

Chairperson, OASIS DITA Help Subcommittee

JavaScript-Based Context Sensitive Help

Overview

If you are deploying a browser-based Help system in conjunction with a software application, you can configure your main HTML Help file to display a specific target Help file when so requested by a calling software application.

 **Note:** The following JavaScript functions do not depend technically on anything inside your DITA source files or anything generated by the DITA Open Toolkit. They are generic functions that support generic HTML.

Single-pane Browser Help Systems

In some situations, you may not need to deploy a fancy, multiframe HTML Help solution to support a software application. In these situations, you can add a few lines of JavaScript to the beginning of your main Help file to process incoming hyperlink calls that target a particular HTML Help file.

For example, if my software application needed to display a particular Help screen (`targetHelpTopic.html`), I could build a hyperlink that would call the main Help file (`1file.html`) and ask it to display my target Help topic.

Here's the HTML hyperlink call.

```
<a href="1file.html?targetHelpTopic.html">link</a>
```

Here's the JavaScript to put at in your main Help file.

```
<HTML>
<!-- One-pane Browser Help System (1file.html) -->
<body>
<script language="javascript">
targetTopic = window.location.search.substring(1);
window.location.href=targetTopic
</script>
</body>
</HTML>
```

When the main Help file (1file.html) receives the HTML hyperlink call `link`, it identifies any text that follows the question mark (?) as one or more arguments being passed to it. In this example, the script identifies `targetHelpTopic.html` as that argument and makes that HTML file name the current topic to be displayed in the browser.

Note: This setup works well for captive or embedded Help systems with restrictive or highly predictable calling procedures. If a user or an application calls the main Help file without specifying a target topic, the main Help file will generate an Error 404 "File not found".

Multiframe Browser Help Systems

Many browser-based Help systems use multiple frames to provide the customer with topic navigation, search, indexing, and other basic amenities. To get a multiframe Help system to respond to the same context-sensitive hyperlink call is a bit more complicated because of some restrictions in how HTML frames and JavaScript interact.

Here are the elements of our little example.

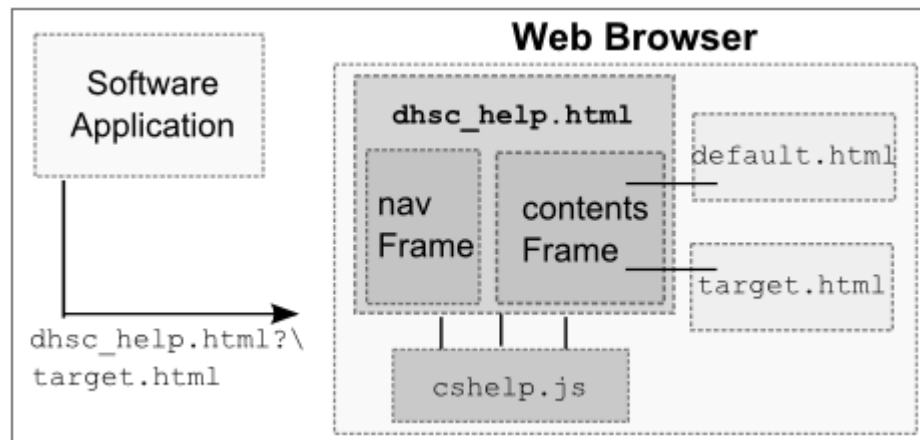


Figure 35: JavaScript Elements

Elements	Description
Software application	Any software application that can generate HTTP calls to a browser will work. For the sake of simplicity, I use an HTML document that contains one hypertext link.
	<pre><html> </pre> <p>Call the help system file and pass it an argument</p>

Elements	Description
	<pre data-bbox="662 205 1468 268"><(the name of the target help topic). </html></pre>
Hyperlink	<p data-bbox="662 310 1468 342">The software application passes two arguments to the Help browser:</p> <ul data-bbox="662 363 1468 457" style="list-style-type: none"> <li data-bbox="662 363 1468 394">• Main HTML Help file name (<code>dhsc_help.html</code>) <li data-bbox="662 405 1468 457">• Target Help file name to be displayed (<code>target.html</code>) in the main Help window. <p data-bbox="662 478 1468 604">Preserving the question mark (?) between the name of the main Help file name and the target Help topic file name is critical. Your browser uses this question mark to delimit the name of the target Help topic (<code>target.html</code>) from the name of the main Help file (<code>dhsc_help.html</code>).</p>
Main Help file	<p data-bbox="662 646 1468 709">The main Help file here contains little more than a call to a JavaScript library file that performs all the real work.</p>
	<pre data-bbox="662 741 1468 1161"><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd"> <html xmlns="http://www.w3.org/1999/xhtml"> <head> <!-- Sample: dhsc_help.html --> <title>DHSC CS Help Demo</title> <!--Insert cshelp.js library --> <script language="Javascript" src="cshelp.js">/script> <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"> </head><script language="javascript"> insertFrameset() </script></html></pre>
cshelp.js	<p data-bbox="662 1213 1468 1276">The <code>cshelp.js</code> library contains only one JavaScript function named <code>insertFrameset</code>.</p>
	<pre data-bbox="662 1308 1468 1843">function insertFrameset () {defaultTopic = "default.html";targetTopic = window.location.search.substring(1); if (targetTopic==" ") { target_contentsframe=defaultTopic } else { target_contentsframe=targetTopic } document.write('<FRAMESET cols="20%, 80%">'); document.write('<FRAME src="nav.html" name="navframe">'); document.write('<FRAME src="'+target_contentsframe+' " name="contentframe">'); document.write('</FRAMESET>'); }</pre>
	<p data-bbox="662 1864 1468 1925">The variable <code>targetTopic</code> points to the name of the target topic (<code>target.html</code>) embedded in the hyperlink call. If the JavaScript function</p>

Elements	Description
	receives no argument from the hyperlink that calls it, it uses the name of the HTML topic declared as <i>defaultTopic</i> . The final lines of the script build the frameset calls that specify frame dimensions and contents (HTML files).

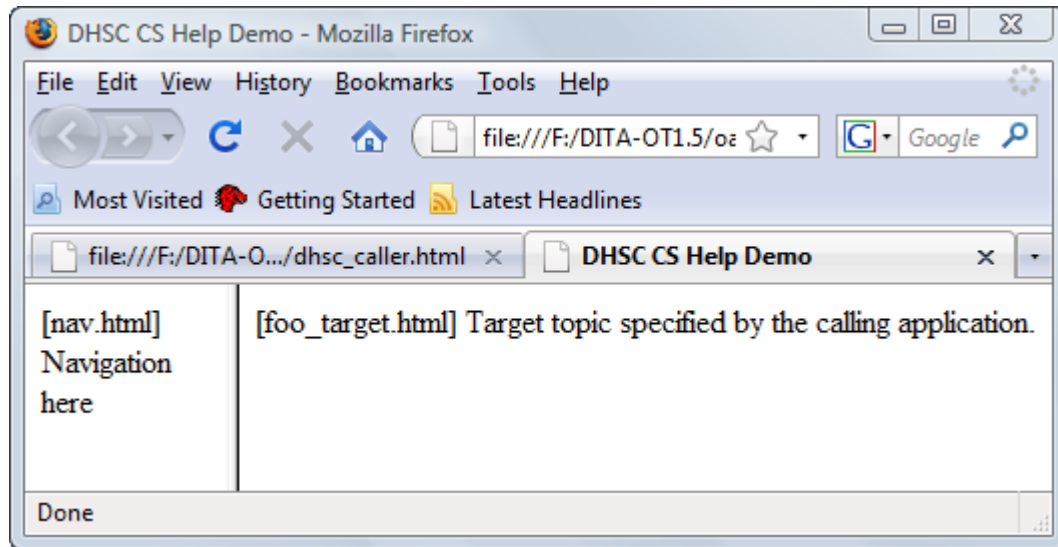


Figure 36: Sample Context Sensitive Help

Summary

These JavaScript functions support small-to-medium Help systems in any language because they place the entire burden of constructing and managing context-sensitive hypertext links on the calling application. With larger Help systems or with distributed Help systems, this overloading becomes risky or impossible.

Resources

OASIS Resources and Sites

Consult the following resources to learn more about the activities of OASIS DITA Technical Committees in general and the OASIS DITA Help Subcommittee in particular.

- OASIS DITA Technical Committee:** This working committee is responsible for the technical development and specification of the OASIS DITA architecture and language.
 TC home page: <http://www.oasis-open.org/apps/org/workgroup/dita/>
 TC Wiki site: <http://wiki.oasis-open.org/dita/FrontPage>
- OASIS DITA Adoption Committee:** The working committee is responsible for promoting the adoption of the OASIS DITA standard by educating the global marketplace on the value of DITA for document creation and management.
 Adoption home page: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita-adoption
- OASIS DITA Help Subcommittee:** This subcommittee of the DITA Technical Committee is responsible for developing a top-level design for authoring of Help systems and user assistance content using DITA and for making recommendations for the integrating of DITA-authored Help systems and software applications using context-sensitivity.
 Help home page: <http://www.oasis-open.org/apps/org/workgroup/dita-help/>
 Roster: <http://www.oasis-open.org/apps/org/workgroup/dita-help/members/roster.php>
 Working documents: <http://www.oasis-open.org/apps/org/workgroup/dita-help/documents.php>

Online Demonstrations

Consult the following resources to learn more about

- Expandable Content in XHTML:** Tony Self has an article on how to customize the output from DITA and the DITA-OT to support a piece of user assistance -- the "Note" tag that provides expandable content when clicked. See the following web page for a working example:
<http://www.hyperwrite.com/Articles/showarticle.aspx?id=71>
- Adobe AIR Help:** Scott Prentice has an online demonstration of "AIR Help," a cross-platform Help format that has been made possible by Adobe's introduction of the AIRTM (Adobe Integrated Runtime) development technology. You should be able to take your "web help" files, and wrap them up in an AIR application that can be installed as a Help system. The same AIR application can be installed on Windows, Mac, and Linux systems.
<http://leximation.com/airhelp/>

Downloads

Consult the following resources to learn more about

- HTMLSearch Plug-in:** This DITA Open Toolkit plug-in is available at the Yahoo DITA site. The plug-in is also integrated with Tony Self's WinANT application.
 Yahoo DITA plug-ins site (you'll need a yahoo account): <http://tech.groups.yahoo.com/group/dita-users/files/Demos/>
- WinANT:** Tony Self has developed a Windows-based application that 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.

http://www.helpml.com/winant_setup.exe

- **FAR HTML:** FAR HTML is a collection of file and HTML Help utilities for authors. If you need to quickly manipulate HTML files, or create and edit HTML Help 1.x or MS Help 2.x projects then FAR will save you lots of time. You can safely use FAR HTML and MS Workshop side by side. Download the full working version of FAR HTML free today. FAR can be used in conjunction with the HTMLHelp2 plug-in to test context-sensitive callbacks once they have been compiled into Microsoft HTML Help.

<http://www.helpware.net/FAR/index.html>

- **Plus HTML Plug-in:** An updated plug-in for Microsoft HTML Help generation from DITA sources is also available at Yahoo. See the `dita-plus.zip` download.

Yahoo DITA plug-ins site (you'll need a yahoo account): <http://tech.groups.yahoo.com/group/dita-users/files/Demos/>

- **Eclipse Help:** Eclipse Help documentation and links to downloads are available at:

Eclipse User Assistance <http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/ua.htm>

Eclipse Help

http://help.eclipse.org/ganymede/topic/org.eclipse.platform.doc.isv/reference/extension-points/org_eclipse_help_toc.html

Further Reading

Consult the following resources to learn more about

- **Dynamic Rendering to HTML from DITA Sources:** Tony Self has an article on how he dynamically renders DITA content on his HyperWrite website. In the "Articles" section, a DITA map is transformed into an HTML web page listing the titles of articles. When a user clicks one of these links, MS ASP.net fetches the DITA source for the selected article and renders it dynamically as HTML. There are no links or conrefs between articles.

Article: <http://www.hyperwrite.com/Articles/showarticle.aspx?id=68>

HyperWrite web site: <http://www.hyperwrite.com>

- **Help Context IDs in Separate Files:** Tony Self has an article on how you can store your Help context IDs in a separate file that can be used by DITA when it produces .chm output for MS Help.

<http://www.hyperwrite.com/Articles/showarticle.aspx?id=71>

OASIS DITA Help Proposals for Further Development

Consult the following resources to learn more about

- **DITA Help Implementation Scenarios:** Tony Self has shared several models for how organizations producing Help from DITA sources can manage their tools workflow.

http://wiki.oasis-open.org/dita/Implementation_Scenarios

- **DITA Help features to consider:** Tony Self has a list that was reviewed favorably by the DITA Help Subcommittee.

<http://www.hyperwrite.com/Articles/showarticle.aspx?id=65>

- **DITA Help APIs:** Tony Self offered several models for context-sensitive Help callback conventions.

http://wiki.oasis-open.org/dita/Help_API

- **Char James-Tanny HAT Matrix:** Char James-Tanny has a feature matrix for Help Authoring Tools (HATs). It is a useful compendium of Help features across Help applications.

http://hat-matrix.com/compare_hats/

- **Help Metadata in DITA Attributes:** Tony Self has proposed the idea that we can embed Help context IDs and other related information in the metadata in topic and topicref attributes.

<http://www.oasis-open.org/apps/org/workgroup/dita-help/download.php/28362/dita-proposalcsh1.html>

- **DITA Target Window Specification:** Tony Self offered a proposal for specifying in DITA a target Help window.

<http://www.oasis-open.org/apps/org/workgroup/dita-help/download.php/28363/dita-proposalcsh-win.html>

- **DITA Help in a DITA Exchange Package (DXP):** Eliot Kimber is working on a standard for compiled DITA called the DITA Exchange Package (DXP). This may be a way to bundle DITA Help sources and/or output conveniently.

<http://ditadxp.sourceforge.net/>

Index

A

AIR Help 50
Arbortext Digital Media Publisher 62

C

CSHelp plug-in 38
Custom DITA-based Help systems 74

D

DHTML effects 75
DITA Help 5
 defined 5
dynamic XHTML rendering 77

E

Eclipse Help 34, 38, 42
 CSHelp plug-in 38
 Eclipse_CSH plug-in 42
Eclipse_CSH plug-in 42
existing Help authoring tools 19
existing Help environments 74

F

further reading 82
 downloads 82
 OASIS resources 82
 online demonstrations 82

H

HTMLSearch plug-in 19

I

introduction 4

J

JavaScript context-sensitive Help 78

L

Leximation AIR Help plug-in 50

M

Microsoft HTML Help DITA-OT transform 48
Microsoft HTML Help htmlhelp2 plug-in 47

R

RoboHelp 54

T

tocjs plug-in 28
tocjsbis plug-in 28

U

user assistance 5

W

WebHelp 54
WinANT 68, 71
 support for HTML output 68
 support for Microsoft HTML Help 71