

Read me: Documentation for Eclipse Help File Conversion Plugin

author: Jaime Wren

last modified: June 9th, 2006

Purpose This program generates a html website out of the files created for the Eclipse Help documentation system. That is, given the XML and HTML source required by the Eclipse help, this task outputs a html website.

Interface The user can start the generation of such a website in a variety of ways:

- 1) Ant script on the command line to launch Eclipse in a headless mode, and create the documents.
- 2) [not implemented] Execute an Ant script from within a launched Eclipse environment.
- 3) [not implemented] Java SWT GUI wizard which walks the user through the steps necessary to generate the files.

1) Ant Script with headless Eclipse

We assume that the workspace has been setup: the workspace directs the ant input task “eclipsetools.generateDocumentation” to the task BuildEclipseDocumentationTask.

eclipsetools.generateDocumentation

<project> and <alternate> can be nested into <eclipsetools.generateDocumentation>. While <alternate> is entirely optional, at least one <project> attribute must be included (containing a set of documentation files).

Assumptions

An Eclipse workspace exists and includes the projects which contain the Eclipse documentation which will be generated into the website.

<i>Attribute</i>	<i>Description</i>	<i>Required</i>
primary	This is the string name of the primary Eclipse help project.	Yes
destinationDirectory	The export directory for the generated documents. This	Yes

<i>Attribute</i>	<i>Description</i>	<i>Required</i>
	directory must be writable.	
templateFile	This is the default template file.	No, a default is provided
titlePrefix	This is a string prefix to appear before every title in the final html documentation.	No, empty string is the default
titlePostfix	This is a string postfix to appear after every title in the final html documentation.	No, empty string is the default

project

Each <project> attribute indicates an Eclipse project containing documentation or table-of-contents to be used in the generation. The named project is required to exist within the Eclipse environment which the ant script is being launched. If a project has documentation (html documents, images, other documents) they must be specified as a collection <fileset> attributes within the <project>. For information on <fileset>, see <http://ant.apache.org/manual/CoreTypes/fileset.html>. The project is searched for the "plugin.xml" file, which is used to reference the *.xml table-of-contents files.

<i>Attribute</i>	<i>Description</i>	<i>Required</i>
name	This is the string name of the required Eclipse project for generation.	Yes

alternate

Each <alternate> attribute indicates a set of files which will use a template file other than the default (see templateFile attribute above). Files associated with an <alternate> attribute are specified via ant's <fileset> structure, see <http://ant.apache.org/manual/CoreTypes/fileset.html>. Each <alternate> file set is required to be contained within the set of files listed under <project>. Also, each <alternate> file set must not intersect any other <alternate> file set.

<i>Attribute</i>	<i>Description</i>	<i>Required</i>
template	This is the string path to a template file to be used instead of the templateFile above.	Yes

Example

```
<property name="workspace.path" value="C:\java\workspace_codepro_3.0.1" />
< eclipsetools.generateDocumentation
    primary = "WSAssistSource"
    destinationDirectory = "C:\tempCodePro"
    templateFile = ".\template_default.html"
    titlePrefix = "CodePro Documentation : "
    titlePostfix = " - Instantiations Inc." >
    <project name="WSAssistSource">
        <!-- The documentation pages -->
        <fileset dir="${workspace.path}\WSAssistSource\doc">
            <exclude name="**/Thumbs.db"/>
        </fileset>
        <fileset file="${workspace.path}\WSAssistSource\CPS-EvalGuide.pdf" />
    </project>
    <project name="WSAssistSourceAnalysisDependencyUI"/>
    <project name="WSAssistSourceAnalysisUI"/>
    ...
    <alternate template = ".\template_empty.html">
        <fileset
file="${workspace.path}\WSAssistSource\doc\features\audit\audit_report.html"/>
        <fileset
file="${workspace.path}\WSAssistSource\doc\features\audit\audit_series_report.html"/>
        ...
    </alternate>
</eclipsetools.generateDocumentation>
```

2) Ant Script within Eclipse

Same as **1)** above.

3) Java SWT GUI wizard

Not implemented yet.

Template File

The aim of the template file is to allow the user maximum freedom in designing the layout of the generated documents. The template files are simply html documents with marks indicating where the navigation bar, bread crumbs, and content should be inserted. When the documentation is being generated: the content of the html documentation is removed; the content of the template is copied in; and then the title, the bread crumb text, the navigation bar text, and the content for the documentation is inserted. If the tags do not exist for one of these insert-able elements, then nothing is inserted.

Examples:

A header-footer pattern template

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<!-- EmbedFromOriginal area="title" --><!-- EndEmbed -->
<!-- import a CSS file -->
<link href="http://www.SomeCompany.com/styles/style.css" rel="stylesheet" type="text/css"
media="screen" />
</head>

<!-- HEADER -->


<!-- BREAD CRUMBS -->
<!-- EmbedFromOriginal area="breadcrumbs" -->
<!-- EndEmbed -->

<!-- NAVIGATION TREE -->
<div class="nav-tree-section">
<!-- EmbedFromOriginal area="navTree" --><!-- EndEmbed -->
</div>
```

```
<!-- CONTENT -->
<div class="content-section">
<!-- EmbedFromOriginal area="content" --><!-- EndEmbed -->
</div>
```

```
<!-- FOOTER -->
<img src= "www.SomeCompany.com/footer.jpg">
</body>
</html>
```

A template for a generated document

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
<body>
```

```
<!-- BREAD CRUMBS -->
<!-- EmbedFromOriginal area="breadcrumbs" -->
<!-- EndEmbed -->
```

```
<!-- CONTENT -->
<!-- EmbedFromOriginal area="content" -->
<!-- EndEmbed -->
```

```
</body>
</html>
```

Technical Notes This section aims to bring future programmers up to speed on the source behind the product.

The source is broken (and conceptually broken) into the following sections:

- The main class, BuildEclipseHelpDocsMain, which takes an abstract ITOCManager, ProjectAttribute[], AlternateAttribute[], destination File (java.io), default template File (java.io), a title prefix and finally a title postfix. This class contains the method “run” which is the single thread which performs the operation. In order, the following summarizes the steps taken by the this method (run). For more information on used classes /resources, see below.
 - 1) Reads input and throws exceptions if possible. For example, a inputting a template file which does not exist or is not readable will cause an IllegalArgumentException to be thrown.
 - 2) Reads the contents of the template file (with FileUtilities class) and stores the file contents locally.
 - 3) Uses the given ITOCManager and the EclipseHelpTreeGenerator class to generate an EclipseHelpTree. (This tree is in the same structure as the table-of-contents tree structure.)
 - 4) Verifies that all the referenced files in the generated tree are included in the documentation files (files in ProjectAttribute[]).
 - 5) Copies all documentation files into the destination directory.
 - 6) Update the nodes in the tree to include the location of the copied file (step 5).
 - 7) For each node in the tree that doesn't have a page to reference, a page is created with placeholder information (a subset of the navigation tree). We call these pages **blank** pages.
 - 8) An **index** (.html) file is created and placed in the destination directory which redirects the browser to the first non-blank page.
 - 9) All of the files that the tree points to are converted to their new contents using the default template (unless there is an alternate defined for the documentation file.)
 - 10) All of the documentation files that were given are searched through and we verify that they were are converted (in step 9). If they were not, then they are **orphan** files (files in the documentation but not referenced by the [table-of-contents] tree). In this case, we find a file in the same directory as the orphan and use the same navigation menu and bread crumbs text. (Actually it is a small variant of the text from the adopting documentation file, see BreadCrumbGenerator and NavigationTreeGenerator below for more.)
-
- Source to interface with the file system to retrieve the required table-of-contents *.xml files. Classes which implement ITOCManager are such files, see the package com.instantiations.eclipse.doc.tocmanagers.*. (Note: only “hard-coded” examples exists, that is, ITOCManager is implemented in a fashion for testing (2) and (3) below. The usable classes

to implement ITOCManager haven't been written yet.)

- Source to interact with an ITOCManager to generate an EclipseHelpTree (next step). See EclipseHelpTreeGenerator.
- The data structures to hold an Eclipse help tree (i.e. the data contained in the *.xml files). See EclipseHelpTree and EclipseHelpNode.
- [doesn't exist yet] Source to interface with the file system to retrieve required documentation, *.html, files. Classes which implement IDOCManager are such files, see the package com.instantiations.eclipse.doc.html.managers*.
- [doesn't exist yet] Source which takes an EclipseHelpTree and an IDOCManager and verifies/reports: that all files referenced in the EclipseHelpTree exist, can be read, and are not empty files (the *.html files).
- [doesn't exist yet] Source which copies the documentation directory into some specified location.
- [most source doesn't exist yet] Source which parses through all the copied over files and places a header, footer, bread crumb (BreadCrumbGenerator practically done!), navigation tree (NavigationTreeGenerator practically done!), and parse content in the file.
- [doesn't exist yet] Source which generates an “index.html” in the root of the generated documentation file to point to the top file – preference to change this action.