Oracle® Fusion Middleware

Developing Help Systems with Oracle Help

12c (12.2.1.3.0)

E80019-01

August 2017

Documentation for Oracle Help developers that describes how to develop and display HTML-based help systems for Java applications and web applications.

ORACLE

Oracle Fusion Middleware Developing Help Systems with Oracle Help 12c (12.2.1.3.0)

E80019-01

Primary Author: Ralph Gordon

Contents

Preface

Oracle Fusion Middleware Developer's Guide for Oracle Help explains how to use Oracle Help to develop and display HTML-based help systems for Java applications and for web applications.

Audience

This document is intended for authors who wish to create a single help system that can be displayed both in a Java environment and in a web environment.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, refer to Oracle Help Technologies at http://www.oracle.com/technetwork/topics/index-083946.html.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Oracle Help Overview

This part describes an overview of the Oracle Help. It contains the following chapters:

- Chapter 1, "Introduction to Oracle Help"

  This chapter provides an introduction to Oracle Help.
- Chapter 2, "Oracle Help for Java User Interface"

  This chapter describes the user interface components of Oracle Help for Java.
- Chapter 3, "Oracle Help for the Web User Interface"

  This chapter describes the user interface components of Oracle Help for Web.

1 Introduction to Oracle Help

This chapter introduces Oracle Help technologies, Oracle Help for Java and Oracle Help for the Web. It also provides an overview of developing and displaying HTML-based help systems for Java-based environment applications and web applications.

This chapter includes the following sections:

1.1 About Oracle Help

Oracle Help technologies can be categorized into two: Oracle Help for Java and Oracle Help for the Web. Authors can create a single help system that can be displayed—without modification—both in a Java environment, using Oracle Help for Java (OHJ); and in a web environment, using Oracle Help for the Web. Alternatively, authors can just use Oracle Help for Java to create a help system for a Java environment application, or use Oracle Help for the Web to create a help system for a web environment application. Oracle Help for the Web is available in two versions: Oracle Help for the Web and Oracle Help for the Web – UIX.

Throughout this guide, *Oracle Help* is used when the comments apply to both Oracle Help for Java and Oracle Help for the Web. *OHJ* is used when the comments apply only to Oracle Help for Java. *OHW* is used when the comments apply only to Oracle Help for the Web.

1.2 Oracle Help for Java

Oracle Help for Java is a set of Java components, a Java API, and a file formats specification for developing and displaying HTML-based help content in a Java environment. OHJ is designed primarily for displaying help for Java applications, although it can also be implemented as a standalone document viewer for use in a Java environment.

The Oracle Help for Java Developer's Kit (OHJDK) includes the OHJ technology plus tools and documentation for developing context-sensitive help for Java applets and applications. This includes the following:

- **Java components**: OHJ includes a set of default Java user interface components that comprise a complete help system, with a table of contents, index, search, and topic windows
- **API**: The OHJ API includes features for implementing context-sensitive help, for programmatically controlling how help is displayed (size, position, and so on), and for customizing and extending the help system. For example, you can replace a default component with your own, create custom controls, or embed selected components in an application
- **Documentation**: Documentation includes this developer's guide, plus the API reference (provided as JavaDoc documentation).
- **Helpset Authoring Wizard**: The Helpset Authoring Wizard helps you create Oracle Help control files without using a third-party authoring tool.

For more information about OHJ features, see Chapter 2, "Oracle Help for Java User Interface".

1.3 Oracle Help for the Web

Oracle Help for the Web, also known as Oracle Help for the Web – Rich Client (OHW-RC) because of its rich interface, delivers HTML-based Help content in a Web environment. It uses the Oracle Application Developer Framework (ADF), which is based on the Java Server Faces (JSF) technology, to build a user interface that follows Oracle's Browser Look And Feel Plus (BLAF+) behaviors and guidelines.

The Oracle Help for the Web can be used in many different situations:

- As a help system providing context-sensitive Help to a rich client application in a new browser window
- As a standalone document viewer of Help content on a public website
- When a user performs a search on any popular search engine and the results link to indexed Oracle Help for the Web content
- When an ADF Faces component's runtime implementation requires to retrieve Embedded Help information (Definition Text, Instructions Text, or Full Help) using the HelpProvider interface it defines

Oracle Help for the Web includes the following:

- **The Oracle Help for the Web Front Servlet**: This is installed on a Web server, which enables Oracle Help for the Web to support URL syntax for context sensitive help requests. It also enables an easy configuration to support multiple Oracle Help for the Web helpsets in a single web application.
- **The Oracle Help for the Web Servlet Filter**: This is used to preprocess requests sent to the JSF servlet.
- **Documentation**: Documentation includes this developer's guide.

1.4 Oracle Help in Oracle JDeveloper

Besides being used for the internal JDeveloper help system itself, JDeveloper includes the Oracle Help for Java runtime library, so if you are developing Java applications with JDeveloper, it is easy to include OHJ as the Java help system technology. For more information, see Chapter 12, "Introduction to Oracle Help for Java Developer's Kit".

JDeveloper does not include the Oracle Help for the Web. You can obtain it, and the complete Oracle Help for Java development kit, from the Oracle Technology Network (OTN).

1.5 Oracle Help Licensing and Support

As a service to the customers and the software community, Oracle provides Oracle Help software and support for free. This includes both Oracle Help for Java and Oracle Help for the Web.

Oracle Help is available for free and may be redistributed as the help system for your application. For full information, see the license distributed with the release.

Post your questions on the Oracle Help Technologies Forum on the Oracle Technology Network.

2

Oracle Help for Java User Interface

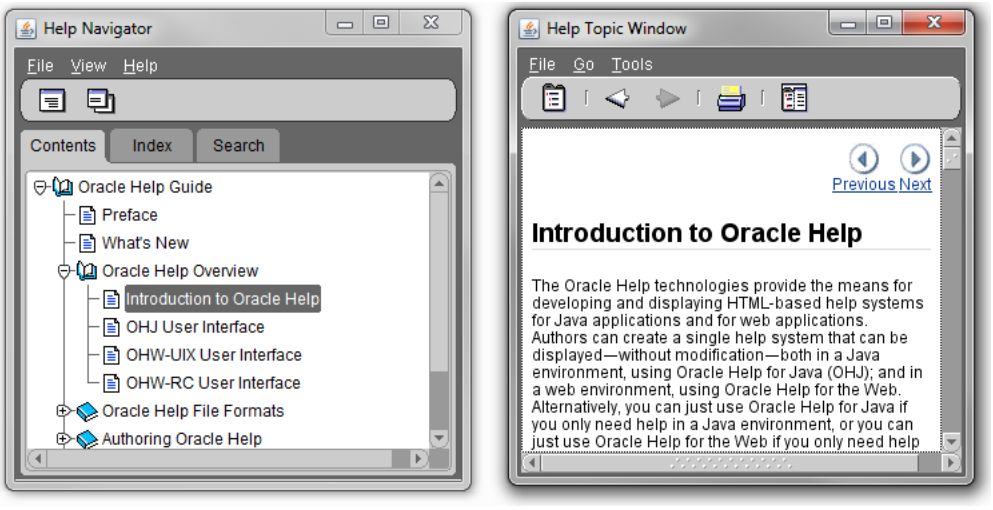
This chapter describes Oracle Help for Java user interface components, such as Navigator window, Topic window, Index tab, Search tab, and Favorites tab.

This chapter contains the following sections:

2.1 About Oracle Help for Java User Interface

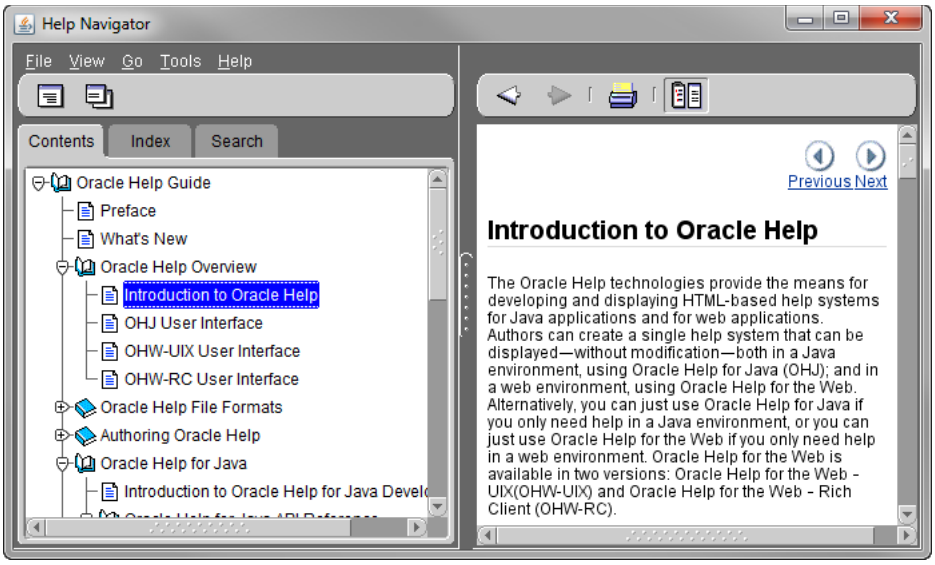
The Oracle Help for Java (OHJ) user interface has two main parts, Help Navigator window and Help Topic window. The Help Navigator window includes controls for finding topics and the Help Topic window displays HTML content.

Figure 2–1 Help Navigator and Help Topic Windows



Users can undock the windows, so they appear as panes in separate windows, as shown in Figure 2–1, or dock them so they appear in as single window as shown in Figure 2–2.

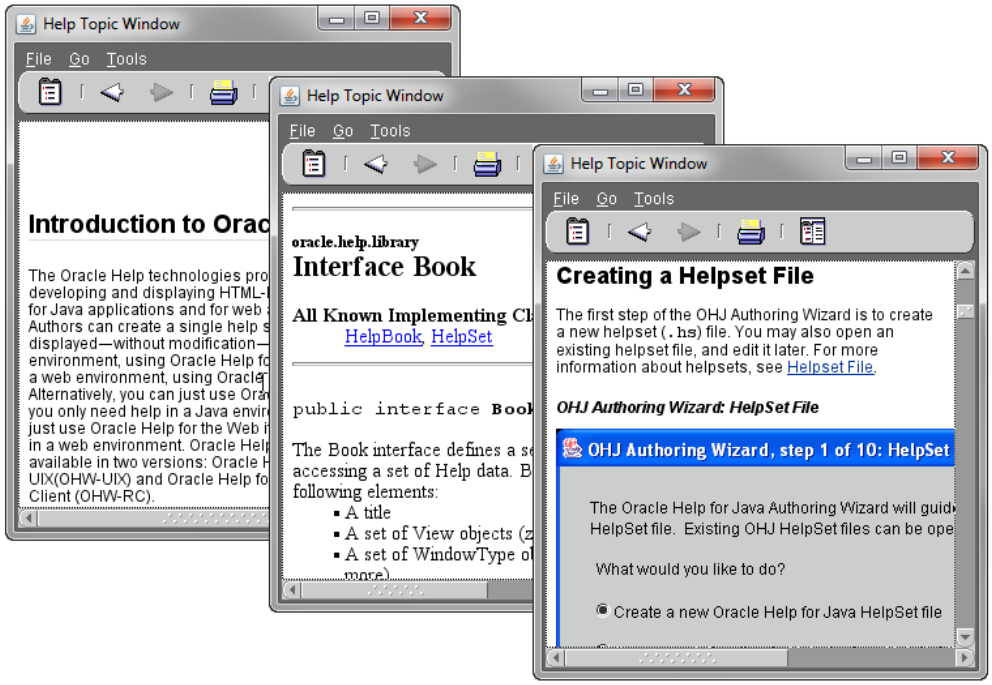
Figure 2–2 Docked Windows



2.2 OHJ Topic Windows

The OHJ Help Topic windows (or topic panes, when docked) display HTML content. Figure 2–3 shows topic windows with different types of HTML content.

Figure 2–3 Topic Windows



The default HTML display component included in the OHJDK is a special implementation of the ICEbrowser from ICEsoft Technologies, Inc. For more information about the browser and its supported technologies, see http://www.icesoft.com. You may use and redistribute this component free of charge

if it is used as part of a help system using OHJ. This HTML display component supports the following:

- HTML 4.0
- Cascading Style Sheets (CSS) 1 and most of CSS 2. CSS 3 is not supported.
- Java applets
- Multimedia, as supported by Java Media Framework 2.0.
- JavaScript
- Support for Screen Reader software
- Single topic and multiple topic printing
- GIF animation
- Popups with HTML support
- Associative links, where a single index word or phrase can be associated with multiple topics. When the user selects one of these links, a list of all topics associated with the link is presented, and the user can choose a topic from the list.
- Author-defined window types, where authors can specify colors such as background color, text color, and link color, window size and position, window title, and toolbar buttons.
- Topic ID linking—hyperlink targets are specified by ID rather than URL
- Synchronization with items in the table of contents

You do not have to use the default HTML display. You can replace it with a different HTML display component. Or, if your application and the help system run as an applet in a Web browser, you can use a browser window as the topic window. Consequently, the display capabilities for your implementation of OHJ rely on the HTML display you chose to embed in the system.

2.3 OHJ Navigator Window

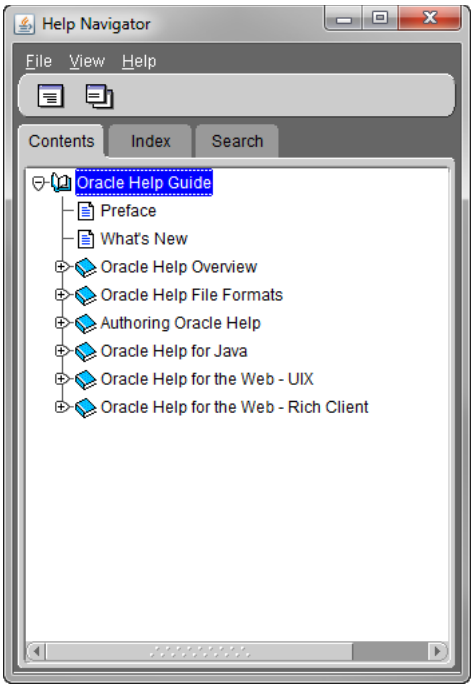
The navigator window is a tabbed control for navigating and finding topics in the help system. By default, the navigator window contains tabs for a **Contents**, **Index**, and **Search**. Authors can control several characteristics of the navigator window simply by setting parameters for the help system. For example, you can change the labels on the tabs and add icons. You could also display multiple tables of contents, for example, one for product help and one for a tutorial. For a more complex system, a Java programmer can create custom tabs, and the author can add them to the navigator window.

This topic contains the following sections:

2.3.1 Contents Tab

The **Contents** tab displays topics in a hierarchical tree. The contents and structure of the tree are specified by the author. Multiple file formats are supported for defining the tree.

Figure 2–4 Contents Tab in the Navigator Window



When a user double-clicks a topic title in the table of contents, that topic is displayed in the topic window. The user may also open a topic in a new, additional, topic window by selecting a button on the toolbar, or by selecting a command from the right-click context menu.

The table of contents view has the following features:

- The item selected (highlighted) in the table of contents is automatically synchronized to the topic shown in the topic window. For example, if you click a hyperlink in a topic and jump to a new topic, the selection in the table of contents switches from the old topic to the new one.
- Unlimited levels of hierarchy are allowed.
- The list of items in the navigator window can be printed.

2.3.2 Index Tab

The **Index** tab displays an alphabetical list of keywords associated with topics. The keywords are defined by the help author, and, like the table of contents, multiple file formats are supported for specifying the list.

Figure 2–5 Index Tab in the Navigator Window

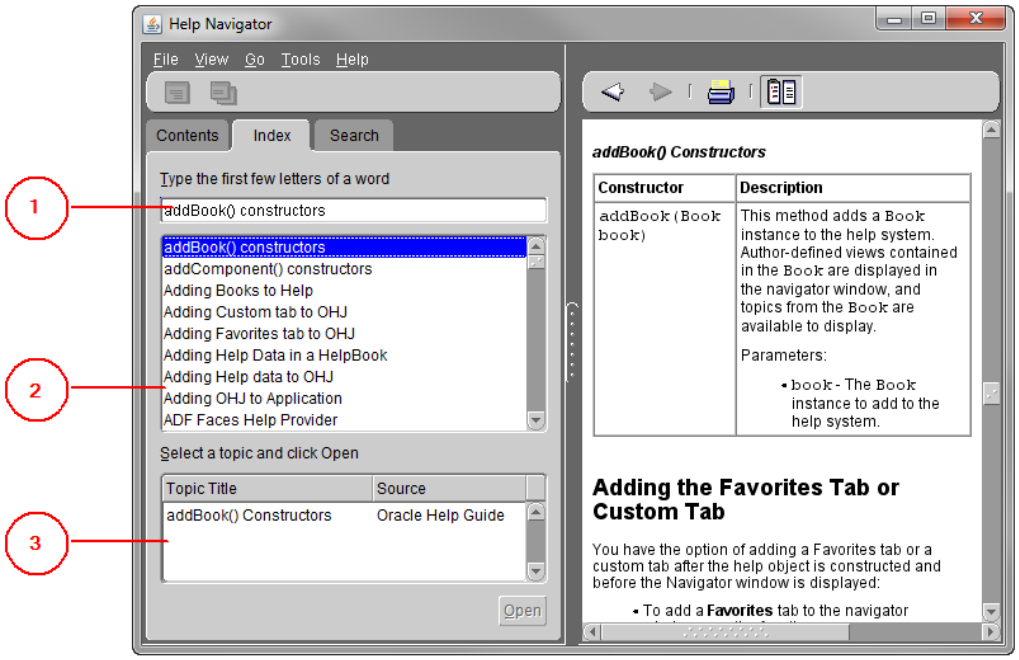


Figure 2–5 numbered callouts identify the following user interface components:

1. **Text entry field**: The user types a word or words in this field.
2. **Keyword list**: As the user types, the first keyword in the list that matches the typed letters is selected. As more letters are typed, a more accurate selection is made. Alternatively, the user can simply select a keyword from this list.
3. **Topic list**: The titles of any topics that are associated with the keyword selected in Keyword List are displayed in this list. When the user double-clicks one of these titles, the topic is displayed in the topic window. The user may also select the topic and click **Open**.

The index provides many useful features:

- One keyword can be associated with many topics, and many keywords can be associated with a single topic.
- Two levels of indenting are supported.
- When merging helpsets, one unified index is created with proper alphabetization for the entire index. In other words, the entries are merged and resorted; they are not concatenated.
- Indexes are properly alphabetized when translated into languages other than that used to create the index.
- Identical entries are collapsed to show only one.

2.3.3 Search Tab

The **Search** tab displays a text field where the user can enter text, then select **Search**. The titles of topics whose content contains that word or phrase are listed in the **Results** list at the bottom of the tab. When the user double-clicks a title, that topic appears in the topic window.

Figure 2–6 Search Tab in the Navigator Window

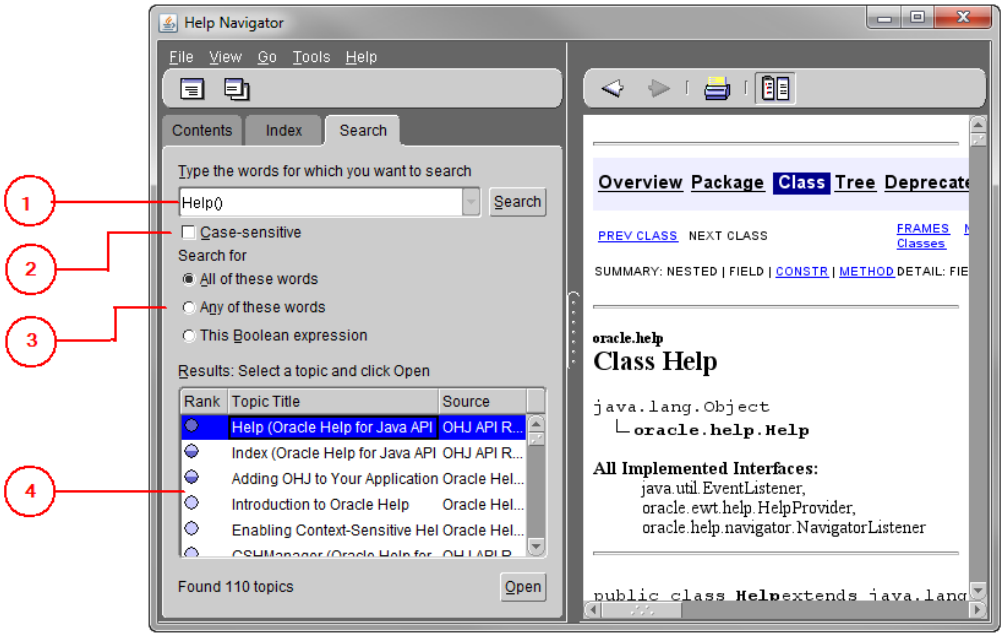


Figure 2–6 numbered callouts identify the following user interface components:

1. **Search text**: The user enters a word or words in this field. To search for an exact phrase, enclose the search phrase in double quotes.
2. **Case Sensitive**: A checkbox, when selected, enables searching for words having the same case as the words entered by the user.
3. **Search for**: A set of option buttons that allows a user to specify whether to list topics that contain any or all of the specified words, or perform a search based on a Boolean expression (AND, OR).
4. **Topic List**: The results area displays the **Rank, Topic Title**, and **Source** of each topic that matches the search criteria. The **Rank** column indicates the ranking of the topics according to how well they match the search criteria. All columns can be sorted in the ascending or descending alphabetical order. By default, all topics are sorted by **Rank**. When the user selects a particular result, the associated topic is displayed in the topic pane.

Users can set the following options when performing a search:

- Enable or disable case-sensitivity of search keywords.
- List topics that contain all search words or at least one search word.
- Search based on a Boolean expression (AND, OR).

Search tab provides many useful features:

- Results are ranked according to how well they match the search criteria.
- Previously entered search criteria are saved and can be displayed in a drop-down list.
- Results list the title of the topic and its source.
- Users can sort results by rank, topic title, or topic source.

The search database is generated when authoring the help system. The OHJDK and as other authoring tools that support OHJ include a utility for generating this database, called the Text Search Indexer. The search database uses an Oracle-defined file format. This search database is always used when you implement it on the client. You can also implement your own search on a server. For example, if you store your topics in an Oracle database, you can use the database's text processing capabilities to perform the search.

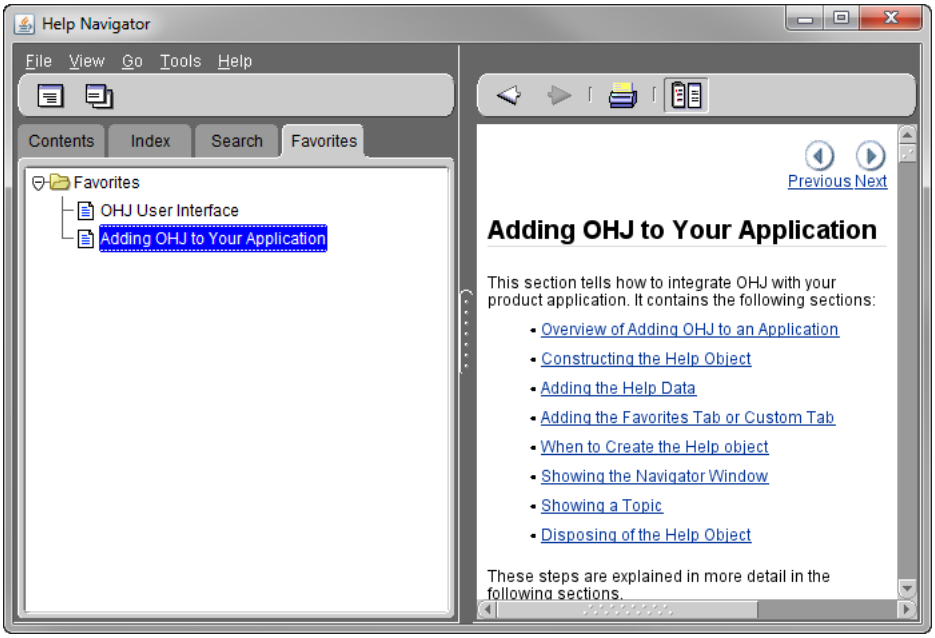
For more information about Text Search Indexer, see Chapter 11, "Using the Text Search Indexer".

Note: An Oracle database is not required to use OHJ.

2.3.4 Favorites Tabs

Users can mark topics in a helpset as favorites using the Favorites Navigator, similar to the Favorites functionality in web browsers.

Figure 2–7 Favorites Tab in the Navigator Window



Users can identify and manage favorite topics from a helpset:

- Add or delete favorite topics
- Create folders and subfolders
- Rename current topics from the default topic title name

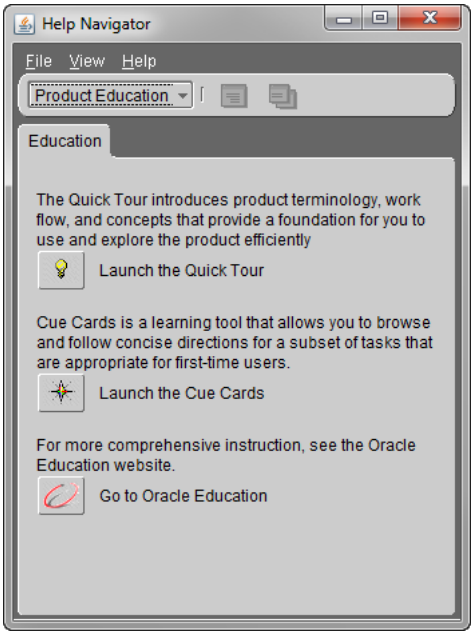
Users can access Favorites navigator functionality from the **Tools** menu of Help Topic window, which displays an Add Favorites dialog, or by right-clicking a favorite in the Favorites navigator.

Note: Unlike **Contents**, **Search**, or **Index**, the **Favorites** navigator is displayed by invoking the method `enableFavoritesNavigator()` URL, which specifies a file, `favorites.xml`, to contain favorites information. For more information, see Section 13.4, "Adding the Favorites Tab or Custom Tab"

2.3.5 Custom Navigator Tabs

Among other features, the OHJ API enables you to customize the default OHJ user interface. For example, you can program custom tabs, also called navigators, in Java and add them to the navigator window. Figure 2–8 shows a custom tab **Product Education**.

Figure 2–8 Custom Navigator Tab in the Navigator Window



2.4 Merged Helpsets

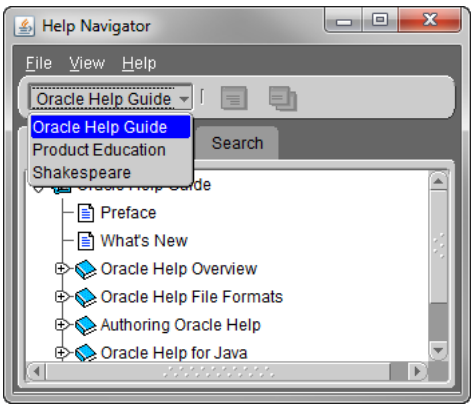
A collection of help topics, with their associated control files, is called a *helpset*. Helpsets can be merged at runtime; multiple authors can create multiple helpsets that are seamlessly merged after authoring completed. Similarly, new components can be added to a user's help system without having to rework the entire system.

Several features support merged helpsets:

- The **Contents** navigator merges helpsets into one tree with the contents from all merged helpsets. As of Oracle Help for Java and Oracle Help for the Web, the Contents navigator merges all tree nodes that have the same labels and do not have conflicting targets.
- The index shows the index entries from all helpsets, properly sorted.
- The text search searches through all merged helpsets.

- Alternatively, authors can specify that selected helpsets are displayed in a drop down list in the toolbar. This can simplify the presentation of a complex help system to the user.

Figure 2–9 Merged Helpsets in a Choice List



When this implementation is used, the **Contents** and the **Index** tabs show only items from the helpset that is selected from the list. The **Search** text searches only for items in the selected helpset.

- When an author provides associative links, the list of topics associated with a link includes items from the merged helpsets.

2.4.1 Merging Table of Contents

Oracle Help for Java and Oracle Help for the Web, both support the merging of TOC files from multiple helpsets to create one tree. The TOC that results from merging multiple tables is essentially the result of laying all trees on top of one another. If multiple TOCs contain identical nodes (nodes that have the same text and target topic Id), then these nodes are combined into one node that has all original nodes' children.

Consider the scenario when you have two helpsets that make use of `toc1.xml` and `toc2.xml`:

toc1.xml

```
<?xml version='1.0'?>
<toc version="1.0">
  <tocitem text="1" target="1_topic" />
  <tocitem text="2" target="2_topic" >
    <tocitem text="2.1" target="2.1_topic" />
    <tocitem text="2.2" target="2.2_topic" />
  </tocitem>
  <tocitem text="3" target="3_topic" />
</toc>
```

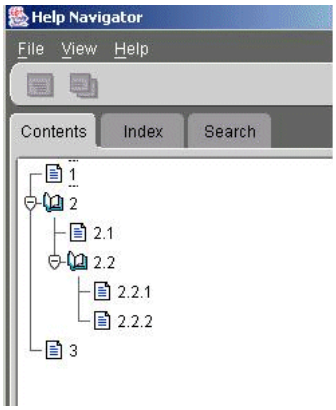
toc2.xml

```
<?xml version='1.0'?>
<toc version="1.0">
  <tocitem text="2" target="2_topic" >
    <tocitem text="2.2" target="2.2_topic">
      <tocitem text="2.2.1" target="2.2.1_topic" />
      <tocitem text="2.2.2" target="2.2.2_topic" />
    </tocitem>
```

```
  </tocitem>
</toc>
```

Figure 2–10 shows the TOC that results from merging `toc1.xml`and `toc2.xml`. You can see that there is only one node for `2` because the target is `2_topic` for both the `<tocitem>` items with `text=2`. The same applies for the node with `text=2.2`.

Figure 2–10 Results of the Merging



This example assumes that the `<view>` elements defining the Contents navigators for the helpsets do not have titles. If a `<view>` for a Contents navigator includes a `<title>`, then this title is used as a parent node to all `<tocitem>` items defined by the `toc.xml` file. For more information, see Chapter 5, "Metadata Files".

2.5 Other OHJ Features

Oracle Help for Java (OHJ) features, not otherwise mentioned in this Oracle Help Guide, include the following features:

- Accessibility Features
- Internationalization Features
- Java Foundation Class (JFC) Swing Components
- Open, Pluggable Architecture

2.5.1 Accessibility Features

OHJ has the following accessibility features:

- All user interface features are keyboard accessible, including the default HTML display component (the ICEbrowser). Oracle has modified the ICEbrowser that ships with OHJ to provide this accessibility support.
- The most recent versions of OHJ and of the ICEbrowser have been successfully tested with the JAWS for Windows Screen Reader from Freedom Scientific. For more information, see `http://www.freedomscientific.com`.

2.5.2 Internationalization Features

OHJ has the following internationalization features:

- All text strings in the user interface are stored in resource files, for easy translation.
- The locale and encoding can be set programmatically.

- The default HTML display component (the ICEbrowser) supports wrapping for non-space separated languages.
- Help authors can set the `charset` of the HTML file, using the IANA character set encoding names (when using the default ICEbrowser HTML display component).
- The OHJ user interface has been translated by Oracle into several languages.

2.5.3 Java Foundation Class (JFC) Swing Components

The OHJ Help system is implemented using Java Foundation Class (JFC) Swing components.

2.5.4 Open, Pluggable Architecture

OHJ has an open, pluggable architecture. That means that you can substitute your own components for default components such as the search facility or the HTML display component. In addition, components such as the Navigator tabs and the HTML display can be embedded into an application's user interface, to provide completely integrated help.

All OHJ application class files, control files, and content files can be encapsulated and compressed into JAR (Java Archive) files. It is not necessary to unJAR these files to run the help system.

3

Oracle Help for the Web User Interface

This chapter describes Oracle Help for the Web user interface components, such as global toolbar, topic navigator, index navigator, and search navigator.

This chapter contains the following sections:

3.1 About Oracle Help for the Web User Interface

The Oracle Help for the Web (OHW) user interface provides the same features as that of OHJ. However, as OHW is a Web application, there are some differences in appearance and behavior.

Figure 3–1 shows OHW in a Web browser.

Figure 3–1 Oracle Help for the Web User Interface

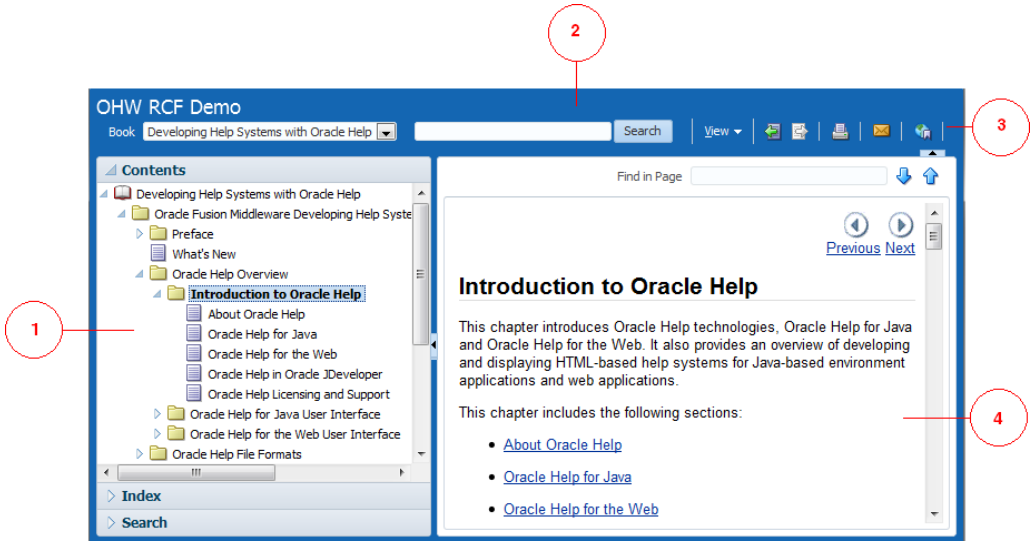


Figure 3–1 numbered callouts identify the following user interface components:

1. **Navigator area**: Located on the left side of the page, this area uses the accordions approach, and each accordion panel represents one navigator. The OHW framework provides three standard navigators: **Contents**, **Index**, and **Search**. The exact set of navigators in an OHW instance is dynamically determined from the helpset definition file (for example, the `.hs` file).
2. **Branding area**: Located at the top of the rich client interface, this area can contain text, an image, or both. Typically, this area identifies the help content or provides company information, such as the name and logo.
3. **Toolbar area**: Located below the branding area, this area provides the main menu items, a helpset switcher (where applicable), and a quick search control.
4. **Topic view area**: When a topic is selected in any tab of the Navigator area, the contents of that topic appear in this area.

With the exception of the branding area, these elements are configured in the helpset file. OHW and OHJ use the exact same file formats, including the helpset file. That means that one can take an existing OHJ help system and deploy it as an OHW system, without changing any of the existing control files.

For more information about deploying OHW system, see Chapter 16, "Deploying OHW Demo File".

3.2 OHW Global Toolbar

The top area of the rich client contains a helpset switcher (where applicable), a quick search control, a **View** menu that enables users to manipulate areas on the page, and a toolbar.

Figure 3–2 OHW Global Toolbar

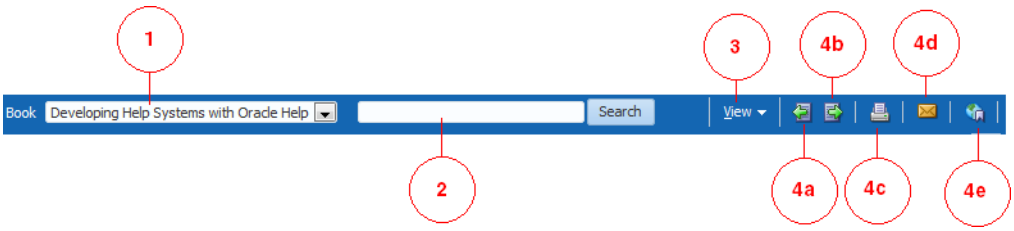


Figure 3–2 numbered callouts identify the following user interface components:

1. **Helpset Switcher**: A dropdown list that contains all helpsets defined in an OHW application. The helpset that is currently open is shown in the dropdown box, and users can use the list to switch to a different helpset. The Helpset Switcher is visible only when there are multiple independent sets of content and the `combineBooks` parameter, in the `ohwconfig.xml` file, is set to `false`.
2. **Search This Helpset**: A quick search control that allows users to perform a quick search without switching to the Search navigator. The default search options in this case are set as case insensitive, all words and all sources. If there are multiple search navigators, the quick search control is not displayed. Multiple search navigators can be present when the helpset file contains more than one views with the navigator of type:
`oracle.help.navigator.searchNavigator.SearchNavigator`.

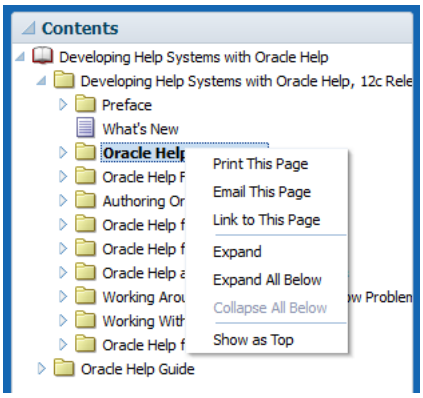
3. **View**: A menu that provides a list of command menu items to control visibility and navigation of the helpset. The command items are grouped into similar functionalities as follows:
   - **Maximize Reading Pane** toggles the expansion or collapse of the navigator pane.
   - **Restore Default Window Layout** rearranges the panes according to the default layout.
   - The navigator items section contains command items to navigate to all available navigators panes (**Contents**, **Index**, and **Search**, and the custom navigators, if any).
   - **Show bookmarkable link for this topic page** displays a popup that contains the permanent link to the current topic, which can then be used to create a bookmark.
4. The toolbar contains the following buttons:
   a. **Back To Topic**: reopens the topic the user previously visited.
   b. **Forward To Topic**: reopens the topic the user just returned from.
   c. **Print this topic page**: prints the current topic.
   d. **Email link to this topic page**: opens the default email application, copies the URL of the current topic into the message body, and the topic title into the subject field.
   e. **Show permanent link for this topic page**: displays a popup that contains the permanent link to the current topic, which can then be used to create a bookmark.

A **Collapse Pane** button is also available to hide the toolbar and maximize the Topic Pane.

3.3 OHW Contents Navigator

The Contents Navigator displays topics in a hierarchical tree. The contents and structure of the tree shows the merged data from the table of contents views in the loaded helpsets. Users can expand or collapse branches of the tree and select leaves (and branches that have associated topics). When a leaf or branch associated with a topic is selected, the topic is displayed in the Topic Pane on the right. A scroll bar, if required, appears when necessary.If the Table of Contents in the **Contents Navigator** is not visible, use the **View** menu in the global toolbar to navigate to the tab.

Figure 3–3 OHW Contents Navigator



The Contents Navigator is always synchronized with the topic displayed in the Topic Pane. For example, if you click a link in a topic and jump to another topic, the new topic is automatically highlighted in the **Contents** tab.The state of the **Contents** tab is maintained even when a user switches between navigators. For example, if a user selects a topic and switches to the Index navigator and back (without opening any other topic), the selected topic, scroll position, and expansion state of the tree remain unchanged. This works with the auto-synchronization functionality.The context menu provides a list of command menu items to control visibility and navigation of the helpset. The items are:

- **Print This Page**: prints the contents of the selected topic.
- **Email This Page**: opens the default email application, copies the URL of the current topic into the message body, and the topic title into the subject field (also available through the global toolbar).
- **Link to This Page**: displays a popup that contains the permanent link to the selected topic, which can then be used to create a bookmark (also available through the global toolbar).
- **Expand**: expands the tree to display the child topics of the selected node.
- **Collapse**: collapses the tree to hide the child topics of the selected node.
- **Expand All Below**: expands all nodes under the selected node to display their respective child topics.
- **Collapse All Below**: collapses all nodes under the selected node to hide their respective child topics.
- **Show As Top**: displays the current node as the root of the tree.

> **Note:** The functions provided by the first three context menu items are also available through the global toolbar. For more information, see Section 3.2, "OHW Global Toolbar".

3.4 OHW Index Navigator

The Index Navigator displays a sorted list of keywords, in a hierarchy of two levels. A keyword can be associated with multiple topics. The hierarchy is indicated by indented child items.

Figure 3–4 OHW Index Navigator

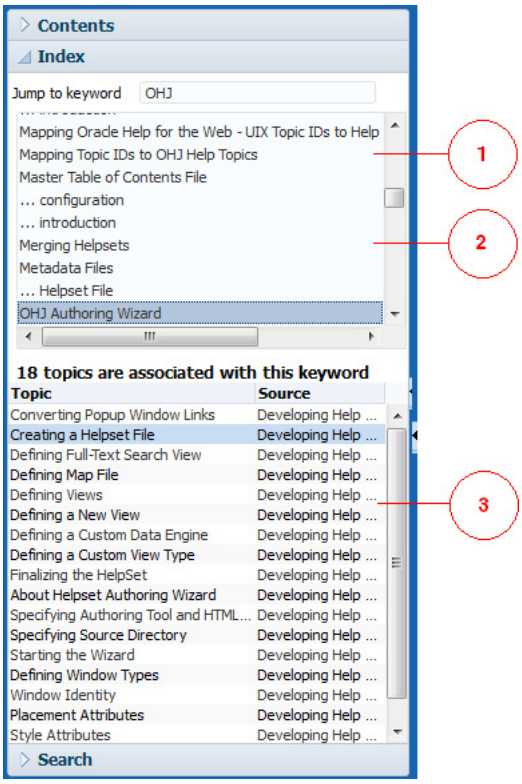


Figure 3–4 numbered callouts identify the following user interface components:

1. **Jump to keyword**: The user enters a word or words in this field. As the user types, the first keyword in the list that matches the typed letters is selected. As more letters are typed, a more accurate selection is made. Alternatively, the user can simply select a keyword from the keyword list.
2. **Keyword list**: A list that contains a 2-level locale-sensitive collated list of keywords. Selecting a keyword in the list displays the associated topics in the topic list.
3. **Topic list**: A list that displays the title and source of each topic that is associated with the keyword selected in the keyword list. When the user selects one of these titles, the topic is displayed in the Topic pane. Both the Topic and Source columns can be sorted in the ascending or descending alphabetical order.

3.5 OHW Search Navigator

The Search Navigator provides a user interface for constructing a full-text search query. The user may enter multiple words for the search string.

Figure 3–5 OHW Search Navigator

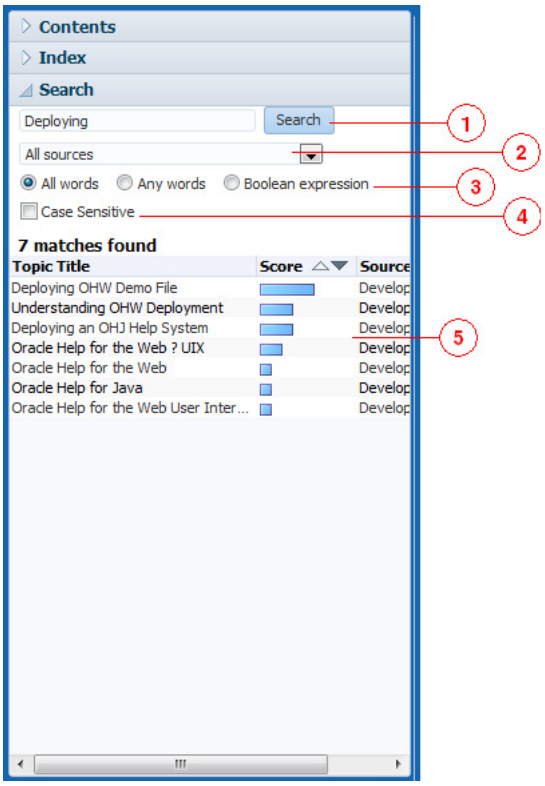


Figure 3–5 numbered callouts identify the following user interface components:

1. **Search text**: The user enters a word or words in this field. To search for an exact phrase, enclose the search phrase in double quotes.

   You can also use the * wildcard character in your search string. Note that the support for the * wildcard character is available in between and in the end of the search string, but not in the beginning of the search string. For example, `John*Doe` and `John*` are valid search strings, `*Doe` is an invalid search string.

2. **Source**: A dropdown that allows users to specify the set source topics among which the search is to be performed.

3. **Match**: A dropdown that allows users to specify whether to list topics that contain any or all of the specified words, or perform a search based on a Boolean expression (AND, OR).

4. **Case Sensitive**: A checkbox, when selected, enables searching for words having the same case as the words entered by the user.

5. **Topic List**: The results area displays the **Topic Title**, **Score** and **Source** of each topic that matches the search criteria. The **Score** column indicates the ranking of the topics according to how well they match the search criteria. All columns can be sorted in the ascending or descending alphabetical order. By default, all topics are sorted by **Score**. When a user hovers the mouse on an item, a tooltip appears to show the full names of relevant folders, topics, and source data. When the user selects a particular result, the associated topic is displayed in the topic pane.

The Oracle Help for the Web includes the Oracle Help Full Text Search indexer that adds word position information into the IDX files. The Oracle Help Full Text Search indexer enables exact phrase searches and improves the result ranking mechanism

(giving more points if the search terms are found near to each other). To search for an exact phrase, place the phrase in double quotes. For example, `"John Doe"`.

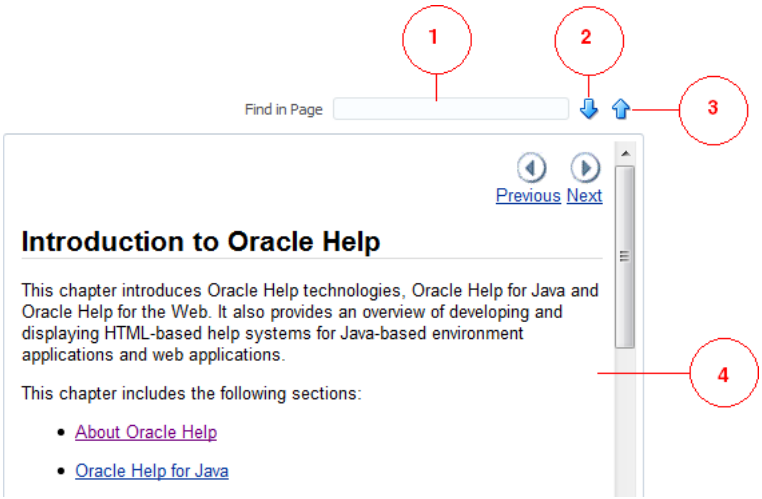
Oracle Help for the Web can also read IDX files created with previous versions of the Oracle Help Indexer, and enables you to use both old and new IDX files together, if required. Using an IDX file created with the new indexer, of a previous OHJ or OHW release, is not supported and raises a `SearchException` exception to be logged stating that the IDX version is not supported.

> **Note:** The IDX files created with Oracle Help Full Text Search indexer are not backward compatible. The indexer gives an error if you use an IDX file created with the current indexer, with an old version of Oracle Help for the Web.

3.6 OHW Topic Pane

The Topic pane displays the HTML help content.

Figure 3–6 OHW Topic Pane



The Topic pane has its own toolbar to manipulate the currently displayed topic. Figure 3–6 numbered callouts identify the following user interface components:

1. **Find in page**: The user enters some specific text, which is to be searched in the topic content, in this field.
2. **Next**: Searches for the next occurrence of the text specified by the user, and highlights the same if a match is found.
3. **Previous**: Searches for the previous occurrence of the text specified by the user, and highlights the same if a match is found.
4. **Topic Pane**: The pane with the help content.

3.7 Other OHW Features

OHW features, not otherwise mentioned in this Oracle Help Guide, include the following.

3.7.1 About OHW Feature

The *About OHW* feature displays information about the OHW help system, Oracle copyright information, and the OHW build you are using.

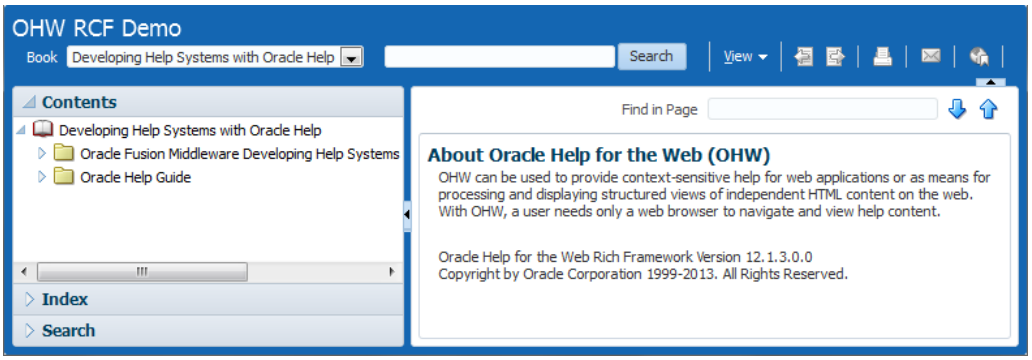
> **Note:** Oracle Help for Java supports the **About OHW** feature using the request query parameter `aboutOHW`.

To view the About OHW page, the user must navigate to the following URL:

```
http://localhost:<port_number>/<help ohw-rcf
context-root>/faces/helppages/main.jspx?config=OHW+Servlet+1&aboutOHW=true
```

Upon navigating to this URL, the Topic pane displays the current OHW build information along with copyright information.

Figure 3–7 About OHW Window



3.7.2 Bookmarking Feature

Users can bookmark a page, and the same topic is displayed when navigating to the bookmark. Users can also save a URL link and send the link by email. The following points must be noted for Bookmarking feature:

- To bookmark a page, users must use the **Show permanent link for this topic** page button on the global toolbar, which refreshes the rich client URL to a bookmarkable URL. Users can then use then browser's bookmarking feature to save the URL.
- The state of the **Index** and **Search** navigators are not a part of the bookmark information. The **Contents** navigator is automatically in sync with the displayed topic.

3.7.2.1 Backward Compatibility

A bookmark from OHW can display the same topic view in OHW given the topic ID, group, and locale. However, as OHW bookmarks have many other parameters besides the ones that are supported, the state of all navigators may not be restored.

OHW URLs indexed by search engines continue to link to valid help content.

3.7.3 Single Pane Layout Feature

Users can view the current topic in a single pane layout by altering the URL.

For example:

topicId

```
http://localhost:7101/help-ohw-rcf-context-root/ohguide/?topic=ohg_about_
about_html&linkHelp=false
```

vtTopicFile

```
http://localhost:7101/help-ohw-rcf-context-root/ohguide/?vtTopicFile=ohgui
de/ohg_about_about.html&linkHelp=false
```

Part II

Oracle Help File Formats

This part describes the Oracle Help file formats. It contains the following chapters:

- Chapter 4, "Introduction to Oracle Help File Formats"

  This chapter provides an introduction to Oracle Help file formats.

- Chapter 5, "Metadata Files"

  This chapter describes the metadata files used in Oracle Help. These include helpset files and map files.

- Chapter 6, "Help Information Files"

  This chapter describes the information files used in Oracle Help. These include Table of Contents file, Index file, Search Index file, and Link file.

- Chapter 7, "Topic Files"

  This chapter describes the topic files used in Oracle Help.

- Chapter 8, "Oracle Help for the Web Configuration File"

  This chapter describes the configuration files of Oracle Help for Web.

4

Introduction to Oracle Help File Formats

This chapter describes the various file formats (such as metadata files, information files, and topic files) used by Oracle Help for Java (OHJ) and Oracle Help for the Web (OHW).

This chapter includes the following sections:

4.1 About Oracle Help File Formats

The following help file formats are used by OHJ and OHW:

- Metadata files provide information about the structure and operation of the help system. For more information, see Chapter 5, "Metadata Files".
- Help information files contain information about the content of the help system. For more information, see Chapter 6, "Help Information Files".
- Topic files are the HTML files containing the content for the help system. For more information, see Chapter 7, "Topic Files".

In addition to the above files, OHW uses a configuration file to configure the servlet filter. For more information, see Chapter 8, "Oracle Help for the Web Configuration File".

The Oracle Help file formats are based on the JavaHelp™ specification. For more information about differences between Java Help and Oracle Help file formats, see Appendix A, "Oracle Help and JavaHelp File Formats".

4.2 File Name Extensions

When you use helpsets, you do not have to use specific extensions for the names of the associated control files, but you must ensure that the correct file name and extensions are used when the file is referenced.

Table 4–1 shows conventional (but not required) extensions for the helpset-related file formats. It also shows where these files are referenced, so you ensure that the correct name and extension are used.

Table 4–1 OHJ and OHW Files

Type of File	Conventional Extension	Referenced By
Helpset	.hs	In OHJ, the calls from the Java program that launches the help. In OHW, from the configuration file: <books> <helpSet location="filename.hs" /> </books>
Helpset used as a subhelpset	.hs	Master helpset file in <subhelpset location="filename.hs"/>
Map	.xml	Helpset file in <maps> <mapref>filename.xml</mapref> </maps>
TOC	.xml	Helpset file in <maps> <view> <data engine="oracle.help.engine.XMLTOCEngine">filename.xml</data> </view>
Keyword Index	.xml	Helpset file in <view> <data engine="oracle.help.engine.XMLIndexEngine">filename.xml</data> </view>
Link	.xml	Helpset file in <links> <linkref>filename.xml</linkref> </links>
Search Index	.idx	Helpset file in <view> <data engine="oracle.help.engine.SeearchEngine">filename.idx</data> </view>

5

Metadata Files

This chapter describes the metadata files (helpset and map files) used by OHJ and OHW. *Metadata files* provide information about the structure and operation of the help system.

This chapter includes the following sections:

5.1 About Metadata Files

Metadata files can be categorized into two types, Helpset files and Map files. Both files are XML files.

5.2 Helpset File

The helpset file is an XML file with `.hs` extension that organizes project-level information about the helpset. For example, it points to other control files to be used for the helpset, including map, table of contents, index, associative links, and search. These references are used, in part to define the set of navigational views that Oracle Help uses to construct the user interface.

The helpset file consists of the following elements and their child elements:

5.2.1 The <helpset> Element

The contents of a helpset file are entirely contained within a single `<helpset>` element.That is, a helpset file must begin with `<helpset>` and end with `</helpset>`. Only one `<helpset>` element is allowed in a helpset file.

5.2.2 The <title> Element

The `<title>` element assigns a name to the helpset, for example:

```
<title>API Reference</title>
```

Under certain conditions, this title is displayed in the Oracle Help user interface as the name of the helpset. For example, it is displayed in the dropdown list of helpsets when Oracle Help is implemented to display a list of merged helpsets, instead of concatenating them.

5.2.3 The <maps> Element

The `<maps>` element points to one or more map files, which are used to map topic IDs to topic files. The `<maps>` element has the following child elements:

Table 5–1 <maps> Child Elements

Element	Description
`<mapref>`	The location of a map file for this book. When multiple map files are specified, they are merged. Each `<mapref>` item is searched in order and a merged map is constructed. The `<mapref>` element has the following attributes: ■ `location` – The URL of the map file relative to the helpset. ■ `class` – [optional] A class whose location is the base location for the map file. Any path information in the location attribute is relative to this base location.
`<homeID>`	The ID (defined in the map file) of a topic that is used in some cases as the default topic for the helpset. That is, if no topic is specified when OHJ is launched, this topic is displayed by default. The `<homeID>` element is functional only in the following OHJ implementations: ■ OHJ Documentation Viewer (installed with OHJ). ■ When OHJ is implemented so that the navigator window and the topic window are docked by default. The `<homeID>` element is not used in OHW.

For example:

```
<maps>
  <mapref location="main_map.xml"/>
  <mapref location="tutorial/tut_map.xml"/>
</maps>
```

5.2.4 The <wintype> Element

The `<wintype>` element defines the characteristics of one or more windows that can be used by OHJ to display topics, including size, screen placement, text color, and background color. For more information about how this looks in the OHJ user interface, see Section 2.2, "OHJ Topic Windows".

Note: OHW does not recognize <wintype> element. This information is also ignored if topics are displayed directly in a web browser.

The helpset file can have any number of <wintype> sections; one for each window type. The <wintype> tag has one valid attribute:

- default – If true, the current window type is the default window type to be used if a topic file does not designate a window type. If false, this window type is not a default and is not used unless explicitly listed in the map file. When the default attribute is not present, its value is assumed to be false.

The <wintype> element can have the following child elements:

Table 5–2 <wintype> Child Elements

Element	Description
<name>	The name of this window type. This name is used to associate topics with the window type. If multiple windows are defined in a helpset, they must have unique names.
<height>	The height of the window. A numeric value indicates a distance in pixels. A numeric value followed by a percent sign (%) indicates a percentage of the visible screen.
<width>	The width of the window in pixels. A numeric value indicates a distance in pixels. A numeric value followed by a percent sign (%) indicates a percentage of the visible screen.
<x>	The horizontal position of the window. A numeric value indicates a distance in pixels. A negative value indicates distance between the right edge of the window and the right edge of the screen. A numeric value followed by a percent sign (%) indicates a percentage of the visible screen.
<y>	The vertical position of the window in pixels. A numeric value indicates a distance in pixels. A negative value indicates distance between the bottom edge of the window and the bottom of the screen. A numeric value followed by a percent sign (%) indicates a percentage of the visible screen.
<textfg>	The six-digit hexadecimal RGB value of the foreground color of the text in the window. A preceding # on the value may be present, but is ignored. For any single topic, a foreground color specified in the HTML topic file or CSS overrides this value.
<linkfg>	The six -digit hexadecimal RGB value of the foreground color of the links in the window. A preceding # on the value may be present but is ignored. For any single topic, a link color specified in the HTML topic file or CSS overrides this value.
<bg>	The six -digit hexadecimal RGB value of the background color of the window. A preceding # on the value may be present but is ignored. For any single topic, a background color specified in the HTML topic file or CSS overrides this value.
<title>	Text that appears in the title bar of the window.

Table 5–2 (Cont.) <wintype> Child Elements

Element	Description
<toolbar>	Defines the buttons to display in the window's toolbar. A five-digit hexadecimal value which is the sum of one or more of the following: ■ **00004** – Turns off the default buttons on the toolbar. ■ **00040** – Adds a URL address field to display the current URL. ■ **00400** – Add the Navigator button. ■ **02000** – Add the Print button ■ **04000** – Add the Back and Forward buttons. ■ **08000** – Add the Search button. ■ **10000** – Add the Dock/Undock buttons A preceding # on the value may be present, but it is ignored.

Note that attributes specified in HTML topic content files always take precedence over attributes specified in the <wintype> section of the helpset.

The following <wintype> element defines a window to be used for tutorial topics:

```
<wintype default=false>
  <name>Tutorial</name>
  <height>50%</height>
  <width>200</width>
  <x>10</x>
  <y>10</y>
  <textfg>000000</textfg>
  <linkfg>0000cc</linkfg>
  <bg>ffffff</bg>
  <title>Tutorial</title>
  <toolbar>06004</toolbar>
</wintype>
```

This example defines the following:

- a window type called Tutorial with a default height that is 50% of the screen height and a default width of 200 pixels
- window is displayed 10 pixels from the top and left side of the screen
- window background is white, with black text and blue links
- window title bar reads **Tutorial**
- window toolbar contains the **Print**, **Back**, and **Forward** buttons

5.2.5 The <links> Element

The <links> element points to one or more link files, which are used to associate multiple targets with link IDs. For more information about links, see Section 6.6, "Link File".

The <links> element has the following child element:

Table 5–3 <links> Child Elements

Element	Description
<linkref>	The location of a map file for this book. When multiple map files are specified, they are merged. Each <linkref> item is searched in order and a merged link database is constructed. <linkref> has the following attributes: ■ location – The URL of the link file relative to the helpset. ■ class – [optional] A class whose location is the base location for the map file. Any path information in the location attribute is relative to this base location.

For example:

```
<links>
  <linkref location="linkfile1.xml"/>
  <linkref location="linkfile2.xml"/>
</links>
```

5.2.6 The <view> Element

The <view> element specifies how Oracle Help should render navigational views. A navigational view is a representation of the data in the navigation control files (such as TOC, index, and search) plus the user interface controls for navigating through them. Oracle Help includes Java classes to render standard types of views. Each type of view is presented on its own accordion panel: by default, the **Contents**, **Index**, and **Search** panels.

A helpset can include multiple views of each type. That is, the user interface can display multiple TOC tabs, Index tabs, and so forth. Oracle Help can also merge views of the same type. It does this by merging all views of the same type that have the same label. Views with the same type and label are merged as follows:

- TOC views are appended together.
- Index views are merged in a sorted order.
- Search views are merged in no order. Order is irrelevant, since all search results are sorted by the user interface.

The <view> element can have the following child elements:

Table 5–4 <view> Child Elements

Element	Description
<label>	The label displayed on the navigator tab in the user interface. This label is optional. If no label is provided, Oracle Help uses Contents, Index, and Search for the appropriate tabs. The <label> element supports the following attribute: ■ image - [optional] An ID in the map file that is associated with an image file. The image appears next to the label text in the navigator tab.

Table 5–4 (Cont.) <view> Child Elements

Element	Description
`<title>`	A title for the view. This title appears in different places in the different navigational views: ■ In the TOC, the title is displayed as the title of the top-level node. All top-level nodes from the TOC file (or from multiple TOC files, if they are merged) are then placed under this title node. If a title is not provided for a TOC view, then all top-level nodes in the TOC definition appear as top-level nodes in the TOC view. ■ In the search and index views, the title is displayed as the Source location for any topics listed as the result of a search or from choosing an index item. If a title is not provided for a search or index view, the Source location is blank. The title tag supports the following attribute: ■ `image` - [optional] An ID in the map file that is associated with an image file. If provided for a table-of-contents view, the image appears next to the text for the top-level node. If provided for an index or search view, the image is ignored.

`<type>`	The name of the Java class to be used as the user interface for this view. The following types are provided with the default implementation:
	`oracle.help.navigator.tocNavigator.TOCNavigator`
	`oracle.help.navigator.keywordNavigator.KeywordNavigator`
	`oracle.help.navigator.searchNavigator.SearchNavigator`
	If JavaHelp types are specified, Oracle Help maps these correctly to Oracle Help types.

`<data>`	The path to the data used by this view, in other words, to the pertinent navigational control file, such as table-of-contents file, index file, or search file. The contents of this element can be a file name or a URL. The `<data>` element supports the following attributes: ■ `class` – [optional] A class whose location is the base location for the data file. Any path information is relative to this base location, including any references to HTML files. ■ `engine` – The name of the Java class to process the data file. A different engine is required for each of the file types supported by Oracle Help, including Oracle Help's XML formats (based on the JavaHelp formats), the MS HTML Help HHC/HHK file formats, and the TOC/TOK formats supported by earlier versions of OHJ. For more information about supported engines and view type, see Section 5.2.6.1, "Data View Type and Engines".

In the following example, two TOC tabs are created, one labeled **User's Guide** and the other labeled **Reference**. They both use the XML file format for the table of contents (specified in the `<data>` element), but they have different values in the `<label>` element. If both of the labels were the same, the TOC files, `ug_toc.xml` and `ref_toc.xml`, would be combined into a single TOC, and it would be shown in a single tab.

```
<view>
  <label>User's Guide</label>
  <type>oracle.help.navigator.tocNavigator.TOCNavigator</type>
  <data engine="oracle.help.engine.XMLTOCEngine">ug_toc.xml</data>
</view>
<view>
```

```
  <label>Reference</label>
  <type>oracle.help.navigator.tocNavigator.TOCNavigator</type>
  <data engine="oracle.help.engine.XMLTOCEngine">ref_toc.xml</data>
</view>
```

In the following example code, only one Contents tab is created, even though the two views use different file formats (XML compared to HHC) and different data engines. That is because the label and the type are the same. One advantage of this feature is that you can merge help systems using the old format with help systems using the new one without compromising the way the tabs are presented.

```
<view>
  <label>Table of Contents</label>
  <type>oracle.help.navigator.tocNavigator.TOCNavigator</type>
  <data engine="oracle.help.engine.XMLTOCEngine">new_toc.xml</data>
</view>
<view>
  <label>Table of Contents</label>
  <type>oracle.help.navigator.tocNavigator.TOCNavigator</type>
  <data engine="oracle.help.engine.HHCEngine"old_toc.hhc</data>
</view>
```

The next example shows one view each for a table of contents, an index, and a search. They do not have labels, so Oracle Help creates three tabs with the default labels, **Contents**, **Index**, and **Search**. Each view contains a `<title>` element with the value User's Guide. This produces the following results:

- In the **Contents** tab, the items from the `ug_toc.xml` table of contents file appear under a single top-level node called User's Guide.
- When a user selects an item from the keyword list in the **Index** tab, a list of associated topics is displayed. **User's Guide** appears as the source for any topic from this helpset.
- When a user performs a search in the **Search** tab, **User's Guide** is shown in the list of results as the source for any topics found from this helpset.

This feature is useful when you merge several helpsets. It helps to keep the user oriented by reducing the number of top-level nodes in the table of contents and by showing the sources of topics found when using the index and search.

```
<view>
  <title>User's Guide</title>
  <type>oracle.help.navigator.tocNavigator.TOCNavigator</type>
  <data engine="oracle.help.engine.XMLTOCEngine">ug_toc.xml</data>
</view>
<view>
  <title>User's Guide</title>
  <type>oracle.help.navigator.keywordNavigator.KeywordNavigator</type>
  <data engine="oracle.help.engine.XMLIndexEngine">ugindex.xml</data>
</view>
<view>
  <title>User's Guide</title>
  <type>oracle.help.navigator.searchNavigator.SearchNavigator</type>
  <data engine="oracle.help.engine.SearchEngine">search.idx</data>
</view>
```

5.2.6.1 Data View Type and Engines

The following tables lists valid engine values for each view type:

Table 5–5 View Type: oracle.help.navigator.tocNavigator.TOCNavigator

Engine	Description
`oracle.help.engine.XMLTOCEngine` (default)	Oracle Help XML table of contents (extension of JavaHelp TOC)
`oracle.help.engine.HHCEngine`	Microsoft HTMLHelp 1.x table of contents
`oracle.help.engine.TOCEngine`	Table of contents from previous versions of OHJ

Table 5–6 View Type: oracle.help.navigator.keywordNavigator.KeywordNavigator

Engine	Description
`oracle.help.engine.XMLIndexEngine` (default)	Oracle Help XML keyword index (extension of JavaHelp keyword index)
`oracle.help.engine.HHKEngine`	Microsoft HTMLHelp keyword index.
`oracle.help.engine.TOKEngine`	Keyword index from previous versions of OHJ

Table 5–7 View Type: oracle.help.navigator.searchNavigator.SearchNavigator

Engine	Description
`oracle.help.engine.SearchEngine` (default)	Oracle help text search database.

5.2.7 The <subhelpset> Element

The `<subhelpset>` element is used to include other helpsets with the one defined in this helpset file. The views from combined subhelpsets are merged in the same way multiple views are merged in a single helpset. That is, subhelpset views with the same `<type>` and `<label>` are merged. For more information, see Section 5.2.6, "The <view> Element".

The `<subhelpset>` element supports the following attributes:

- `location` – Specifies the URL of the helpset to be merged.
- `class` – [optional] A class whose location is the base location for the subhelpset file. Any path information in the location attribute is relative to this base location.

Oracle Help assumes that subhelpsets may not always be present. For example, a master helpset for a suite of products may have a subhelpset for each product in the suite: product A, product B, and product C. The user might initially install just product A. At a later time, the user might install product C. Subhelpsets aid in this situation, because you can specify subhelpsets that are loaded if they are found, and are ignored if they are not found.

5.2.8 Sample Helpset File

The following listing shows a sample helpset with all of the sections discussed above, including five views: two tables of contents, two keyword indexes, and a text search.

```
<?xml version='1.0'?>
<helpset version="1.1">
```

```
<maps>
  <mapref location="topics.xml"/>
</maps>

<wintype>
  <name>Tutorial</name>
  <height>50%</height>
  <width>200</width>
  <x>10</x>
  <y>10</y>
  <title>Tutorial</title>
  <toolbar>06004</toolbar>
</wintype>

<links>
  <linkref location="linkfile1.xml"/>
  <linkref location="linkfile2.xml"/>
</links>

<view>
  <label image="tocgif">Table of Contents</label>
  <title image="uggif">Forms User's Guide</title>
  <type>oracle.help.navigator.tocNavigator.TOCNavigator</type>
  <data engine="oracle.help.engine.XMLTOCEngine">ugcontents.xml</data>
</view>

<view>
  <label>Keyword Index</label>
  <type>oracle.help.navigator.keywordNavigator.KeywordNavigator</type>
  <data engine="oracle.help.engine.XMLIndexEngine">ugindex.xml</data>
</view>

<view>
  <label image="tocgif">Table of Contents</label>
  <title image="dggif">Forms Developer's Guide</title>
  <type>oracle.help.navigator.tocNavigator.TOCNavigator</type>
  <data engine="oracle.help.engine.HHCEngine"
    class="oracle.forms.documentation.dev">
    dgcontents.hhc
  </data>
</view>

<view>
  <label>Keyword Index</label>
  <type>oracle.help.navigator.keywordNavigator.KeywordNavigator</type>
  <data engine="oracle.help.engine.HHKEngine"
    class="oracle.forms.documentation.dev">
    dgindex.hhk</data>
</view>

<view>
  <label>Search</label>
  <title>Forms Documentation</title>
  <type>oracle.help.navigator.searchNavigator.SearchNavigator</type>
  <data engine="oracle.help.engine.SearchEngine"
    class="oracle.forms.documentation.dev">
    search.idx</data>
</view>

<subhelpset location="prodsup.hs"/>
```

```
  <subhelpset location="advanced.hs"/>

</helpset>
```

5.3 Map Files

The map file is an XML file that associates IDs with files. The primary use of the map file is to define topic IDs and associate them with topic files. You can also associate topic IDs (and thereby the topics) with any window types defined in <wintype> elements in the helpset file. These IDs are used in the table of contents files, in index files, and in the API for context-sensitive calls.

The map file can also be used to define image IDs and associate them with image files. You can use these image IDs to display images next to tab labels (specified in <view> elements in the helpset file). They can also be used to display images next to items in the table of contents specified in <tocitem> elements in the table of contents file.

5.3.1 Map File Elements

The following table describes the elements used in the map file:

Table 5–8 Map File Elements

Element	Description
<map>	Defines the mappings. The <map> element contains only <mapID> elements.
<mapID>	An ID and its associations. The <mapID> element has the following attributes: ■ target – The ID for the associated file. Valid characters are any alphanumeric character or punctuation symbol except the equals sign (=). IDs must be unique across all helpsets in a given help instance. However, you can assign multiple IDs to a single file. For example, you could assign one ID for a topic file to appear in one window type and a different ID for the same topic file to appear in a different window type. ■ url – The location of the file to associate with this ID (target). Relative URLs are resolved against the absolute URL for the map file. You may also specify a section of the file using anchor links. ■ wintype – [optional] The name of a windows type, defined using <wintype> elements in the helpset file. If a windows type is not specified, the topic associated with this ID is displayed in the default window defined in the helpset. To open the topic links in a native browser window, specify wintype="external" for all the relevant topics in map.xml file. The attribute is applicable for OHJ only.

In the following example, the map IDs topic_1 and topic_2 are not associated with window types and therefore use the helpset's default window type. The map IDs topic_3 and topic_4 map to topic files displayed in the window defined by the intro window type. Map ID topic_5.tsk displays File_5.html in the window defined by the task window type. Map ID topic_5.cncpt displays the same topic file (File_5.html) in a different window type (concept). The association between URL and wintype is be used when linking from topic to topic using URLs instead of topic IDs. For example, if a topic had a hard-coded target to File_5.html, clicking the link would display the HTML content in a task window type; and if a topic had a hard-coded

target to File_6.html, clicking the link would display the HTML content in the native browser window.

```
<?xml version='1.0' ?>

<map version="1.0">
    <mapID target="topic_1" url="file_1.html" />
    <mapID target="topic_2" url="file_2.html#a1" />
    <mapID target="topic_3" url="file_3.html" wintype="intro" />
    <mapID target="topic_4" url="file_4.html#a2" wintype="intro" />
    <mapID target="topic_5.tsk" url="file_5.html" wintype="task" />
    <mapID target="topic_5.cncpt" url="file_5.html" wintype="concept" />
    <mapID target="topic_6" url="file_6.html" wintype="external" />
</map>
```

This scheme allows authors to assign window types to HTML files and to also override those associations by linking to an alternate topic ID. For example, for topic-to-topic links, TOC links, index links, and hard-coded links to File_5.html, the author might use topic_5.tsk, but for links from a tutorial, the author might use topic_5.cncpt. By keeping this information in the map file, the author has one central repository for managing these assignments.

If the url attribute specifies an external location (or external URL), the target file will open in a new browser window. For example, if a TOC link uses topic_ext as the topic ID, the link will open the OTN page in a new browser window.

```
<mapID target="topic_ext" url="http://www.oracle.com/technetwork/index.html" />
```

6 Help Information Files

This chapter describes the help information files (table of contents, index, and link files) used by OHJ and OHW. *Help information files* contain information about the content of the help system.

This chapter includes the following sections:

6.1 About Help Information Files

Help Information files can be categorized into three types, Table of Contents files, Index files, and Link files. All these files are XML files.

6.2 Table of Contents File

The Table of Contents (TOC) file is an XML file that describes the content and layout of the table of contents, and it is typically rendered as the **Contents** tab. Note that each helpset jar file must have one Table of Contents file.

If a helpset jar file contains several child helpsets, use the master Table of Contents file to control the content and layout of parent helpset's table of contents.

6.2.1 TOC Elements

Table 6–1 describes the elements used in the table of contents file:

Table 6–1 TOC File Elements

Element	Description
`<toc>`	Defines the table of contents. This element contains `<tocitem>` elements. The `<toc>` element supports the following attribute: ■ `version` - The version of the table of contents file format specification. The version documented here is 1.0.

Table 6–1 (Cont.) TOC File Elements

Element	Description
`<tocitem>`	Defines a table of contents entry. A `<tocitem>` element can contain other `<tocitem>` elements to create a hierarchy of topics. Nesting "tocitem_1" within "tocitem_2" defines "tocitem_1" to be hierarchically contained within "tocitem_2." The `<tocitem>` element supports the following attributes: ■ `target` - The topic ID of the topic to display when this item is selected by a user. Topic IDs are maintained in the map file. A parent node (that is, an item that contains others beneath it in the hierarchy) may or may not include a target. If it does not have a target, selecting that node displays its child items and nodes. To open an external URL from a table of contents entry, use a topic ID with `url` attribute set to the desired external URL. Note that all external URLs automatically open in a new browser window. For more information, see Section 5.3.1, "Map File Elements." ■ `text` - The text displayed in the table of contents for this item. ■ `image` - An image to display next to this item. The value of this attribute is an image ID specified in the map file.

The following example shows a short table of contents file:

```
<?xml version='1.0'?>
  <toc version="1.0">
  <tocitem text="Introduction to My Product" image="prodIcon">
    <tocitem target="aboutmyp" text="About My Product" />
    <tocitem target="architec" text="My Product Architecture" />
    <tocitem target="feature1" text="About Feature One" />
    <tocitem target="feature2" text="About Feature Two" />
    <tocitem target="feature3" text="About Feature Three" />
    <tocitem text="Learning My product">
      <tocitem target="qtour" text="Quick Tour" />
      <tocitem target="docguide" text="Guide to Documentation" />
    </tocitem>
  </tocitem>
  <tocitem text="Step-by-step Procedures" image="prodIcon">
    <tocitem text="Basic Tasks">
      <tocitem target="task1" text="To..." />
      <tocitem target="task2" text="To..." />
      <tocitem target="task3" text="To..." />
    </tocitem>
    <tocitem text="Working with ...">
      <tocitem target="task4" text="To..." />
      <tocitem target="task5" text="To..." />
    </tocitem>
  </tocitem>
  </toc>
```

This definition produces the following TOC hierarchy:

- Introduction to My Product
  - About My Product
  - My Product Architecture
  - About Feature One

  - About Feature Two
  - About Feature Three
  - Learning My Product
    * Quick Tour
    * Guide to Documentation
- Step-by-step Procedures
  - Basic Tasks
    * To...
    * To...
    * To...
  - Working with ...
    * To ...
    * To ...

6.3 Master Table of Contents

A master table of contents file controls the TOC structure of all child helpsets, and manages other table of contents files. You must declare the master table of contents before declaring helpset files.

6.3.1 How To Configure Master Table of Contents

To use a master table of contents, you must first configure the Oracle Help for the Web configuration file.

1. Declare master table of contents helpset file in the Oracle Help for the Web configuration file.

   The following example shows a code extract from the configuration file:

   ```
   <books>
     <helpSet id="master" location="master/src/helpset/master.hs"/>
     <helpSet id="ohguide" location="ohguide/src/helpset/ohguide.hs"/>
     <helpSet id="blafdoc" location="blafdoc/src/helpset/blafdoc.hs"/>
     <helpSet id="shake" location="shakespeare/src/helpset/shakespeare.hs"/>
   </books>
   ```

   In the above example, the following code declares a master table of contents helpset file:

   ```
   <helpSet id="master" location="master/src/helpset/master.hs"/>
   ```

   Note that the master table of contents helpset file must be declared first before any other helpset files.

2. Create a master helpset file.

   The entry for master table of contents helpset looks similar to any other TOC entry, as shown in the following code extract example of `master.hs` file.

   ```
   <view>
     <type>oracle.help.navigator.tocNavigator.TOCNavigator</type>
     <data engine="oracle.help.engine.XMLTOCEngine">toc.xml</data>
   </view>
   ```

3. Create a master table of contents XML file that defines the structure of all table of contents.

   This master table of contents XML file looks similar to any other `toc.xml` file, but it contains `<tocfile>` elements. The `<tocfile>` elements define the location of child table of contents XML files, as shown in the following example.

```
<toc version="1.0">
 <tocitem text="Root">
   <tocitem text="Some random node">
     <tocfile location="ohguide/toc.xml" />
     <tocfile location="javadoc/toc.xml" />
   </tocitem>
   <tocfile location="blafdoc/toc.xml;shakespeare/shakespeare.hhc" />
 </tocitem>
 <!-- All helpset toc files not explicitly declared in this file will be
      inserted under the following node. This includes toc files that are
      located in a subhelpset. -->
 <tocitem text="Extension Help Content">
   <tocfile location="*" />
 </tocitem>
</toc>
```

   Note that the master table of contents file controls the structure of the content, you must define your content through helpset files.

6.4 Index File

The index file is an XML file that describes the content and layout of the index. It is typically rendered as the **Index** tab.

6.4.1 Index Elements

Table 6–2 describes the elements used in the index file.

Table 6–2 Index File Elements

Element	Description
`<index>`	Defines the index. It can contain `<indexitem>` and `<indexentry>` tags.
`<indexitem>`	Defines an index item that appears in the keyword list. Nesting index_item_1 within index_item_2 defines index_item_1 to be hierarchically contained within index_item_2, listed and indented below index_item_2 in the index. Oracle Help currently supports only two levels of keywords. The index view collapses any nesting beyond two levels. If an index item has multiple topics associated with it, the topics should be listed as index entries defined in `<indexentry>` elements. The `<indexitem>` element has the following attribute: ■ `target` - The topic ID (defined in the map file) of the topic to display when the entry is chosen by the user.

Table 6–2 (Cont.) Index File Elements

Element	Description
<indexentry>	Defines an index entry displayed in the topics list when the parent index item is selected in the index list. This tag uses the following attribute: ■ target - The topic ID (defined in the map file) of the topic to display when the entry is chosen by the user.

This example defines a very short index file:

```
<?xml version='1.0' ?>
  <index version="1.0">

  <indexitem target="Add_Icon"
      text="Add Icon Command">
    <indexitem target="addtosheet"
        text="Adding an icon to a sheet />
    <indexitem target="addtoworkbook"
        text="Adding an icon to a workbook" />
  </indexitem>

  <indexitem target="Sheet_Background"
       text="Adding a background to a sheet" />

  <indexitem text="Adding a new sheet to a workbook">
    <indexentry target="New_Sheet_command"
        text="New Sheet Command />
    <indexentry target="Add_new_sheet"
        text="To add a new sheet to a workbook" />
  </indexitem>

  <indexitem target="Add_item_to_sheet"
      text="Adding an item to a sheet" />
  </index>
```

The file in the example above produces this index list:

```
Add Icon Command
    Adding an icon to a sheet
    Adding an icon to a workbook
  Adding a background to a sheet
  Adding a new sheet to a workbook
  Adding an item to a sheet
```

If a user selects *Adding a new sheet to a workbook* from that list, a list of the following topics is displayed:

```
New Sheet Command
  To add a new sheet to a workbook
```

This topic list appears at the bottom of the index pane, as opposed to indented topics, *Adding an icon to a sheet* and *Adding an icon to a workbook*, which appears in the keyword list at the top of the index pane.

Selecting a keyword that does not have index entries but has a directly associated target (for example, *Adding an item to a sheet*) displays the same text in the topics list as it does in the keyword list. Because of the way Oracle Help displays the index, it is a better practice never to use targets in the <indexitem> tags. Instead, always use the

<indexentry> tags to specify topics associated with an <indexitem> -- even when there's only one target for a keyword.

In other words, the following code:

```
<indexitem text="sheet backgrounds">
    <indexentry target="Sheet_Background" text="adding a background to a sheet" />
</indexitem>
```

...is better than the following code:

```
<indexitem target="Sheet_Background" text="adding a background to a sheet" />
```

6.5 Search Index File

The search index file is used when a user performs a text search in Oracle Help, ordinarily from the Search tab. This file uses a proprietary binary format. Any third-party help authoring tool that supports Oracle Help should be able to generate this file. In addition, the OHJDK includes two utilities that generate a search index file:

- The Helpset Authoring Wizard provides limited authoring support, including generating search indexes. For more information, see Chapter 9, "Authoring Oracle Help Systems".
- The Text Search Indexer is a Java command-line tool that generates search indexes. For more information, see Chapter 11, "Using the Text Search Indexer".

6.6 Link File

The link file is an XML file that defines link IDs and associates them with multiple topic IDs (which are defined in the map file). A link ID, or a link keyword, can be used with the `alink` protocol in a topic file to display a list of links to the topics associated with the ID. In other words, associative links make it possible to associate an HTML link with multiple targets. The user can then choose which target to follow.

6.6.1 Link File Elements

Table 6–3 describes the elements used in the link file:

Table 6–3 Link File Elements

Element	Description
<link>	Defines the link file. The <link> element can contain only <linkitem> elements and their child <linkentry> elements.
<linkitem>	Defines the associative link. The <linkitem> element can only include <linkentry> elements. It has the following required attribute: ■ topicid – Defines the link ID.
<linkentry>	Defines an entry in the list of links displayed when an associative link is clicked. This element supports the following attributes: ■ target – Specifies the topic ID (defined in the map file) of the topic to display when the entry is chosen by the user. ■ text – The text displayed in the link list

The following example defines two associative links:

```
<?xml version='1.0' ?>
  <link version="1.0">

  <linkitem topic="dog_links">
    <linkentry target="about_dogs" text="About Dogs" />
    <linkentry target="dog_species" text="A List of Dog Species" />
    <linkentry target="dog_stories" text="Dog Stories" />
    <linkentry target="dog_lore" text="Dog Lore" />
  </linkitem>

  <linkitem topic="cat_links">
    <linkentry target="about_cats" text="About Cats" />
    <linkentry target="cat_species" text="A List of Cat Species" />
    <linkentry target="cat_stories" text="Cat Stories" />
  </linkitem>
  </link>
```

Using the first link ID from that example, you could define a link `<a href="alink:dog_links">dogs</a>`. When a user selected the **dogs** link, the following list of links would be displayed:

About Dogs
A List of Dog Species
Dog Stories
Dog Lore

Clicking *`About Dogs`* would display the topic mapped to the `about_dogs` ID in the map file.

7

Topic Files

This chapter describes the topic files, topic ID and associative links, popups, window types, and custom protocol links used by OHJ and OHW.

This chapter includes the following sections:

7.1 About Topic Files

Topic files are HTML files that contain the content for a help topic. The features supported in the HTML files depend on the browser (or HTML display component) used to display them.

The HTML display component that ships with OHJ is an Oracle-modified version of the ICEbrowser from ICEsoft Technologies, Inc. Versions 5.01 and later of the ICEbrowser are compliant with the HTML 4.0 standard and can display tables and frames also runs Java applets. However, the standard ICEbrowser does not support several features important for a full-featured help system. Therefore, under license from ICEsoft Technologies, Oracle has modified the ICEbrowser to support conventions developed by Oracle to provide the following features:

For these features to work in OHJ, you must use the default ICEbrowser version of OHJ. If you substitute a different HTML display component, these features will not work. Also, if you display your HTML topic files directly in a web browser, that browser will not recognize the protocols and metadata documented on this page; therefore these features will not work.

OHW does support these features. However, the process is different than in OHJ. An Oracle Help for the Web help system can be viewed in any current web browser, and those browsers do not directly support these conventions. Therefore, these custom features are processed in the OHW servlet on the web server and are streamed to users' browsers as standard HTML links and as browser-specific JavaScript.

7.2 Topic ID Links

The target of an HTML `<a href="target">` link can be specified using either a URL (as with standard HTML links), anchor links, or by using the Oracle Help `topicid` protocol, along with a topic ID specified in the helpset's map file. For example:

```
<a href="topicid:getting_started">Getting Started</a>
```

When the *`Getting Started`* link is clicked, Oracle Help references the map file and jumps to the HTML file associated with the link's topic ID.

The `topicid` protocol also supports anchor links. For example:

```
<a href="topicid:getting_started#advanced">Getting Started</a>
```

When the *`Getting Started`* link is clicked, Oracle Help references the map file and jumps to the `advanced` anchor position in HTML file associated with the link's topic ID.

7.3 Associative Links

An associative link is a link that is associated with multiple targets. When the user selects an associative link in a topic, a list of all topics associated with the link is displayed, and the user can choose a topic from the list.

Oracle Help supports associative links through the Oracle Help `alink` protocol, along with the link file that specifies the associative links for the helpset. For example, an associative link that displays all topics associated with the "worksheet" keyword is specified as follows:

```
<a href="alink:worksheet">Related Topics</a>
```

Oracle Help uses the associative link keyword to search the link file (or files) and display a pop-up window with a list of related topics. The feature is particularly useful when link files from multiple helpsets are merged.

For example, select *`this link`* to display a list of associative links defined as follows:

```
<a href="alink:alinkexamples">this link</a>
```

For more information about link file, see Section 6.6, "Link File".

7.4 Custom Protocol Links

Oracle Help for Java support links for custom protocols through the Oracle Help `custom` protocol. For example, a link that uses a custom protocol named `myProtocol` is specified as follows:

```
<a href="custom:myProtocol:myValue">Link to activate custom protocol</a>
```

Defining custom protocols is a powerful way for your help system to call back into your application. You can handle such links in your application by registering a `CustomProtocolHandler` with your Help instance. Create an implementation of `oracle.help.CustomProtocolHandler` and register it with your `oracle.help.Help` instance through the `registerCustomProtocolHandler` method. For the example link, you would register an instance of your `CustomProtocolHandler` using the string `myProtocol` as the first argument to the `registerCustomProtocolHandler` method.

When Oracle Help for Java encounters custom protocol links, it searches for a `CustomProtocolHandler` registered with the `Help` object using the identifier

myProtocol. If one is found, the handleValue(String value) method of the CustomProtocolHandler is invoked, passing myValue as the value.

Note: If you want to provide link to an external URL, you are not required to register a custom protocol handler. OHJ provides a registered custom protocol external for the same.

For example, the following code creates a link that navigates to OTN page in the default internet browser instead of OHJ browser.

```
<a
href="custom:external:http://www.oracle.com/technetwork/inde
x.html">Oracle Technology Network</a>
```

7.5 Popups

Popups are supported through the Oracle Help popup protocol. For example:

```
<a href="popup:sheetdefinition">Sheet Definition</a>
```

The keyword that follows the popup protocol is a topic ID, as specified in the helpset's map file. When the pop-up link is clicked, the contents of the file associated with the topic ID is displayed in a lightweight pop-up window.

For example:

```
<a href="popup:ohff_popupdemo_html">this link</a>
```

7.6 Topic IDs

Oracle Help topic IDs are maintained in the map file, and when Oracle Help must reference a topic ID, it uses the data from the map file. However, you can specify topic IDs in the topics file themselves and then use the Helpset Authoring Wizard to generate a map file from that information. To define a topic ID in a topic file, insert a META tag with the following syntax:

```
<META name="topic-id" content="topic_id_name">
```

where topic_id_name specifies the topic ID to be used in the map file.

Note: Third-party authoring tools may use the META tag for generating map files

---

For more information about Helpset Authoring Wizard, see Chapter 10, "Helpset Authoring Wizard".

7.7 Window Types

The helpset file can contain a WinType section where you can define one or more named windows with characteristics such as size, position, and background color. You can associate topics (and topic IDs) with these window types in the map file so that whenever the topic is displayed, it is displayed in the specified window.

Note: Window Types are available for OHJ only.

If you plan to use the Helpset Authoring Wizard, you can associate a window type with a topic in the topic file itself. and must also specify a topic ID in the `topic-id` META tag for the topic. Then the wizard uses the information from both META tags to generate the map file.

To associate a window with a topic in a topic file, insert a META tag with the following syntax:

```
<META name="window-type" content="window_name">
```

where `window_name` is the name of a window defined in the helpset file.

Note:

- You do not have to use this method for associating topics with window types. It may be easier to do it directly in the map file.
- Third-party authoring tools may use this META tag for associating topics with window types.
- In older versions of OHJ, the OHJ display engine read and used this META tag directly. This is no longer the case: the map file is now the central repository for this information.

7.8 Dynamic Mapping of Topic IDs to Files

If your helpset uses a simple convention to map between topic IDs and map files, you may be able to significantly enhance Oracle Help's memory usage and startup time with dynamic mapping.

Oracle Help supports an engine attribute on the `<mapref>` subelement of the helpset's `<maps>` area. By setting the engine attribute, one can use a custom engine to parse the map file and create an object used to map between topic IDs and files. In fact, by using certain engines, you may actually eliminate the map file altogether.The engine attribute is optional, so if it goes unspecified, Oracle Help expects the location attribute to be set on the `<mapref>`, and the map file is parsed and stored in the same manner as it was in older versions of Oracle Help.However, Oracle Help supports two engines that support certain common conventions for mediating between topic IDs and files:

- `oracle.help.engine.XMLMapFixedConventionEngine`
- `oracle.help.engine.XMLMapConventionEngine`

If those two engines do not satisfy your needs for dynamic mapping, you can write a custom implementation of `oracle.help.engine.DataEngine`.

7.8.1 The oracle.help.engine.XMLMapFixedConventionEngine Help Engine

In many cases, for a filename of `myfile.html`, the corresponding topic ID is just `myfile_html`. If your map file is long and redundant list of obvious topic mappings of this form, you must set the engine attribute on `<mapref>` to `oracle.help.engine.XMLMapFixedConventionEngine`.

While using Oracle Help, setting the engine to this value makes your old map file expendable. However, if your help content may be viewed using an older version of Oracle Help, you should keep your old map file around so that the older versions of Oracle Help can fall back to the standard mechanism of topic mapping.

If you are concerned about the help system's memory usage and startup time, it is strongly recommended that you use this new engine. Doing so implies that your map

file is never read, and therefore its contents are not stored in the memory. However, there is one caveat to the engine's use:

All help content (HTML files) must reside in the same directory as the helpset file. In addition, any subhelpsets must also reside in the same directory as the master helpset file. Subdirectories for subhelpsets are not permitted because the help system is not able to find your content unless it is in the same directory as the master helpset. However, different helpsets may reside in different directories.:

In the following example, the map IDs `topic_1` and `topic_2` are not associated with window types and therefore use the helpset's default window type. The map IDs `topic_3` and `topic_4` map to topic files that are displayed in the window defined by the intro window type. Map ID `topic_5.tsk` displays `File_5.html` in the window defined by the task window type. Map ID `topic_5.cncpt` displays the same topic file (`File_5.html`) in a different window type (`concept`). Note also that the association between `URL` and `wintype` is used when linking from topic to topic using URLs instead of topic IDs. For example, if a topic had a hard-coded target to `File_5.html`, clicking the link would display the HTML content in a task window type.

```
<?xml version='1.0' ?>
<map version="1.0">
    <mapID target="topic_1" url="file_1.html" />
    <mapID target="topic_2" url="file_2.html#a1" />
    <mapID target="topic_3" url="file_3.html" wintype="intro" />
    <mapID target="topic_4" url="file_4.html#a2" wintype="intro" />
    <mapID target="topic_5.tsk" url="file_5.html" wintype="task" />
    <mapID target="topic_5.cncpt" url="file_5.html" wintype="concept" />
</map>
```

This scheme allows authors to assign window types to HTML files and to also override those associations by linking to an alternate topic ID. For example, for topic-to-topic links, TOC links, index links, and hard-coded links to `File_5.html`, the author might use `topic_5.tsk`, but for links from a tutorial, the author might use `topic_5.cncpt`. By keeping this information in the map file, the author has one central repository for managing these assignments.

7.8.2 The oracle.help.engine.XMLMapConventionEngine Help Engine

If on your `<mapref>` element you set engine to be `oracle.help.engine.XMLMapConventionEngine` you may define your own topic name convention in your map file. For example, consider the following `<maps>` definition in a helpset:

```
<maps>
  <mapref location="map.xml" engine="oracle.help.engine.XMLMapConventionEngine"/>
</maps>
```

The `XMLMapConventionEngine` supports the standard mechanisms for setting up topic ID and window type mappings. However, it also supports the new `<topicNameConvention>` element.

If using the `XMLMapConventionEngine`, your `map.xml` may resemble the following:

```
<map version="1.1">
  <topicNameConvention urlbase="http://www.example.org/help">
    <text>beginningText</text>
    <filename/>
    <text>_</text>
    <extension/>
    <text>endingText</text>
```

```
  </topicNameConvention>
</map>
```

The idea of the <topicNameConvention> is simple.You simply specify how your topic IDs are structured. If you set the urlBase attribute on the <topicNameConvention>, all help content files are assumed to be located at that URL. If all of your topic IDs begin with a string that is not a part of the filename or extension, you can specify a value for <text> at the beginning of the convention. Then you must specify either the <filename/> or the <extension/> to indicate whether the filename or extension appears first in your topic name convention. Then you can specify the <text> that separates the filename and extension. Either the <filename/> or the <extension/> should follow to indicate whether the filename or extension appears second in the convention. A final <text> may be specified if all topic IDs end with some text that is not part of the filename or extension.

According to the above topic name convention, the topic ID of beginningTextmyfile_htmlendingText would resolve to the file http://www.example.org/help/myfile.html. If the urlBase attribute was unspecified, it would be assumed that myfile.html is in the same directory as the helpset file.

If you want to set up some standard topic mappings and window types in your map file but still use the topic name convention provided by the XMLMapFixedConventionEngine, you could define a topicNameConvention in your map file as follows:

```
<map version="1.1">
  <topicNameConvention>
    <filename/>
    <text>_</text>
    <extension/>
  </topicNameConvention>
  <mapID .../>
</map>
```

In the above convention and the XMLMapFixedConventionEngine, the text that separates the filename and extension can appear multiple times in the topic ID. For example, consider the topic my_file_html. The engines assume that the separator between filename and extension is actually the last appearance of the "_" character in the topic ID. Therefore, the topic resolves to my_file.html.

7.8.3 Optimizing Dynamic Maps

Dynamic mapping of topic IDs to files can result in great improvements in your help system performance. However, context sensitive help calls to specific topics may take a long time to resolve if your library includes many helpsets.

The fundamental reason for this is that the convention-based mapping engines return URLs for topic IDs even if the URLs do not resolve to anything. Because of this, context sensitive help calls go through each helpset in the library and check whether the URLs generated by the engines actually resolve.

In the worst case, for a single context sensitive help call, the help system attempts to connect to as many URLs as there are helpsets in your library. However, Oracle Help provides a simple remedy to alleviate the problem. If you set an engine on your <mapref> element, you may also set the engineParams attribute.

If you use the XMLMapConventionEngine or the XMLMapFixedConventionEngine, you may want to set engineParams to be a space-separated list of prefixes for the topics in

your helpset. For example, if all topics in your helpset begin with either `oh` or `help`, your `mapref` would look like the following:

```
<mapref engine="XMLMap..." engineParams="oh help">
```

Setting `engineParams` for either of the convention-based engines ensures that the helpset only tries to resolve topics if they start with a valid prefix, preventing an attempted connection to an URL. Failure to set `engineParams` does not break your help system, but performance will not be optimal.

Part III

Authoring Oracle Help

This part contains information on authoring Oracle Help systems. It contains the following chapters:

- Chapter 9, "Authoring Oracle Help Systems"

  This chapter provides an introduction to Oracle Help authoring.

- Chapter 10, "Helpset Authoring Wizard"

  This chapter describes how to use the Oracle Help for Java authoring wizard to convert help systems created using other formats.

- Chapter 11, "Using the Text Search Indexer"

  This chapter describes how to use the text search indexer.

8

Oracle Help for the Web Configuration File

This chapter describes the Oracle Help for the Web configuration file and the various elements (such as `helpConfiguration`, `brandings`, `locales`, `parameters`, and `navigatorAliases`) of the file.

This chapter includes the following sections:

8.1 About Oracle Help for the Web Configuration File

The Oracle Help for the Web configuration file is an XML file that defines a OHW configuration. This configuration controls all adjustable features of the OHW servlet. A typical name for this file is `ohwconfig.xml`, but it can have any name, if that name is specified as the value of the `configFileName` initialization parameter for the servlet. The OHW demonstration files uses `ohwconfig.xml`.

8.2 The <helpConfiguration> Element

The `<helpConfiguration>` element is the top-level element in the help configuration file. All of the elements of the configuration, described below, should appear between the `<helpConfiguration>` tag and the `</helpConfiguration>` tag. The `helpConfiguration` element has two attributes, `version` and `debugMode`.

You must always set the `version` attribute of `helpConfiguration` element. For Oracle Help for the Web, the value of the attribute should be `2.0`.

For example:

```
<helpConfiguration version="2.0">
```

8.2.1 The debugMode Attribute

Enable debug mode by setting `debugMode="true"` on the `<helpConfiguration>` element in `ohwconfig.xml`. This allows helpsets to be loaded in debug mode, where malformed helpsets are skipped over. The debug text is displayed along with the branding information in the upper left corner of the screen and in the title bar of the browser. The debug text indicates how many malformed helpsets have been skipped over while running the application.

If there are no malformed helpsets, following message is displayed in the branding area, also shown in Figure 8–1:

```
Debug Mode: 0 Missing Helpsets".
```

Figure 8–1 Helpset Information in Debug Mode

There could be many reasons for a malformed helpset. The following are some common reasons:

- `id` attribute is missing
- `id` attribute is a duplicate id
- `location` attribute is missing
- `location` points to an invalid location

When you deploy the help system, a WARNING message is logged to indicate that debug mode is being used. Whenever a helpset is skipped over, a SEVERE message is logged (instead of throwing an exception), and the log message displays the `id` and `location` attributes of the helpset that was skipped over. By default, the log messages are logged through `System.err` stream and are displayed in the developer environment's console window. If you're using JDeveloper, the messages are logged in the Log window and a log file is created at `<JDEV_HOME>\jdeveloper\system`*`version`*`\DefaultDomain\servers\DefaultServer\logs\DefaultDomain.log`.

For example:

Let's assume you enabled the `debugMode` as follows:

```
<helpConfiguration version="2.0"
  xmlns="http://xmlns.oracle.com/help/web/config" debugMode="true" >
```

Then, if you changed the location of your `ohguide` helpset to `ohguide_new` directory, but forgot to update the corresponding path in `ohwconfig.xml`:

```
<books>
    <helpSet id="ohguide" location="ohguide/ohguide.hs"/>
    <helpSet id="shake" location="shakespeare/shakespeare.hs"/>
</books>
```

The following message is displayed in the branding area, also shown in Figure 8–2:

```
Debug Mode: 1 Missing Helpsets
```

Figure 8–2 Missing Helpset Information in Debug Mode

8.3 The <brandings> Element

The `<brandings>` element specifies the product branding text or image that appears above the tab bar. The `<brandings>` does not have any attributes, and is a placeholder for all brandings information.

The `<brandings>` element can contain only one element described in the following table. If no branding information is specified, the default branding text is used.

Table 8–1 <brandings> Child Elements

Element	Description
`<branding>`	Renders branding information from attribute information and can be specified to work with certain locales. This element supports the following attributes: ■ `text` – The text to be rendered. Note that you can provide simple HTML attributes such as italics by using the special character named entities (escape characters). For example, to preserve the italics in the name My*Italics*, the branding text line would be: `<branding text="My<i>Italics</i>" />` ■ `imageHeight` – The image height for the branding image. If not specified, the image is displayed with default height of 25 pixels. ■ `imageSource` – The image source for a branding image. This takes precedence over the text if both are set. The declaration of an image location is not always the same as the translated path: ■ For a servlet path like `/foo/myImage.jpg`, the translated path is `/<context-root>/foo/myImage.jpg` ■ For an absolute path like `http://mydomain.com/myImage.jpg` the translated path is `http://mydomain.com/myImage.jpg` ■ For an off the server root path like `//foo/myImage.jpg` the translated path is `/foo/myImage.jpg` ■ `imageShortDesc` – A short description for the image. This text is displayed in the same way an ALT description is handled in HTML. For example, it appears when the mouse rolls over the image. ■ `locales` – A space delimited list of locales that this branding should be supplied for. If this is missing, the branding information is applied across all registered locales. OHW narrows the locale used, but does not expand it. For instance, if you specified a locale `nl` and the browser was set to use locale `nl_NL`, OHW first tries and fails to find `nl_NL`, then tries and succeeds in finding `nl`. Thus, if you specify a more specific locale in OHW (`nl_NL`) and a less specific locale in the browser (`nl`), the specification is ignored, in alignment with other standard locale mechanisms such as resource bundle.

Table 8–1 (Cont.) <brandings> Child Elements

Element	Description
<brandingFromResource>	Renders branding information from a ResourceBundle. This element supports the following attributes: ■ resource - Java classname of ResourceBundle to use. ■ textKey - The key into the ResourceBundle for the branding text. ■ imageSourceKey - The key into the ResourceBundle for the branding image source. This takes precedence over the textKey is both are present. ■ imageShortDescKey - The key into the ResourceBundle for the branding image description. This text is displayed in the same way an alt description is handled in HTML. For example, it appears when the mouse rolls over the image. ■ locales - A space-delimited list of locales that this branding should be supplied for. If this is missing, the branding information is applied across all registered locales. OHW narrows the locale used, but does not expand it. For instance, if you specified a locale nl and the browser was set to use locale nl_NL, OHW first tries and fails to find nl_NL, then try and succeed in finding nl. As a consequence of this, if you specify a more specific locale in OHW (nl_NL) and a less specific locale in the browser (nl), the specification is ignored, in alignment with other standard locale mechanisms such as the resource bundle.

Note: If you are using XLIFF resource bundles, add the resource bundle support (resourcebundle.jar) library in the classpath.

The resourcebundle.jar file is available in <MW_HOME>\oracle_common\modules\oracle.javatools_11.1.1\ directory, where MW_HOME is the middleware home.

For example:

```
<brandings>
  <branding text="Oracle Help" />
</branding>
```

or

```
<brandings>
  <branding text="Help" locales="en en_US" />
  <branding text="Ayuda" locales="es" />
</brandings>
```

or

```
<brandings>
  <brandingFromResource resource="myApp.resource.MyBundle" textKey="title" />
</brandings>
```

8.3.1 Best Practice for Internationalization

When implementing internationalization, you can avoid having to translate ohwconfig.xml by using these best practices for helpsets that require translation:

1. Use <brandingFromResource> to reference a bundle that is translated for all locales, instead of putting the branding information directly into ohwconfig.xml. Because the branding information comes from a standard Java resource bundle, the bundle can be translated as part of the regular translation process.
2. Put an XML declaration at the top of all help control files: map, TOC, index, and helpset files. If you do this, you must not use <controlFileEncoding>. Use the same encoding for all control files in a given localized helpset. The encoding specified in the helpset file is used to read in all of the control files.

   Note: If you are using an old version of OHJ, the helpset may not recognize an XML declaration and so may require <controlFileEncoding> to be set.

3. You need not separate out each different locale into a different file, because the file has nothing that must be translated. Some sites may want to maintain the separation for other reasons.

8.4 The <locales> Element

You can specify a single locale or multiple locales in the <locales> section of the ohwconfig.xml file. The locales section has one or more tags specifying a single locale in the system. These tags are either of type <locale> element or <localeFromFile> element. As the names suggest, the <locale> element specifies the locale inline, whereas the <localeFromFile> element delegates the declaration of the locale to an external file. The external file, in turn, contains a single <locale> element.

Table 8–2 <locales> Child Element

Element	Description
<localeFromFile>	The <localeFromFile> element contains a single attribute and no children. The <locale> element within the external file effectively replaces the <localeFromFile> element functionally. This element supports the following attribute: ▪ source - The source of the file containing the <locale> element for this locale. This filename is relative to the configuration file.

Table 8–2 (Cont.) <locales> Child Element

Element	Description
<locale>	Specifies the ISO language, country, and (optionally) variant codes that is used to construct a Java Locale for locale-sensitive operations. Also specifies the Java-supported encoding name for the character set encoding of the Oracle Help control files (for example, ISO8859_1). The first <locale> element listed is the default locale. The default locale is used if there is no content for the requested locale. If a content for a particular language is defined, the content is used to serve all locales for that language. For example, if a <locale> tag with language="en" defines English books, the same contents is used to serve American English (en-US), British English (en-GB) and all other English speaking users(en-*). This element supports the following attributes: ■ language - A lowercase two-letter language code as defined by ISO-639 (for example, en for English). ■ country - An uppercase two-letter country code as defined by ISO-3166 (for example, US for United States). Specify this value if it is required to distinguish the language from another variant of the language. For example, simplified Chinese (zh-CN) and traditional Chinese (zh-TW). ■ variant - Optional variant code for browser or platform specific locales. ■ group - Optional group that the locale is associated with. Group selection occurs on the URL with the special group parameter. The <locale> section supports children of type <book>.

For example:

```
<locales>
  <locale language="en">
    ... set of books for this locale ...
  </locale>
</locales>
```

8.4.1 The <locale> Child Element <books>

The <books> element specifies the content to be displayed in Oracle Help for the Web. The <books> element can contain any number of helpsets. Helpsets are also called as books.

The <books> element can contain the following elements:

Table 8–3 <books> Child Elements

Element	Description
<helpSet>	A helpset to include in this instance of OHW. This element has the following attributes: ■ id – Unique id to this book. This attribute is required. ■ jar – The location of the JAR (Java Archive) file, if the helpset is in a JAR file. If this attribute is used, the location attribute must also be used to specify the location of the helpset file inside the JAR file. ■ location – The location of the helpset file. If the helpset is not JARred, this is name of the helpset file, with any appropriate path information. The path can be either absolute or relative to the location of the configuration file. If this attribute is used with the jar attribute, the location is the location of the helpset file in the JAR file. ■ controlFileencoding – A Java-supported encoding name. The set of supported encodings vary with JDK version. For a list of supported encodings for Java SE, see http://download.oracle.com/javase/1.5.0/docs/guide/intl/encoding.doc.html.

For example:

```
<books>
 <helpSet id="shakre" location="shakespeare/shakespeare.hs" />
 <helpSet id="myproduct" location="myProduct/myProductHelp.hs" />
 <helpSet id="myhelpset" jar="myJar.jar" location="myHelpset.hs" />
</books>
```

The <helpSet> elements can contain zero or more <contentLocation> elements for situation when help topic files are located in locations other than those expected by Oracle Help.

8.4.2 The <contentLocation> Element

By default OHW automatically processes help topic files that are located in the same locations as helpset (.hs) files.

In helpsets, OHW processes help topic HTML files:

- in the same directory as the .hs file and subdirectories under that location
- or if the helpset is in a JAR file, all help topic files in the same JAR

If you have help topic files in some other location, you must use the <contentLocation> element to point to that location.

The <contentLocation> element has the attribute baseURI. It represents a URI to the root location of a set of help content, using a path that is either absolute or relative.

The <contentLocation> element can be a child of <helpSet> (which are themselves children of <books>). A <helpSet> can contain zero or more <contentLocation> elements.

This element is needed because, unlike plain HTML files, Oracle Help help topic files must be processed by the servlet to be displayed. Therefore, it is necessary to explicitly list the locations where help topic files referenced in the helpset reside, if they are not in the default locations. This can happen if your helpset includes a subhelpset in another location or even on another web server or if your context sensitive map file

contains references to help topic files located elsewhere on the same server or on a different server.

Absolute Path Content Locations

If you know the absolute path of content location, you can specify the absolute path.

For example, if OHW is configured with a local helpset, `myHelpSet.hs`, that references a subhelpset on another server, `http://www.myCompany.com/help/remoteHelpSet.hs`, the configuration file should contain:

```
<books>
  <helpSet id="myhelpset" location="myHelpSet.hs">
    <contentLocation baseURI="http://www.myCompany.com/help/" />
  </helpSet>
</books>
```

This configuration informs OHW that the local `myHelpSet.hs` file references help content on that server. For example, `http://www.myCompany.com/help/remoteHelpSet.hs`.

OHW thus processes help topic files both in the same location as `myHelpSet.hs` and in the remote location.

Relative Path Content Locations

If you do not know the absolute path of the content location, you can specify a relative content location. These locations are relative to the configuration file, not the helpset file, and must be terminated with a trailing slash.

For example, to specify a content location that is under `images` directory, which is located in the same directory as the configuration file, you could declare the following:

```
<contentLocation baseURI="images/" />
```

To specify a content location pointing to an images directory that is located one directory below the configuration file, you can declare the following:

```
<contentLocation baseURI="../images/" />
```

The location is relative to the helpset. If the helpset is inside a JAR file, the JAR file is evaluated as a directory when resolving the relative path. For instance, assume these directory paths:

```
/myhelpsets/myhs.jar
/myhelpsets/myhs.jar!myhelpset.xml
/common_images
```

The `<baseURI>` value would be:

```
../../common_images
```

8.4.3 Sample <locales> Section

This sample of the section of a configuration file specifies an English and Japanese configuration file.

```
<locales>

  <!-- An English locale -->
  <locale language="en" country="US" controlFileEncoding="UTF-8">
    <books>
```

```
      <helpSet id="hs1" location="ohguide/ohguide.hs"/>
      <helpSet id="hs2" location="shakespeare/shakespeare.hs"/>
    </books>
  </locale>

  <!-- A Japanese locale from an external file -->
  <localeFromFile source="jp_config.xml" />

</locales>
```

8.5 Sharing Resources Across Helpsets

If you have many localized helpsets all using the same images, CSS stylesheets, or other resources, OHW can support the sharing of these resources across the helpsets. Typically, shared resources exist in a separate directory, usually below any subdirectories holding the localized helpset information. For instance, the following is a typical directory structure for using shared resources:

```
<main directory>
       ohwconfig.xml
       /en
          - owh_helpset_en.hs
          - English helpset files
       /es
          - ohw_helpset_es.hs
          - Spanish helpset files
       /shared
          /images
             - shared images
          /css
             - shared stylesheets
```

Since OHW by default can only access directories that are underneath the `.hs` file. Thus to make the shared directory available to OHW, you must define a content location for this directory. For best results, use a relative content location for this task.

OHW supports relative paths when dealing with resources, and appropriately converts these paths to the correct paths at runtime. To make use of shared resources, an HTML file can point to the resource using a relative path. For example, if you have an HTML file in the `./en` directory to point to a shared image resource, the HTML file should contain the following code:

```
<img alt="Image of cat" src="../shared/images/cat.gif />
```

If you have the content location defined, OHW fixes this path so that it works at runtime. You can also use relative paths within the control files for your helpset. For example, you could define the following code in your map file that is located in the `./en` directory:

```
<mapID target="someDescriptor" url="../shared/html_files/theContent.html" />
```

Finally, if your localized helpset is contained in a JAR file, then the JAR file is counted as a directory when specifying the relative path. Thus, if all helpset files are contained in a JAR under the `./en` directory, and you want to point to a shared image resource, the HTML file should contain this code:

```
<img alt="Image of cat" src="../../shared/images/cat.gif />
```

The extra `..` in above code is required because the JAR file counts as a directory.

8.6 The <parameters> Element

The <parameters> element specifies the values of various other OHW parameters. These parameters are all case-insensitive.

Table 8–4 <parameters> Child Elements

Element	Description
<combineBooks>	If true, the views from all books are displayed in one set of navigators (tabs). If false, each book has its own set of navigators displaying just the information for that book.
<useLabelInfo>	If true, the author-defined labels are used for the navigators of merged helpsets. If false, the default labels are used, for example, the tab labels **Contents**, **Index**, and **Search**.
<displaySiteNavigation>	If true, turns on site navigation links across all supported navigators. For browsers that support site navigation, the appropriate links are generated in the meta content of pages served by OHW.

When combineBooks is true, only navigators of the same type and with the same label are merged into the same navigator, for example the **Index** tab. That can lead to unintended results if you have set useLabelInfo to true. For example, if one helpset has overridden the default Index label with Keyword Index and left another with the default, the indexes won't be merged in the same tab. You can change this by setting the labels to be the same (in the helpset file for a helpset) or by setting useLabelInfo to false in the configuration file.

For example:

```
<parameters>
  <combineBooks>true</combineBooks>
  <useLabelInfo>false</useLabelInfo>
</parameters>
```

Other important parameters include:

Table 8–5 <parameters> Keyword Child Elements

Element	Description
<keywordBlockSize>	The number of keywords to show on one page. The default value is 10.
<keywordTopicsBlockSize>	The number of topics to show on one page. The default value is 10.
<searchBlockSize>	The number of search results to show on one page. The default value is 10.

These elements are used to support performance tuning or specify nondefault error, state, or locale handling.

Table 8–6 <parameters> Performance Child Elements

Element	Description
<maxSearchThreads>	The maximum number of threads that OHW can use to perform searches. Default value is 10. If you presume that many users would be connected to the help system and using the search feature at the same time, you can increase the value of maxSearchThreads to more than 10, however it is not recommended. If the help system does not respond during search, verify that the index file (.idx file) is not corrupt before you change the maxSearchThreads value.
<errorPage>	When a topic is not found, OHW displays this topic instead. The value of this parameter is a topic ID. OHW provides a standard error page if no value is set.
<stateManager>	Specifies the state manager OHW should use. If set to cookie, user state is saved for one month. If set to session, the user state is saved through single request. The default is 'session'.
<localeDeterminer>	OHW uses the specified locale determiner to select a localized helpset based on a user request. OHW provides a default locale determiner that uses browser settings to determine the locale if no value is set.
<cacheSize>	The number of active localized helpsets to be kept in memory simultaneously. The default value is 3.

8.7 The <navigatorAliases> Element

The optional <navigatorAliases> element enables you to use classnames in your helpset file that do not correspond to the classnames of the navigators in OHW. Alias registrations are done through the use of the <alias> element. The <alias> elements are contained within a single <navigatorAliases> element, and has the following attributes:

- name - The name to use as the alias.
- value - The value this alias should map to.

For example:

```
<navigatorAliases>
  <alias name="oracle.help.navigator.tocNavigator.TOCNavigator"
value="oracle.help.web.navigator.tocNavigator.TOCTreeNavigator" />
</navigatorAliases>
```

Note: The tree-based TOC Navigator of OHW uses the oracle.help.web.navigator.tocNavigator.TOCTreeNavigator class name.

8.8 Custom Protocol Links

OHW supports links for custom protocols through the Oracle Help custom protocol. For information about custom protocol links in OHJ, see Section 7.4, "Custom Protocol Links."

In order to handle custom protocol links in OHW, clients must register Custom Protocol Converters in the ohwconfig.xml file for each custom protocol used in their help content. The syntax in the ohwconfig.xml file looks like this:

```
<customProtocolRegistry>
```

```
  <customProtocol name="xlink"
class="oracle.help.web.converter.ConfigurableCustomProtocolConverter">
    <parameters>
      <prepend>http://www.myserver.com/index.jsp?someParam=</prepend>
      <targetFrame>_blank</targetFrame>
    </parameters>
  </customProtocol>
</customProtocolRegistry>
```

Users may write their own implementations of `CustomProtocolConverter`. However, OHW includes the `ConfigurableCustomProtocolConverter`, which is configurable using parameters set in `ohwconfig.xml`. The supported parameters are:

```
<prepend>optional string to be prepended to the value</prepend>
<append>optional string to be appended to the value</append>
<targetFrame>optional target frame</targetFrame>
```

In an HTML topic file, authors could use the standard Oracle Help custom protocol syntax. For example:

```
<a href="custom:xlink:someXLINKID">An Example Custom Protocol Link</a>
```

OHW processes all `custom:` links and run them through the Custom Protocol Converter registered for that custom protocol name.

The link in the above example would be replaced with:

```
<a href="http://www.myserver.com/index.jsp?someParam=someXLINKID" target="_blank"
/>
```

In OHJ, it is a popular convention to use `custom:external:` to launch a link in a new browser window. In OHW, the built-in `CustomProtocolConverter` for the external protocol enables the links to work without the users having to explicitly register a converter.

8.9 Preloading Helpsets Containing Embedded Help

OHW supports the `<locale>` tag in the OHW configuration file, which defines a single instance of a locale that OHW supports. The `<locale>` tag specifies the ISO language, country, and (optionally) variant codes that are used to construct a Java Locale for locale-sensitive operations. It also specifies the Java-supported encoding name for the character set encoding of the Oracle Help control files (for example, ISO8859_1). The first `<locale>` element listed is the default locale.

In OHW, the loading of helpsets is handled in a different manner. OHW as an RCF application may have a screen with 50 components, with Definition Text content, specified in a single file within its own helpset. To avoid loading all helpsets and reduce the lag that this would cause when a user simply opens that screen, preloading of a selected helpset is desirable.

So, for OHW, the configuration file supports an optional attribute for the `<locale>` tag, which is called `preload`. The possible values that this attribute can take are `NONE`, `ALL`, and `TOPICMAP`. If not specified, its value defaults to `NONE`.

The behaviors of each value are:

Table 8–7 Preload Value Behavior

Preload Value	Initial Start of OHW	When Topic is Accessed	When UI is Accessed
NONE	No action required.	Load the topic map until the requested topic is found.	Load all views and navigators in the selected helpset.
ALL	Load all helpsets in the locale (which initializes all views and navigators) and also load all topic maps.	Access from cache.	Access from cache.
TOPICMAP	Load only the topic maps for all helpsets in the locale.	Access from cache.	Load all views and navigators in the selected helpset.

For example, the configuration file could look like this:

```
<?xml version='1.0' ?>
<helpConfiguration>
  ...
  <locales>
    <locale language="en" preload="ALL">
      <books>
        <helpset id="helpset1" location=" helpset1.hs"/>
        <helpset id=" helpset2" location=" helpset2.hs"/>
      </books>
      ...
    </locale>
  </locales>
  ...
</helpConfiguration>
```

8.10 Reloading Oracle Help for the Web Configuration File at Runtime

OHW provides a method to reload the Oracle Help for the Web configuration file at runtime to include any changes made in the file while help system is running. An example of such a change is including future patched helpsets without redeploying the help system. When the configuration file reloads, it empties the cache and rebuilds it again.

8.10.1 How to Reload Oracle Help for the Web Configuration File at Runtime

If you want that user can reload the configuration file at runtime, add a new parameter `ohwConfigAutoUpdate` in `web.xml`.

To reload Oracle Help for the Web Configuration File at runtime

1. In `web.xml`, create a new parameter `ohwConfigAutoUpdate` and set it to `true`.

   For example, the `web.xml` file could look like this:

   ```
   <?xml version = '1.0' encoding = 'UTF-8'?>
     ...
     <init-param>
        <param-name>ohwConfigAutoUpdate</param-name>
        <param-value>true</param-value>
     </init-param>
   ```

2. Update the configuration file as desired.
3. Open the default home page of the help application after updating the configuration file.

   The old configuration settings are cleared and new settings take effect after the help application home page is opened.

9

Authoring Oracle Help Systems

This chapter introduces the utilities and processes of authoring Oracle Help systems.

This topic contains the following sections:

9.1 About Authoring Oracle Help Systems

Oracle Help does not include a complete authoring environment. While it is fairly straightforward to author a help system without using an authoring tool, you may prefer to use a tool that supports Oracle Help directly, such as Adobe RoboHelp, Quadralay WebWorks, or AuthorIT Software Corporation's AuthorIT.

Because Oracle Help's file formats are based on JavaHelp file formats, you could also start with any authoring tool that creates JavaHelp systems. However, you have to add features specific to Oracle Help manually. Oracle Help also supports the control file formats from Microsoft's HTML Help, so you could start with a help authoring tool that supports HTML Help. Again, you have to add any Oracle Help-specific features manually.

OHJ and OHW use the same file formats, so you can author a help system one time and display it without modification.

The basic steps for authoring an Oracle Help help system are as follows:

1. Write the HTML pages that comprise the topics of your help system.
2. Create a helpset file. This XML file identifies the help system and specifies many of its properties, including certain aspects of the user interface. For example, this is where you specify which "navigators" (generally displayed as tabs) are displayed. The standard navigators are Contents, Index, and Search. This is the main Oracle Help control file, and it is required.
3. Create a map file. This XML file maps topic IDs to HTML pages. These IDs are used in the API for context-sensitive calls; in table-of-contents, index, and link files; and in topic ID links in topic files. This file is required.
4. Create an index file. This XML file defines the keyword index that appears in the Index tab. If your help system does not have an index, this file is not required. However, if you do not have an index, you should not define an index tab in the helpset file.

5. Create a table of contents file. This XML file defines the table of contents that appears in the Contents tab. If your help system does not have a table of contents, this file is not required. However, if you do not have a table of contents, you should not define a Contents tab in the helpset file.
6. Generate a search index. This file contains the data used when a user searches for a word or phrase in the Search tab. This file is optional. However, if you do not have a text search index, you should not define a Search tab in the helpset file. If you are not using an authoring tool to create your help system, you can use the indexing tool, which is included with Oracle Help. See Using the Text Search Indexer.
7. If you want to use associative links, create a link file. This XML file associates topics with associative link keywords. This file is optional.

9.2 Authoring Utilities Included with Oracle Help for Java

Two authoring utilities are included with Oracle Help for Java. You can use these tools to help author an Oracle Help system if you do not have an authoring tool that supports Oracle Help. Follow the links below for more information on each:

- **Helpset Authoring Wizard**: This wizard converts certain noncompatible help content and control files into files that can be used by Oracle Help. For example, you can use the wizard to create an Oracle Help system from files generated by RoboHelp 2000 for Microsoft HTML Help.
- **Text Search Indexer**: This utility generates the search index file used in Oracle Help when a user searches on a word or phrase in the Search tab.

9.3 Authoring Embedded Help

As part of the embedded Help feature in OHW, each component in an application can have multiple levels of Help associated with it that get triggered on a certain user gesture. The levels of help are:

- Definition Text
- Instructions Text
- Full Help

Each of these levels can be any arbitrary HTML content. Definition and Instructions Text are collectively referred to as *Embedded Help Content*. Full Help content is the content that appears as a separate topic in a Help window.

- Definition Text is to be specified within a `<div>` with the hardcoded style class name `definition`. The `title` attribute, if present, represents the topic ID of the Definition Text. Note that only plain text can be entered here, HTML content is not processed for this `<div>` tag. For example,

  ```
  <div title="topicId" class="definition"> Any plain text </div>
  ```

- Instructions Text is to be specified within a `<div>` with the hardcoded style class name `instructions`. The `title` attribute, if present, represents the topic ID of the Instructions Text. For example,

  ```
  <div title="topicId" class="instructions"> Any HTML content </div>
  ```

Definition and Instructions Text content for a topic are to be specified using typical HTML in a topic file as follows:

9.3.1 HTML Files

The tags for the Definition Text and Instruction Text appear within the HTML body. It is required that the content for the both these levels of Help is specified in the same file. If either of them is not available in the same file, the content is assumed to be undefined.

Help authors specify the Definition Text and Instruction Text content for a topic ID in an existing topic HTML file. Here is a sample topic named topic1.html.

```
<html>
<body>

<div title="topic1" class="definition">
  This is Definition Text help content.
</div>

...

<div title="topic1" class="instructions">
  This is <b>Instructions Text</b> help content.
</div>

...

This is the regular Full Help content.

...

</body>
</html>
```

Note:

- The title attribute represents the topic ID, and is optional. If not specified, the topic ID of the container HTML file is considered. If specified, it must match the topic ID of the HTML file in which the embedded Help is contained.
- The can appear anywhere within the <body>.
- The definition and instructions style classes can be defined in the existing style sheet
- The two style classes can be defined in such a way that they provide the necessary visual cues to distinguish them from the full Help content. For example, Help authors can use a light yellow background with a gray border to indicate Definition Text and a light blue background with a blue border to indicate Instruction Text. This is useful for creating and editing content.
- The two style classes can be defined to be invisible (display:none) when Help authors must view the full Help content alone. This is useful for previewing.
- HTML within a that has either of the style class names is not indexed.

10

Helpset Authoring Wizard

This chapter describes the Helpset Authoring Wizard, its various screens, and how to use it to create help system.

This chapter includes the following sections:

10.1 About Helpset Authoring Wizard

The Helpset Authoring Wizard, also known as OHJ Authoring Wizard, generates certain control files used by Oracle Help. The functionality of the authoring wizard is limited: it is primarily intended to be used to convert help systems created using other formats, not to be a full-featured help authoring tool. For example, you can use the helpset wizard to create an Oracle Help system from files generated by an old release of RoboHelp HTML which did not yet support Oracle Help.

The current version of the wizard can generate a helpset file, a search index file, and a map file. It also converts certain JavaScript pop-up links to the Oracle Help format. However, the wizard does not currently convert RoboHelp HTML associative links to their Oracle Help equivalents (despite the option being present).

10.2 Starting the Wizard

When you install OHJ on Windows, a batch file and an initialization file for starting the wizard are generated, using the path into which you installed OHJ. A shortcut for starting the wizard is also added to the Windows Start menu. Select this shortcut to

start the wizard. Alternatively, you can issue the following command at the command prompt:

```
<OHJ_HOME>\bin\authoringWizard.bat
```

For example, if you have installed OHJ in `D:\ohelp` folder, you can run the wizard with the following command:

```
D:\ohelp\bin\authoringWizard.bat
```

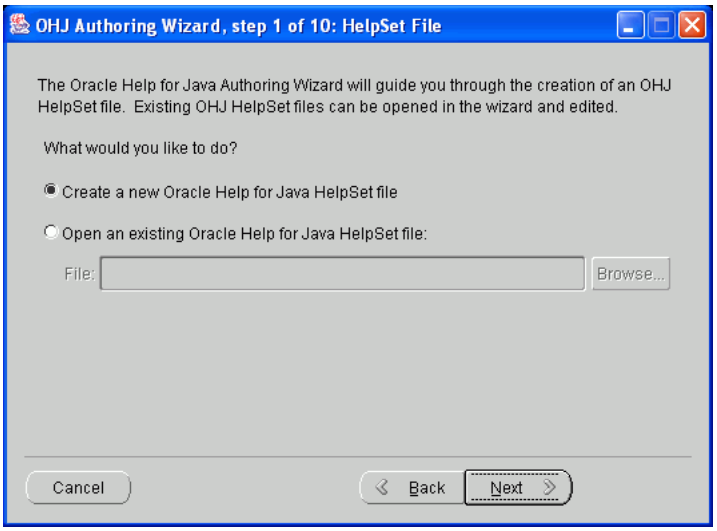
When you run the above command, you are greeted with a Welcome screen of OHJ Authoring Wizard. Click **Next** to continue.

Figure 10–1 OHJ Authoring Wizard: Welcome

10.3 Creating a Helpset File

The first step of the OHJ Authoring Wizard is to create a helpset (`.hs`) file. You may also open an existing helpset file, and edit it later. For more information about helpsets, see Section 5.2, "Helpset File".

Figure 10–2 OHJ Authoring Wizard: HelpSet File



Use the following information to enter data in each field of the HelpSet File page:

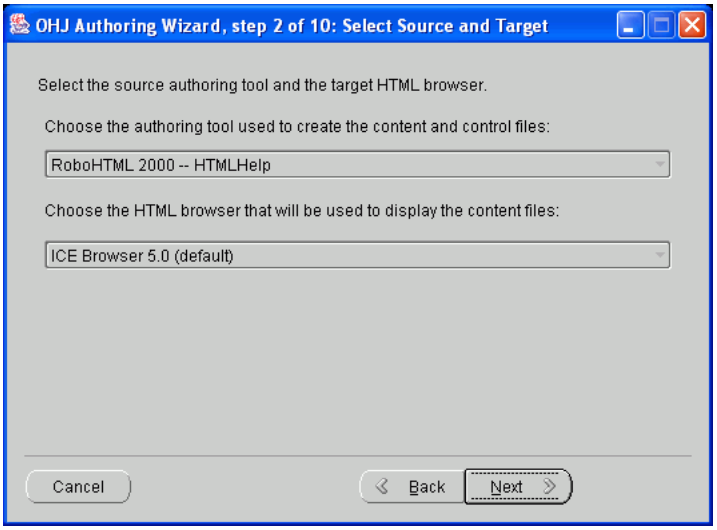
Field	Description
Create a new Oracle Help for Java HelpSet File	Select the option to create a helpset file.
Open an existing Oracle Help for Java HelpSet File	Select the option if you have a helpset file ready and want to edit it.
File	Click **Browse** and select the helpset file you want to edit.

When you are done, click **Next** to continue.

10.4 Specifying Authoring Tool and HTML Browser

The Select Source and Target page of OHJ Authoring Wizard enables you to choose the authoring tool and the browser to display the help files.

Figure 10–3 OHJ Authoring Wizard: Select Source and Target



Use the following information to enter data in each field of the Select Source and Target page:

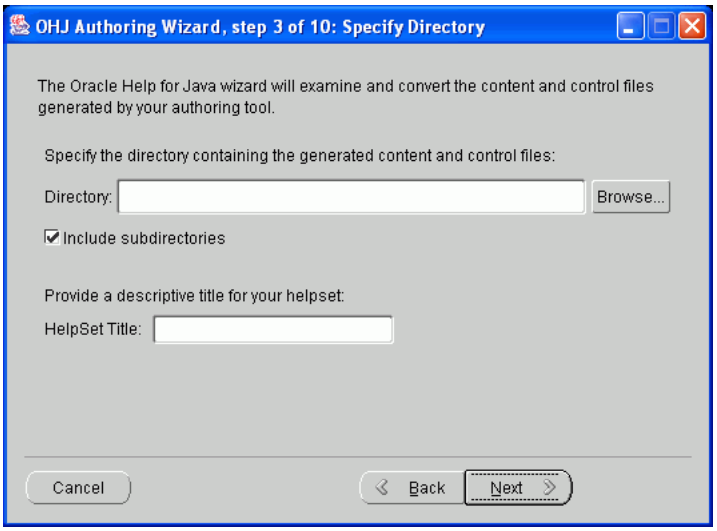
Field	Description
Choose the authoring tool	Select the authoring tool used to create the help files.
Choose the HTML browser	Select the browser you want to use to display the html files.

When you are done, click **Next** to continue.

10.5 Specifying Source Directory

The Specify Directory page of OHJ Authoring Wizard enables you to choose the directory containing the html files and other control files.

Figure 10–4 OHJ Authoring Wizard: Specify Directory



Use the following information to enter data in each field of the Specify Directory page:

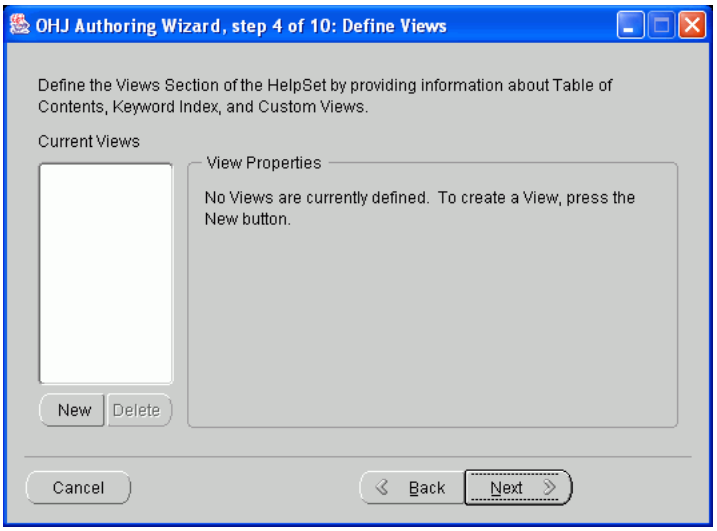
Field	Description
Directory	Click **Browse** and select the source directory of the html files. If you have specified an existing helpset file, the field is not available for editing and displays the directory location of the helpset file.
Include Subdirectories	Select the checkbox to include subdirectories of the selected **Directory**.
HelpSet Title	Specify the title of the helpset file

When you are done, click **Next** to continue.

10.6 Defining Views

The Define Views page of OHJ Authoring Wizard enables you to define views in the OHJ Help Navigator window. You can create views for table of contents, keyword index, and define custom views.

Figure 10–5 OHJ Authoring Wizard: Define Views



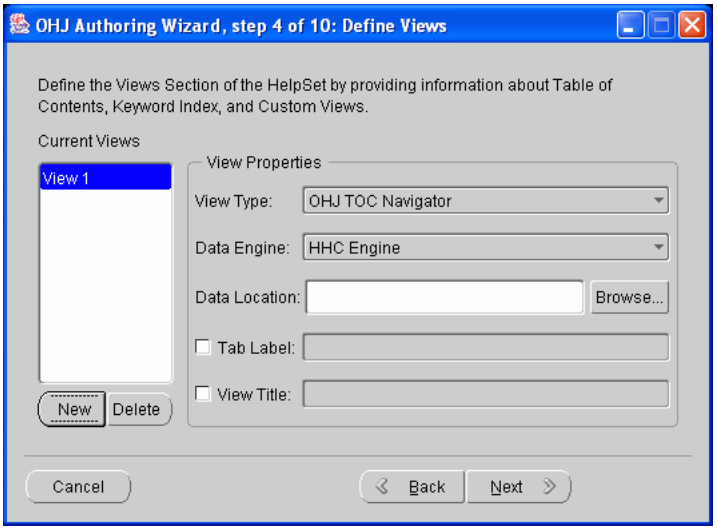
Field	Description
New	Creates a new view and adds the name of the view in **Current Views** list.
Delete	Deletes the selected view from the **Current Views** list.

When you are done, click **Next** to continue.

10.6.1 Defining a New View

OHJ Authoring Wizard enables you to create multiple views in your help. By default, no views are defined. To define a new view, click **New**. Figure 10–6 shows the fields available in the Define Views page of the wizard when a new view is created.

Figure 10–6 OHJ Authoring Wizard: Define New Views



Use the following information to enter data in each field of the Define Views page:

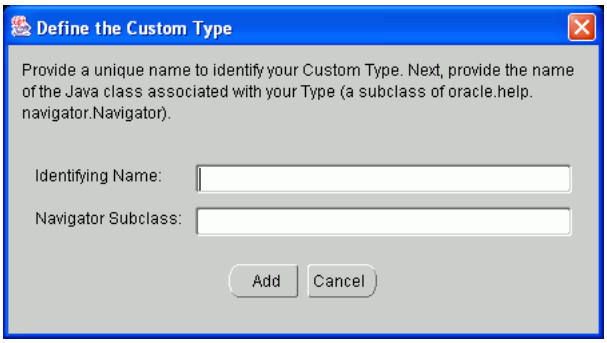
Field	Description
View Type	Select the view type of new view defined. You can choose the view type as **OHJ TOC Navigator**, **OHJ Keyword Navigator**, **OHJ Search Navigator**, or define a custom navigator.
Data Engine	Select the data engine for the new view. You can choose the engine as **HHC Engine**, **HHK Engine**, **XML TOC Engine**, **XML Index Engine**, **TOC Engine**, **TOK Engine**, **Search Engine**; or define a custom data engine.
Data Location	Click **Browse** and select the location of the control file.
Tab Label	Select the checkbox and enter the label of the tab displayed in OHJ Help Navigator window.
View Title	Select the checkbox and enter the title of the view.

10.6.2 Defining a Custom View Type

You can define a custom view if the available views do not meet your requirements.

To create a custom view type, select **Custom** from the **View Type** dropdown list of Define Views page. The Define the Custom Type dialog appears.

Figure 10–7 OHJ Authoring Wizard: Define Custom View Type



Use the following information to enter data in each field of the Define Views page:

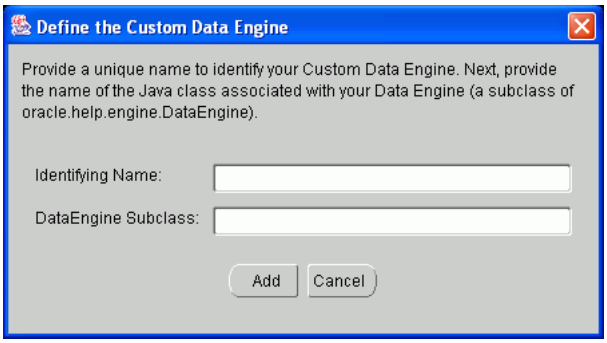
Field	Description
Identifying Name	Specify the unique name of the view.
Navigator Subclass	Specify the name of the Java class. The class must be a subclass of `oracle.help.navigator.Navigator`.

10.6.3 Defining a Custom Data Engine

You can define a custom data engine if the available data engines do not meet your requirements.

To create a custom data engine, select **Custom** from the **Data Engine** dropdown list of Define Views page. The Define the Custom Data Engine dialog appears.

Figure 10–8 OHJ Authoring Wizard: Define Custom Data Engine



Use the following information to enter data in each field of the Define Views page:

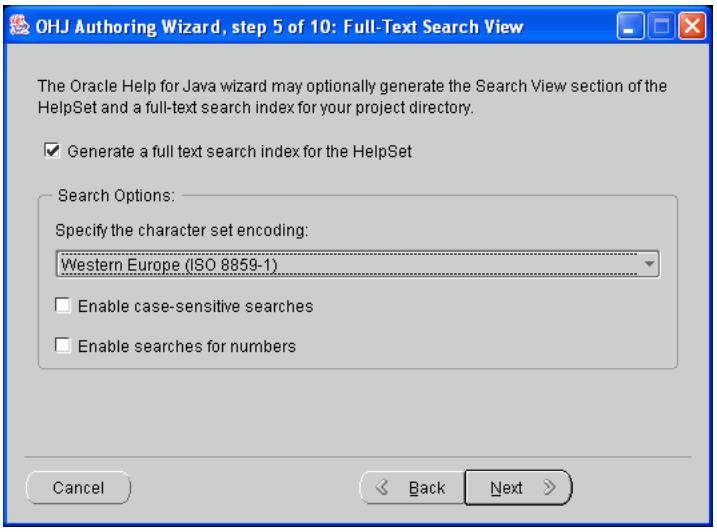
Field	Description
Identifying Name	Specify the unique name of the data engine.
DataEngine Subclass	Specify the name of the Java class. The class must be a subclass of `oracle.help.dataengine.DataEngine`.

10.7 Defining Full-Text Search View

The Full-Text Search View page of OHJ Authoring Wizard enables you to create a Search View and generate a full-text search index for the help.

To create a full-text search index, select the **Generate a full text search index for the HelpSet** checkbox.

Figure 10–9 OHJ Authoring Wizard: Full-Text Search View



Use the following information to enter data in each field of the Full Text Search View page:

Field	Description
Generate a full text search index for the HelpSet	Select the checkbox to create a full-text search index.

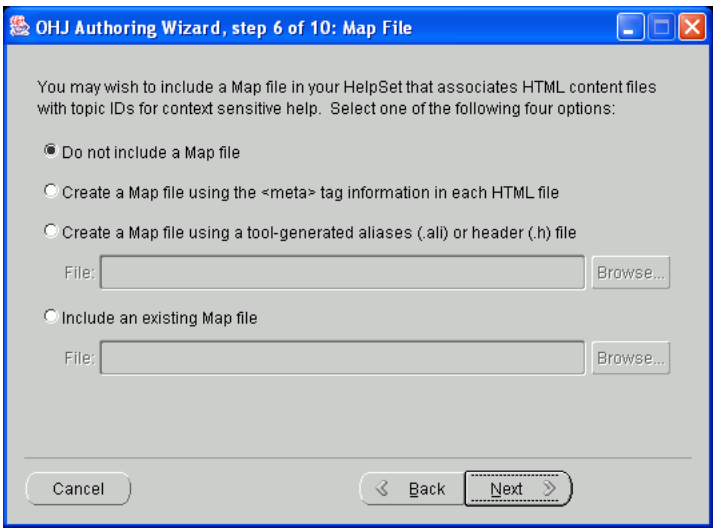
Field	Description
Specify the character set encoding	Select the desired character set encoding from the dropdown list. By default, **Western Europe (ISO 8859-1)** is selected.
Enable case-sensitive searches	Select the checkbox to allow case sensitive search feature in your help.
Enable searches for numbers	Select the checkbox to allow numeric search feature in your help.

When you are done, click **Next** to continue.

10.8 Defining Map File

The Map File page of OHJ Authoring Wizard enables you to create a map file for your help and associate unique topic ids with your help topics.

Figure 10–10 OHJ Authoring Wizard: Full-Text Search View



Use the following information to enter data in each field of the Full Text Search View page:

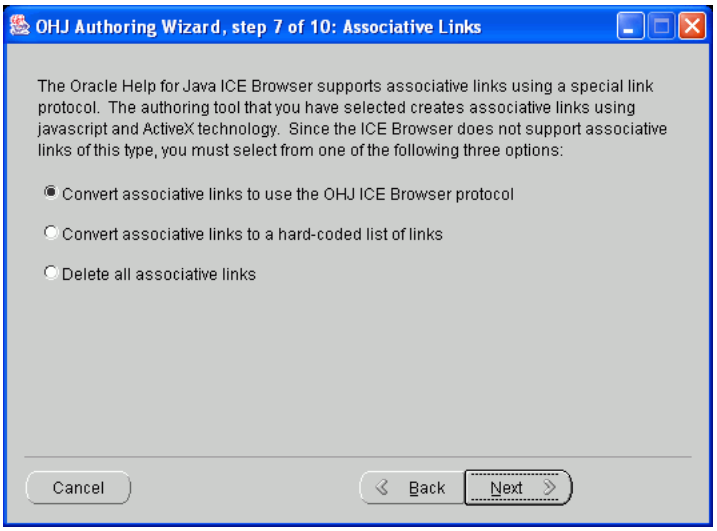
Field	Description
Do not include map file	Specifies that you do not want to provide a map file. This option is helpful if your help is not a context-sensitive help.
Create a Map file using the <meta> tag information in each HTML file	Specifies to create a map file from <meta> tags. Some help authoring tools (such as RoboHelp) insert topic-ids in meta tags in the HTML file. This wizard option finds the meta tags and creates a map file from them
Create a Map file using tool generated aliases (.ali) or header (.h) files	Specifies to create a map file from generated aliases or header files. Click **Browse** to specify the map file name and location.
Include an existing Map file	Specifies to use an existing map file. Click **Browse** to specify the map file name and location.

When you are done, click **Next** to continue.

10.9 Converting Associative Links

The Associative Links page of OHJ Authoring Wizard enables you to convert the associative links defined in the existing help system to it's own associative link protocol.

Figure 10–11 OHJ Authoring Wizard: Associative Links



Use the following information to enter data in each field of the Associative Links page:

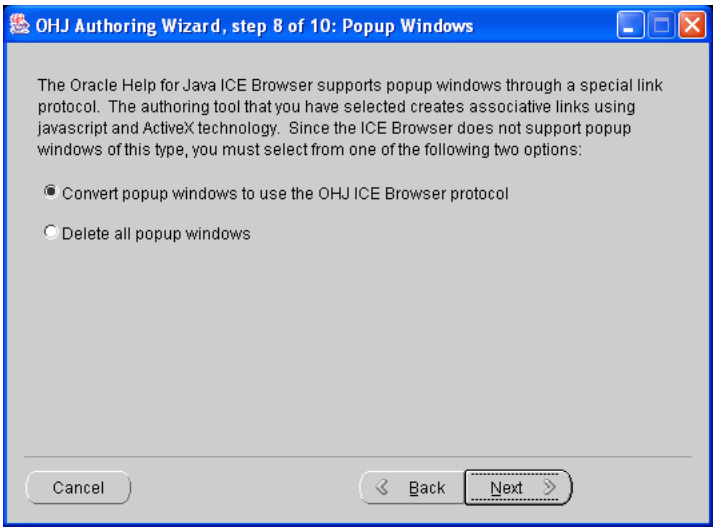
Field	Description
Convert associative links to use OHJ ICEbrowser protocol	Specifies to convert all associative links into ICEbrowser supported `alink` protocol.
Convert associative links to a hard coded list of links	Specifies to convert all associative links into hard coded list of links.
Delete all associative links	Specifies to remove all associative links from the help pages.

When you are done, click **Next** to continue.

10.10 Converting Popup Window Links

The Popup Windows page of OHJ Authoring Wizard enables you to convert the popup window links defined in the existing help system to it's own popup link protocol.

Figure 10–12 OHJ Authoring Wizard: Popup Windows



Use the following information to enter data in each field of the Popup Windows page:

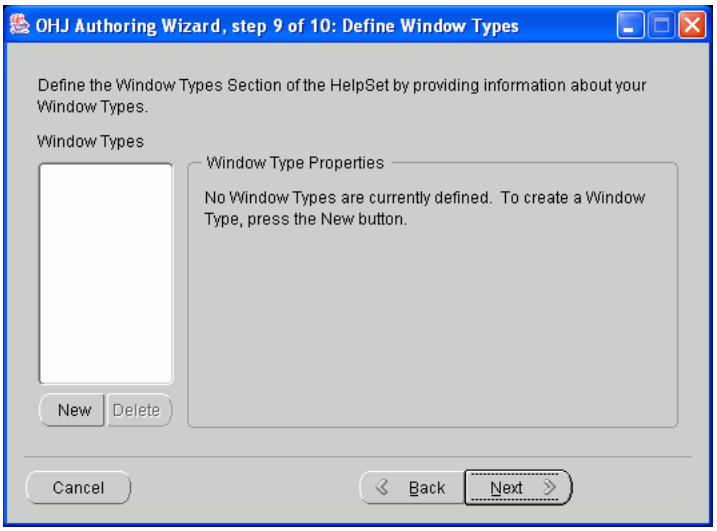
Field	Description
Convert popup windows to use the OHJ ICEbrowser protocol	Specifies to convert all popup window links into ICEbrowser supported protocol.
Delete all popup windows	Specifies to remove all popup window links from the help pages.

When you are done, click **Next** to continue.

10.11 Defining Window Types

The Window Types page of OHJ Authoring Wizard enables you to define window types. By default, no window types are defined. When you define a new window type, you can configure the window's identity, it's placement on the screen, color attributes, and the toolbar buttons available on the window.

Figure 10–13 OHJ Authoring Wizard: Define Window Types



Use the following information to enter data in each field of the Define Window Type page:

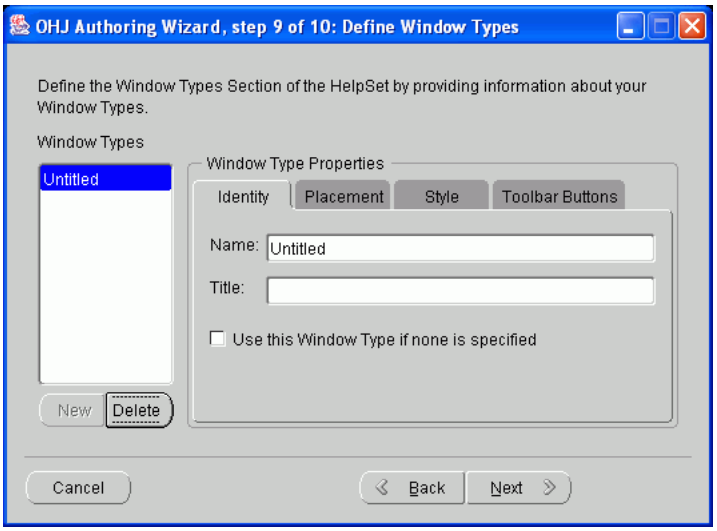
Field	Description
New	Creates a new window type and adds the name of the window in **Window Types** list.
Delete	Deletes the selected window type from the **Window Types** list.

To define a new window type, click **New** and configure its attributes. You can configure following attributes:

10.11.1 Window Identity

The Identity tab of the Define Window Types page enables you to define the window name and title.

Figure 10–14 OHJ Authoring Wizard: Window Type Identity Property



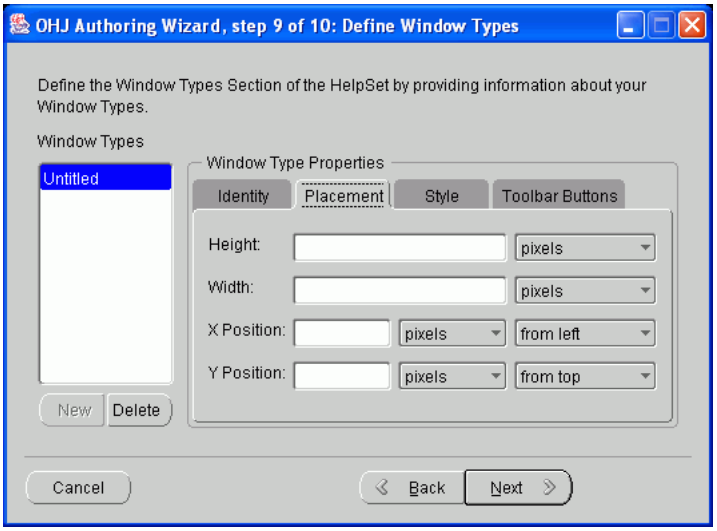
Use the following information to enter data in each field of the Identity tab of the Define Window Type page:

Field	Description
Name	Specifies the unique window name.
Title	Specifies the window title.
Use this Window Type if none is specified	Select the checkbox to make the window as default window.

10.11.2 Placement Attributes

The Placement tab of the Define Window Types page enables you to configure the window size and position on the screen.

Figure 10–15 OHJ Authoring Wizard: Window Type Placement Property



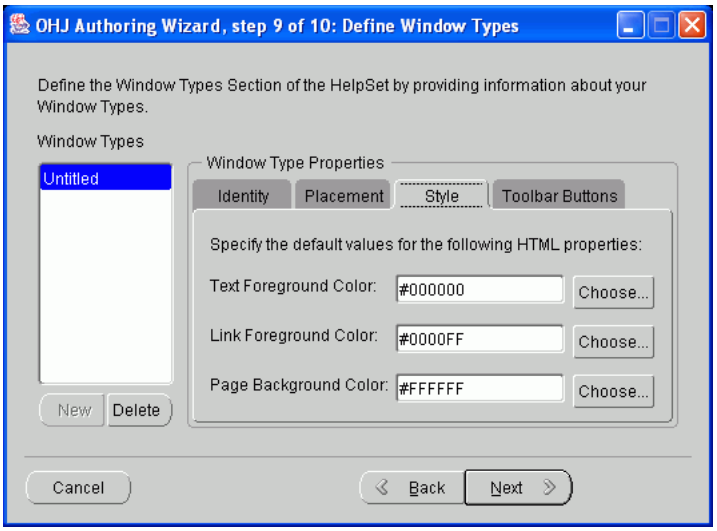
Use the following information to enter data in each field of the Placement tab of the Define Window Type page:

Field	Description
Height	Specifies the windows height, and choose the desired unit as **pixels** or **percent**.
Width	Specifies the windows width, and choose the desired unit as **pixels** or **percent**.
X Position	Specifies the windows X coordinate, choose the desired unit as **pixels** or **percent**, and direction as **from left** or **from right**.
Y Position	Specifies the windows Y coordinate, choose the desired unit as **pixels** or **percent**, and direction as **from top** or **from bottom**.

10.11.3 Style Attributes

The Style tab of the Define Window Types page enables you to configure the window colors, text and background.

Figure 10–16 OHJ Authoring Wizard: Window Type Style Property



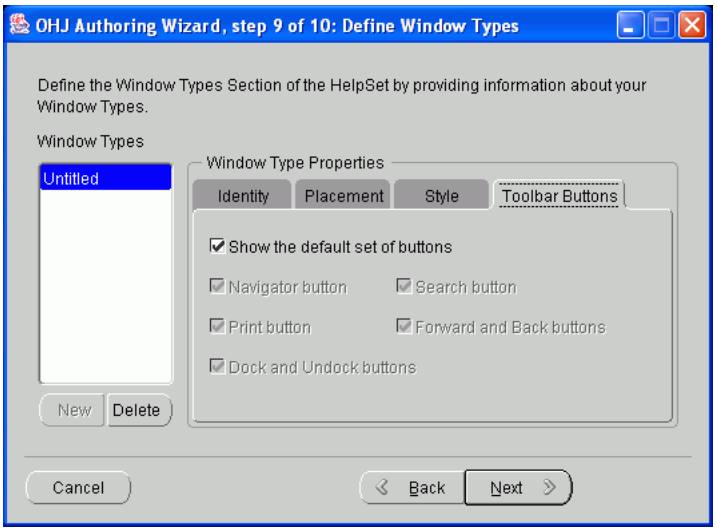
Use the following information to enter data in each field of the Style tab of the Define Window Type page:

Field	Description
Text Foreground Color	Specifies the hexadecimal color code of text in the window. To choose a color, click **Choose** and select your desired text color.
Link Foreground Color	Specifies the hexadecimal color code of link text in the window. To choose a color, click **Choose** and select your desired link text color.
Page Background Color	Specifies the hexadecimal color code of page background in the window. To choose a color, click **Choose** and select your desired page background color.

10.11.4 Toolbar Buttons

The Toolbar Buttons tab of the Define Window Types page enables you to configure the toolbar buttons available in the Help Topic window.

Figure 10–17 OHJ Authoring Wizard: Window Type Toolbar Buttons Property



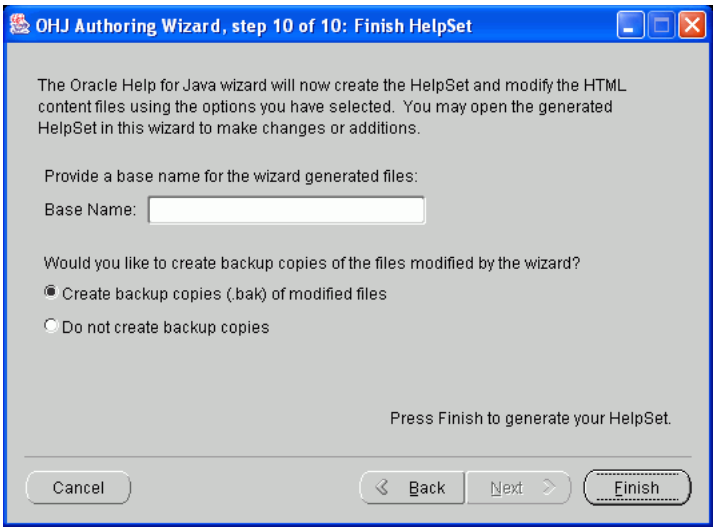
Use the following information to enter data in each field of the Toolbar Buttons tab of the Define Window Type page:

Field	Description
Show the default set of buttons	Select the checkbox to show the default set of toolbar buttons. By default, all toolbar buttons are available. To customize toolbar buttons, clear the checkbox.
Navigator Button	Select the checkbox to add the Navigator toolbar button in the window. The Navigator button enables you to open the Help Navigator window.
Search Button	Select the checkbox to add the Search toolbar button in the window. The Search button enables you to search for a text across the help pages.
Print Button	Select the checkbox to add the Print toolbar button in the window. The Print button enables you to print a topic.
Forward and Back Buttons	Select the checkbox to add the Forward and Backward toolbar buttons in the window. The Forward and Backward buttons allow you to navigate to previously opened topics, and return back to current topic.
Dock and Undock Buttons	Select the checkbox to add the Dock and Undock toolbar buttons in the window. The Dock and Undock buttons allow you to dock and undock the Help Topic and Help Navigator windows.

10.12 Finalizing the HelpSet

The Finish page of the OHJ Authoring Wizard creates the helpset according to options selected in the wizard.

Figure 10–18 OHJ Authoring Wizard: Finish Helpset



Use the following information to enter data in each field of the Finish HelpSet page:

Field	Description
Base Name	Specify the base name of the help set. The base name is without the file name extension, and defines the name of control files. For example, if the base name is `myproject`, the control files are named as `myproject.hhc`, `myproject.hhk`, `myproject.idx`, and so on.
Create backup copies	Specifies to create backup copies of the updated files. The files are saved with `.bak` extension.
Do not create backup copies	Specifies not to create backup copies of the updated files.

When you are done, click **Finish** to generate the helpset file.

11

Using the Text Search Indexer

This chapter describes the text search indexer, and how to use it to generate index files.

This topic contains the following sections:

11.1 About Text Search Indexer

A Java-based text search indexer is included with Oracle Help for Java. The indexer generates the `.idx` files used for text searches within Oracle Help. Two versions of the Text Search Indexer are provided, one for Japanese content and another for non-Japanese content.

11.2 Java Requirements

The Text Search Indexer requires Java5 Standard Edition or later. Performance is greatly enhanced if you leave the Java JIT (Just In Time Compiler) on. Also ensure that you increase the heap size of the Java Virtual Machine to maximum.

11.3 Running the Indexer

Follow these steps to run the indexer:

1. Include the OHJ Indexer JAR file `help`*`indexer-version`*`.jar` in your CLASSPATH.
2. Run the indexer from the command prompt. The indexer supports the following command-line arguments:

   ```
   [-l=locale] [-e=charset] dirnameindexfilename
   ```

   where,

Argument	Description
`-l=locale`	The optional (but recommended) `locale` parameter is specified using the two-letter ISO 639 language codes and ISO 3166 country codes. The format is `language_COUNTRY` or `language_COUNTRY_VARIANT`. If the `locale` is not supplied, the system default locale is used.

Argument	Description
`-e=charset`	The optional (but recommended) charset parameter is the name of the Java-supported character set encoding for the HTML files that are being indexed. If the encoding is not supplied, the default character set encoding of the current system default locale is used. If supplied, the value must be a Java supported character set encoding names; for Java SE, see http://download.oracle.com/javase/1.5.0/docs/guide/intl/encoding.doc.html
`dirname`	The base directory that contains the HTML files you want to index. The indexer indexes all of the files under this directory (and its subdirectories, if any).
`indexfilename`	The name of the index file to be generated.

For example, `java -mx64m oracle.help.tools.index.Indexer -l=en_US -e=8859_1 D:\MyHTMLFiles myIndex.idx`

The above command sets the Locale to language: English, country: Unites States, encoding: `8859_1`, and indexes the `D:\MyHTMLFiles` directory creating the `myIndex.idx` file.

11.4 Running the JapaneseIndexer

Follow these steps to run the indexer:

1. Include the OHJ Indexer JAR file `help`*`indexer-version`*`.jar` in your CLASSPATH.
2. Run the indexer from the command prompt. The indexer supports the following command-line arguments:

   ```
   [-e=charset] dirnameindexfilename
   ```

   where,

Argument	Description
`-e=charset`	The optional (but recommended) charset parameter is the name of the Java-supported character set encoding for the HTML files that are being indexed. If the encoding is not supplied, the default character set encoding of the current system default locale is used. If supplied, the value must be a Java supported character set encoding names; for Java SE, see http://download.oracle.com/javase/1.5.0/docs/guide/intl/encoding.doc.html.
`dirname`	The base directory that contains the HTML files you want to index. The indexer indexes all of the files under this directory (and its subdirectories, if any).
`indexfilename`	The name of the index file to be generated.

For example,`java -mx64m oracle.help.tools.index.JapaneseIndexer -e=MS932 D:\MyHTMLFiles myIndex.idx`

The above command runs the JapaneseIndexer with the encoding set to `MS932`, and indexes the `D:\MyHTMLFiles` directory, creating the `myIndex.idx` file.

Part IV

Oracle Help for Java

This part contains information on using Oracle Help for Java Developer's Kit. It contains the following chapters:

- Chapter 12, "Introduction to Oracle Help for Java Developer's Kit"

  This chapter provides an introduction to Oracle Help for Java Developer's Kit.

- Chapter 13, "Adding OHJ to Your Application"

  This chapter describes how to add an OHJ help to an application.

- Chapter 14, "Enabling Context-Sensitive Help in Your Application"

  This chapter describes how to enable context-sensitive help in an application.

- Chapter 15, "Deploying an OHJ Help System"

  This chapter provides information on deploying OHJ help.

12

Introduction to Oracle Help for Java Developer's Kit

This chapter introduces Oracle Help for Java Developer's Kit (OHJDK), the contents of OHJDK, and how to install OHJDK.

This chapter includes the following sections:

12.1 About Oracle Help for Java Developer's Kit

Oracle Help for Java Developer's Kit is a set of Java components and an API for developing and displaying HTML-based help content in a Java environment. It includes the Oracle Help for Java (OHJ) engine and additional tools necessary for implementing context-sensitive help in Java applets and applications.

The Oracle Help for Java engine is a full-featured Java-based help system for Java applications and applets. It provides pure Java components for navigating and displaying context-sensitive help. Oracle Help for Java is available for free, and may be redistributed as the help system for your application (see the license distributed with the release).

Oracle Help for Java supports help content in several file formats, including extensions of the JavaHelp and Microsoft's HTML Help standards. If you are authoring help content without using a help authoring tool, it is recommended that you read about Oracle Help file formats. If you are using a third party help authoring system that supports OHJ, refer to the documentation provided.

12.2 Oracle Help for Java Runtime in JDeveloper

The Oracle Help for Java runtime library, `ohj.jar`, is distributed as part of JDeveloper. Developers integrating Oracle Help for Java with their applications can modify their project settings to add the Oracle Help for Java library as a dependency. The full Oracle Help for Java release including sample code, helpset authoring wizard, indexer, and demos is available at no cost from the Oracle Help page on Oracle Technology Network at:

`http://www.oracle.com/technetwork/topics/index-083946.html`

12.3 Getting Started with the OHJDK

The Oracle Help for Java Developer's Kit (OHJDK) is a set of Java components and an API for developing and displaying HTML-based help content in a Java environment. It includes the Oracle Help for Java (OHJ) engine and additional tools necessary for implementing context-sensitive help in Java applets and applications.

This section describes the contents of the OHJDK and tells how to set up for developing OHJ help systems. It contains the following subsections:

- Section 12.3.1, "Installing OHJDK"
- Section 12.3.2, "Contents of an OHJDK Release"
- Section 12.3.3, "Setting the Java CLASSPATH for OHJDK Development"

12.3.1 Installing OHJDK

The following steps describe OHJDK installation process:

- Download the latest Oracle Help for Java Developer's Kit from OTN. For Windows, you may download the Windows executable file or Executable JAR file. For Solaris or UNIX, download the binary file.
- In Windows, double click the Windows executable file or executable JAR file to start the installation wizard. By default, the OHJDK is installed at `C:\Program Files\ohelp` location. For Solaris and UNIX, run the binary file to start the installation wizard.
- Open the install folder and verify the files as described in Section 12.3.2, "Contents of an OHJDK Release".

12.3.2 Contents of an OHJDK Release

The OHJDK includes the compiled libraries for the OHJ engine, and libraries for the authoring tools and demonstrations. These libraries are distributed in JAR (Java ARchive) format. The OHJDK also includes documentation, including this document.

The files are discussed in the following sections:

- Section 12.3.2.1, "OHJ Engine"
- Section 12.3.2.2, "Authoring Tools"
- Section 12.3.2.3, "Demonstration Files"
- Section 12.3.2.4, "Documentation"

12.3.2.1 OHJ Engine

These files contain binary files for the OHJ engine implementation and its dependencies. You must redistribute these files with your product application.

Table 12–1 OHJ Binaries

File	Contents
`ohj.jar`	The help engine optimized binaries.
`oracle_ice.jar`	Oracle's customized version of the ICEbrowser from ICEsoft Technologies, Inc. The OHJ engine uses the ICEbrowser for displaying HTML topics.

12.3.2.2 Authoring Tools

The following JAR files contain the implementation for the Helpset Authoring Wizard and the Text Search Indexer. These files are not intended for distribution with your product application.

Table 12–2 OHJ Authoring Tools

File	Contents
`help-wizard.jar`	Helpset Authoring Wizard implementation. Authors can run the Helpset Authoring Wizard using batch files created in the bin directory of their OHJDK installation.
`help-indexer.jar`	OHJ Text Search Indexer implementation. Authors can use the Full-Text Search Indexer to process and create Oracle Help search index (`.idx`) files.

12.3.2.3 Demonstration Files

The following JAR file contains the demonstration programs distributed with the OHJDK. This file is not intended for distribution with your product application.

Table 12–3 OHJ Demonstration JAR File

File	Contents
`help-demo.jar`	Demonstration binaries and source code, plus sample documentation in HTML and the Oracle Help control file formats.

To run the demonstration programs, execute the batch files located in the `bin` subdirectory of your OHJDK installation. The OHJ installer for Windows adds shortcuts to the Windows Start Menu. The sample content used by the demo programs is located in the `demodoc` subdirectory.

When integrating OHJ with your application, it may be helpful for you to examine the source code for the demonstration programs (located in the `help`*`demo-version_num`*`.jar` file. The following demos are particularly helpful:

Table 12–4 OHJ Demonstration Files

File	Contents
`ChoiceDemo.java` (OHJ Features Demo)	Sample Java code that illustrates the following features: ■ Constructing the Help object ■ Adding helpsets (data) ■ Displaying the OHJ navigator window ■ Displaying multiple helpsets
`CSHDemo.java` (Context-Sensitive Help Demo)	Sample Java code that illustrates the following features: ■ Launching the help system from a menu ■ Associating help topics with particular application controls ■ Enabling the F1 key to launch help ■ Implementing right-click pop-up menu help

12.3.2.4 Documentation

The following documentation is included with the OHJDK:

Table 12–5 OHJ Documentation

Document	Content	Location in OHJDK release
OHJ API Reference Documentation	Reference documentation for the OHJ Application Programming Interface (API), provided as JavaDoc.	The doc/javadoc subdirectory of the OHJDK installation. To view the API documentation, open the `index.html` file in an HTML browser.
Oracle Help Guide	This document	The doc subdirectory of the OHJDK installation.

12.3.3 Setting the Java CLASSPATH for OHJDK Development

To develop with the OHJDK, you must add the OHJ engine libraries and toolkit dependencies to your environment class path.

For example:

- On Windows 2000 or XP, set the CLASSPATH variable in the **Environment Variables** dialog, accessed from the **Advanced** tab of the System Properties control panel.
- On Windows NT, set the CLASSPATH variable in the **Environment** tab of the **System Properties** control panel.
- On Windows 95/98, use the `SET` command in your `autoexec.bat` file.
- On UNIX, use the `setenv` command to set an environment variable for your shell.

Consult the documentation for your Java Virtual Machine (JVM) and operating system to determine how to set the CLASSPATH variable.

13

Adding OHJ to Your Application

This chapter describes how to integrate OHJ with your product application, how to construct and create the `Help` object, add help data, how to add tabs, and how to show a topic.

This chapter contains the following sections:

13.1 About Adding OHJ to an Application

The basic steps for adding OHJ to an application are:

1. Construct the `Help` object.
2. Populate the `Help` object with help content, as follows:
   - Create `Book` objects that represent your help data.
   - Add the `Book` objects to the `Help` object.
3. Implement methods for showing the OHJ navigator window and for showing help topics.
4. Dispose of the Help object at the end of your product's lifecycle

13.2 Constructing the Help Object

The `Help` object is the main entry point for Oracle Help for Java. It includes methods for adding help content, showing the OHJ navigator window, and displaying specific topics. There are several options that can only be set at the time the Help object is constructed.

The `boolean` *`combineBooks`* parameter in the `Help` object constructor determines how OHJ displays multiple `Books`, or `HelpSets`. If the boolean *`combineBooks`* parameter is set to `true`, OHJ merges all author-defined views that have the same type and label.

For example, if multiple books include a Keyword Index view with the same label, OHJ displays one keyword index navigator tab with a merged, sorted list of keywords. If the *combineBooks* parameter is set to false, the views from each book are displayed separately, and the end user can select which book to display using a drop-down list in the OHJ navigator window.

The versions of the Help object constructor are summarized below. For more information on API documentation of `oracle.help.Help`, see Section 12.3.2, "Contents of an OHJDK Release".

Table 13–1 Help() Constructors

Constructor	Description
`Help()`	Creates an instance of the `Help` object with the ICEBrowser as the `HTMLBrowser` component used for topic display. This constructor instructs the help system to show all of the views from the added books in one tab panel, and to ignore author defined tab labels in favor of standard tab labels.
`Help(boolean combineBooks, boolean useLabelInfo)`	Creates an instance of the `Help` object with the ICEBrowser as the `HTMLBrowser` component used for topic display. Parameters: ■ `combineBooks` – If `true`, the help system shows all of the views from the books added to the help system in one tab panel. If `false`, the help system creates a different tab panel for each book and allows the end user to choose which book is displayed. ■ `useLabelInfo` – If `true`, the help system uses the author-defined label information for display and view merging. If `false`, the help system uses default labels.
`Help(Class htmlBrowserClass, boolean combineBooks, boolean useLabelInfo)`	Creates an instance of the `Help` object using the specified `HTMLBrowser` component for topic display. The ICEBrowser is the only `HTMLBrowser` subclass currently distributed with OHJ. Parameters: ■ `htmlBrowserClass` – The `HTMLBrowser` subclass to use as the topic display component ■ `combineBooks` – If `true`, the help system shows all of the views from the books added to the help system in one tab panel. If `false`, the help system creates a different tab panel for each book and allows the end user to choose which book is displayed. ■ `useLabelInfo` – If `true`, the help system uses the author-defined label information for display and view merging. If `false`, the help system uses default labels.

Table 13–1 (Cont.) Help() Constructors

Constructor	Description
`Help(Class htmlBrowserClass, boolean combineBooks, boolean useLabelInfo, boolean standAloneMode)`	Creates an instance of the Help object using the specified `HTMLBrowser` component for topic display. The ICEBrowser is the only `HTMLBrowser` subclass currently distributed with OHJ. This constructor contains an extra parameter enabling a standalone mode for running OHJ, where the `Help` object exits the JVM (through `System.exit`) after all help windows have closed. The `standAloneMode` parameter should be set to false if you are launching OHJ from a Java application (otherwise closing help would exit your application!). Parameters: ■ `htmlBrowserClass` – The `HTMLBrowser` subclass to use as the topic display component ■ `combineBooks` – If `true`, the help system shows all of the views from the books added to the help system in one tab panel. If `false`, the help system creates a different tab panel for each book and allows the end user to choose which book is displayed. ■ `useLabelInfo` – If `true`, the help system uses the author-defined label information for display and view merging. If `false`, the help system uses standard default labels. ■ `standAloneMode` – If `true`, the help system exits the JVM when all help windows have been closed. Set this to `false` if you are launching OHJ from your Java application.

13.3 Adding the Help Data

After creating a `Help` object, you must add one or more `Book` objects to it. A `Book` object encapsulates a collection, or "book," of help content.

The HelpSet book implementation handles the preferred Oracle Help file formats, as documented in Oracle Help File Formats. These files include the helpset file, which defines the characteristics of the help system.

The following sections describe how to add the help sets, and other optional features:

- Section 13.3.1, "Constructing a HelpSet"
- Section 13.3.2, "Adding Books to Help"

13.3.1 Constructing a HelpSet

The Table 13–2 lists `HelpSet()` constructors:

Table 13–2 HelpSet() Constructors

Constructor	Description
`HelpSet(URL fileURL)`	Constructs a `HelpSet` object using the helpset file at the specified URL location. Use this constructor when you know the exact location of the helpset file. Parameters: ■ `fileURL` – A URL specifying the exact location of the helpset file.

Table 13–2 (Cont.) HelpSet() Constructors

Constructor	Description
`HelpSet(Class pathClass, String pathExtension)`	Use this constructor when you know only the path to the helpset file relative to your application implementation. Parameters: ■ `pathClass` – One of your application classes. The `HelpSet` object uses the location of this class to determine the location of your helpset file. ■ `pathExtension` – The path to the helpset file relative to the location of `pathClass`. The value of this parameter is appended to the absolute path to the directory containing the `pathClass`. The resulting path should be the path to your helpset file.

For more information, see the API documentation for `oracle.help.library.helpset.HelpSet`.

13.3.2 Adding Books to Help

After you have constructed a `Book` instance using the `HelpSet` constructors, you must add the `Book` to your `Help` instance. This is accomplished by calling the following method on the `Help` instance:

Table 13–3 addBook() Constructors

Constructor	Description
`addBook(Book book)`	This method adds a `Book` instance to the help system. Author-defined views contained in the `Book` are displayed in the navigator window, and topics from the `Book` are available to display. Parameters: ■ `book` - The `Book` instance to add to the help system.

13.4 Adding the Favorites Tab or Custom Tab

You have the option of adding a Favorites tab or a custom tab after the help object is constructed and before the Navigator window is displayed:

- To add a **Favorites** tab to the navigator window, use the function `enableFavoritesNavigator(URL)`. For more information on API documentation of oracle.help.Help.enableFavoritesNavigator(URL), see Section 12.3.2, "Contents of an OHJDK Release", or find this code in `PreviewHelpSet.java` for an example:

```
if (!"".equals(_favoritesPath))
    {
      try
      {
        File file = new File(_favoritesPath);
        _help.enableFavoritesNavigator(file.toURL());
      }
      catch (MalformedURLException e)
      {
        e.printStackTrace();
      }
    }
```

- To add a custom tab, see the API documentation for `oracle.help.navigator.Navigator` in Section 12.3.2, "Contents of an OHJDK Release".

13.5 When to Create the Help object

A single instance of the `Help` object should be created and help data should be added at application startup. You should use this single instance of the `Help` object throughout the application session. It is not efficient to create unique `Help` objects each time the user requests help in your application.

13.6 Showing the Navigator Window

Instruct your `Help` instance to show the OHJ navigator window by calling the `showNavigatorWindow()` method. Some versions of this method take additional parameters to show the navigator window with a specified navigator tab selected (for example, **Contents**, **Index**, **Search**, and so on).

Table 13–4 showNavigatorWindow() Constructors

Constructor	Description
`showNavigatorWindow()`	Shows the navigator window with the first tab of the first book selected. Parameters: - `activeBook` - The `Book` object whose associated set of navigators should be initially displayed when the navigator window is shown.
`showNavigatorWindow(Book activeBook)`	Shows the navigator window with the associated set of navigators from the given book displayed. Parameters: - `activeBook` - The `Book` object whose associated set of navigators should be initially displayed when the navigator window is shown.
`showNavigatorWindow(Navigator activeNavigator)`	Shows the navigator window with a specific navigator tab selected. Use this method to show a specific navigator from a specific book. Parameters: - `activeNavigator` - The navigator to show when initially selected.

13.7 Showing a Topic

Instruct your `Help` instance to show a specific help topic by calling the `showTopic()` method and providing the topic ID and the `Book` instance for that topic. Some versions of this method take additional parameters to specify how the topic should be displayed.

Table 13–5 showTopic() Constructors

Constructor	Description
`showTopic(Book book, String topicID)`	Shows the given topic from the given book in a currently existing topic window. If no topic windows currently exist, a new topic window is created with the default size and position. Parameters: ■ `book` - The `Book` from which to show the topic. ■ `topicID` - The topic ID for the topic to show (as specified in the map file of the book)
`showTopic(Book book, String topicID, boolean alwaysCreate)`	Shows the given topic from the given book. If `alwaysCreate` is `true`, a new window is always created. If `false`, a new window is created if no windows currently exist. Parameters: ■ `book` - The `Book` to show the topic from. ■ `topicID` - The topic ID for the topic to show (as specified in the map file of the book). ■ `alwaysCreate` - If `true`, always create a new window. If `false`, reuse a window if possible
`showTopic(Book book, String topicID, boolean alwaysCreate, Point location, Dimension size)`	Shows the given topic from the given book. If `alwaysCreate` is `true`, a new window is always created; if it is `false`, a new window is created if no windows currently exist. The topic window is shown using the given location and size. Parameters: ■ `book` - The `Book` to show the topic from. ■ `topicID` - The topic ID for the topic to show (as specified in the map file of the book). ■ `alwaysCreate` - If `true`, always create a new window. If `false`, reuse a window if possible. ■ `location` - Location of the topic window in screen coordinates. ■ `size` - Size of the topic window in pixels.

13.7.1 Catching TopicDisplayExceptions

Exceptions are thrown by the `showTopic()` method when an error is encountered when trying to display a topic. For example, if you attempt to display a topic ID which is not in the map file, a `TopicDisplayException` is thrown. By catching the `TopicDisplayException`, you have the opportunity to act when an error occurs. In the following example, an author-defined error topic is displayed when `TopicDisplayException` is thrown.

For example:

```
try
{
  myHelp.showTopic(myhelpset, "nonExistingTopic");
}
catch (TopicDisplayException e)
{
  //An error has occurred, try to show an error topic
  myHelp.showTopic(myhelpset, "onErrorTopic");
}
```

13.8 Disposing of the Help Object

Disposing of the `Help` object frees OHJ resources. You should dispose of the `Help` object when you no longer need the help engine. Typically, this would be done at end of the user's application session. Disposing closes all files and frees memory used by the `Help` object. To dispose of the `Help` object, call the `dispose()` method:

Table 13–6 dispose() Constructors

Constructor	Description
`dispose()`	Dispose of the help system. This method frees up all resources used by the help system. Applications should call this method when they do not need help anymore. You should not call any methods on the `Help` object after calling `dispose()`.

The `dispose()` method eliminates any references to objects held by the OHJ classes, but not other references that you have created from your application to Help objects.

Therefore, after you call `dispose()` you should eliminate any references to OHJ objects (`Help` or `Book` objects) in your application code. This allows the Java garbage collection process to free the OHJ objects.

14

Enabling Context-Sensitive Help in Your Application

This chapter describes how to enable context-sensitive help in your application and how to map topic IDs with the help topics.

This chapter includes the following sections:

14.1 About Context-Sensitive Help

A *context-sensitive* help topic is one that is associated with a context in a product's user interface and which can be launched from that context. For example, a context-sensitive help system may contain topics that describe the product's menus and dialog boxes. When a user requests help for one of those controls, the appropriate topic for that control is displayed.

Oracle Help for Java provides mechanisms for associating help topics with user interface controls and for launching context-sensitive help when the user presses the F1 key or selects Help from a right-click context menu. OHJ also provides an API for explicitly displaying the help topic associated with a component.

To provide context-sensitive help for an application, the help system source must include one or more map files and you must add the appropriate help code to your application code.

14.2 Mapping Topic IDs to Help Topics

A context-sensitive help system must include one or more map files that map topic IDs to help topics. Ideally, the map file is created by the help author, not the programmer. OHJ supports two kinds of map files:

When an OHJ system is implemented using a HelpSet, the map file must use the XML file format, also the recommended format for Oracle Help systems. For more information about the map file, see Section 5.3, "Map Files".

You may want to define a set of Java constants in your application code that map to topic IDs in the map file. When using OHJ methods that require topic IDs (such as `Help.showTopic()`), you can use the constant variable names instead of using explicit topic IDs from the map file. This helps you avoid changing code in several places if

topic IDs in the map file are changed later. This is particularly useful when help authors are responsible for maintaining map files.

14.3 Programming Your Application to Support Context-Sensitive Help

The standard Java toolkits (AWT and Swing) do not include built-in mechanisms for associating help topics with components. However, OHJ provides a generic way to implement context-sensitive help in Java applications, using the OHJ `CSHManager` class. This help manager provides a way to associate help topics with user interface components and to enable F1 and right-click context-sensitive help.

14.4 Using CSHManager to Implement Context-Sensitive Help

OHJ's `CSHManager` class provides a generic way to associate help topics with user interface controls and to launch context-sensitive help when the end user presses the F1 key or selects Help from a right-click context menu. OHJ also provides an API for explicitly displaying the help topic associated with a component.

The following sections provide an introduction to using `CSHManager`:

14.4.1 CSHManager Constructors

You should create an instance of the `CSHManager` class before creating user interface components. The `CSHManager` constructor is summarized in Table 14–1.

Table 14–1 CSHManager() Constructors

Constructor	Description
`CSHManager(Help help)`	Creates a new instance of the `CSHManager` class. All subsequent calls to this manager object use the specified Help object. Parameters: ▪ `help` – The `Help` object to be used for displaying help.

14.4.2 Setting the Default Book

If you only have one `Book` of help content, you may want to use the `setDefaultBook()` method to define it as the default book for context-sensitive help. This enables you to call the `addComponent()` method without entering the `book` parameter.

Table 14–2 setDefaultBook() Constructors

Constructor	Description
`setDefaultBook(Book book)`	Sets the specified `Book` as the default `Book`. This is used as the default `Book` for components registered without a specified `Book`. Parameters: ▪ `book` – The `Book` to be set.

If you have more than one `Book`, the default `Book` is only used for those components for which you have not assigned a specific `Book`. In a multiple `Book` help system, you should, in general, assign specific `Books` to your components.

14.4.3 Associating Topic IDs with User Interface Components

Use the `addComponent()` method to associate topic IDs, as defined in the map file, with Java user interface components. You can call one of the versions of this method for each component that requires context-sensitive help.

Table 14–3 addComponent() Constructors

Constructor	Description
`addComponent(Component component, String topicId)`	Registers a component with the help manager.The default `Book` is used for looking up the topicId. If no default `Book` is registered at the time of calling this method, then the component is not registered with the help manager, and help is not shown if show`HelpForComponent()` is later called for this component.For components registered using this method, `CSHManager` does not display help in response to an F1 key press or a right mouse-click event. Parameters: ■ `component` – The component to be added. ■ `topicId` – The topic ID to associate with this component.
`addComponent(Component component, Book book, String topicId)`	Registers a component with the help manager. The provided book is used for looking up the topicId. For components registered using this method, `CSHManager` does not display help in response to an F1 key press or right mouse-click event. Parameters: ■ `component` – The component to be added. ■ `book` – The Book containing the help topic. ■ `topicId` – The topic ID to associate with this component.
`addComponent(Component component, String topicId, boolean needF1Help, boolean needPopupHelp)`	The default `Book` is used for looking up the `topicId`. If no default `Book` is registered at the time of calling this method, the component is not registered with the help manager, and help does not show if show`HelpForComponent()` is later called for this component. Parameters: ■ `component` – The component to be added. ■ `topicId` – The topic ID to associate with this component. ■ `needF1Help` – If `true`, displays help for this component in response to F1 key press events. ■ `needPopupHelp` – If `true`, displays a Help pop-up menu upon right mouse click and launches context-sensitive help if the Help menu item is selected.

Table 14–3 (Cont.) addComponent() Constructors

Constructor	Description
`addComponent(Component component, Book book, String topicId, boolean needF1Help, boolean needPopupHelp)`	Registers a component with the help manager. The provided book is used for looking up the `topicId`. Parameters: ■ `component` – The component to be added. ■ `topicId` – The topic ID to associate with this component. ■ `needF1Help` – If `true`, displays help for this component in response to F1 key press events. ■ `needPopupHelp` – If `true`, displays a **Help** pop-up menu upon right mouse click and launches context-sensitive help if the Help menu item is selected. ■ `needF1Help` – If `true`, displays help for this component in response to F1 key press events. ■ `needPopupHelp` – If `true`, displays a Help pop-up menu upon right mouse click and launches context-sensitive help if the **Help** menu item is selected.

14.4.4 Explicitly Showing Help for Components

Call the `showHelpForComponent()` method on the `CSHManager` when you explicitly want to display the help topic associated with a component. For example, to launch context-sensitive help when the user presses a **Help** button in a dialog box, you could call this method in the button's event handler.

Table 14–4 showHelpForComponent() Method

Constructor	Description
`showHelpForComponent(Component component)`	Shows help for the specified component. If the specified component was not registered through `addComponent()`, no help is shown.

15

Deploying an OHJ Help System

This chapter describes how to deploy an OHJ help system with your application, lists files that must be provided with OHJ files, and how to create JAR files.

This chapter includes the following sections:

15.1 Supported Java Virtual Machines

Since OHJ is a Java application (or applet), it must run in a Java Virtual Machine (JVM). That means that you must ensure that your users have an appropriate JVM installed. If you are using OHJ as help for a Java application, that application must also run in a virtual machine, so it is likely that the JVM is being distributed with your application.

15.2 Which OHJ JAR Files Must Be Shipped

The following JAR (Java ARchive) files must be redistributed as part of your product install. These files contain the OHJ engine implementation and its dependencies. For more information about these files, see Section 12.3.2, "Contents of an OHJDK Release".

- `help`*`version_num`*`.jar`
- `oracle_ice.jar`

15.3 Deploying Compressed Help Content in JARs

Java based applications are often distributed in JAR format. You may want to include your Help content as a JAR file in your distribution, as HTML content files are significantly reduced in size by JAR compression.

15.3.1 Creating JAR files

To create JAR files, use the JAR tool that is distributed with your Java Development Kit. If you are using a third party help authoring system that supports OHJ, it may also be able to perform this task.

Special care should be given to the organization of files and their assigned paths within the JAR archives. The directory structure within your help JAR file must match

the directory structure for your product implementation JAR files. Your help content should be after the location of the implementation class that you use as a reference point.

For example,

If your product implementation JAR includes classes with the following directory structure,

```
/com/yourCompany/yourProduct/SomeProduct.class
/com/yourCompany/yourProduct/SomeClass.class
```

then your help JARs should contain the matching paths so that the help content is below the product implementation classes in the shared directory structure:

```
/com/yourCompany/yourProduct/help/ProductHelp.hs
/com/yourCompany/yourProduct/help/TableOfContents.xml
/com/yourCompany/yourProduct/help/SomeHelpTopic.html
```

15.3.2 Which Book Constructor to Use

To create a `Book` object with help content loaded from a JAR file, the help JAR file and implementation JAR files must be on the system CLASSPATH. If you have organized the JAR files as described, then you know the location of the help content is relative to the location of your implementation classes on the CLASSPATH.

Thus, you use the versions of the `HelpSet` constructors that accept a Java class and a relative path from that class.

For example, if you have two JAR files, construct the `helpset` object using the following code:

```
import com.yourCompany.yourProduct.SomeProduct;
helpset myhelpset = new helpset(SomeProduct.class, "help/ProductHelp.hs");
```

Part V

Oracle Help for the Web

This part contains information on using Oracle Help for Web. It contains the following chapters:

- Chapter 16, "Deploying OHW Demo File"

  This chapter describes how to deploy OHW demo files. If you are new to OHW, it is recommended to deploy the demo file before you start deploying your own help system

- Chapter 17, "Understanding OHW Deployment"

  This chapter describes the OHW deployment process.

- Chapter 18, "Implementing Context-Sensitive Help in a Web Application"

  This chapter describes how to implement a context-sensitive OHW help in a web application.

- Chapter 19, "ADF Rich Client Help Provider"

  This chapter describes the ADF Rich client help providers and how OHW could be used as a help provider.

16

Deploying OHW Demo File

This chapter describes how to deploy OHW demo file on standalone Oracle WebLogic Server and JDeveloper integrated Oracle WebLogic Server, and how to run the demo file.

This chapter includes the following sections:

16.1 About Deploying OHW Demo Files

The OHW demo EAR file contains all class libraries that you require to view the demo and try out the release. It includes sample helpsets, OHW servlet file, and XML configuration files. You can deploy the demo file to experience the OHW interface, or replace the existing helpsets and add your own.

16.2 Understanding the OHW Demo Files

The OHW demo file is available in three variants: `ohw-rcf-demo-thin.ear`, `ohw-rcf-demo-thin-shared.ear`, and `ohw-rcf-demo-thick.ear`.

The `ohw-rcf-demo-thick.ear` contains ADF, JSF and JSTL libraries preconfigured for deployment. The file is recommended if you are not using a supported application server or JDeveloper, or if you are using a supported application server but do not have the libraries installed. For more information about supported application servers, see the "Certification Information" page on OTN.

The `ohw-rcf-demo-thin.ear` does not contain ADF, JSF and JSTL libraries, hence it is required that the libraries must be installed on the application server before deployment. JDeveloper is required for this demo file. For more information on how to deploy the demo file, see Section 16.3, "Deploying the OHW Demo EAR File on Standalone Oracle WebLogic Server".

The `ohw-rcf-demo-thin-shared.ear` does not contain ADF, JSF and JSTL libraries. The demo file also does not contain OHW related jars (such as `ohw-rcf.jar`, `ohw-share.jar`, and `help-share.jar`), but uses the OHW jars from OHW-RC library. Hence, if you are using `ohw-rcf-demo-thin-shared.ear` demo file, you should deploy

it from JDeveloper. The process of deploying `ohw-rcf-demo-thin-shared.ear` demo file is same as deploying `ohw-rcf-demo-thin.ear` file in JDeveloper. For more information on how to deploy the demo file, see Section 16.4, "Deploying the OHW Demo EAR File on JDeveloper Integrated Oracle WebLogic Server." If you want to deploy the demo file on WebLogic Server, you'd need to download the OHW-RC library files from OTN. The process of deploying `ohw-rcf-demo-thin-shared.ear` demo file is same as deploying `ohw-rcf-demo-thin.ear` file on standalone WebLogic Server. For more information on how to deploy the demo file, see Section 16.3, "Deploying the OHW Demo EAR File on Standalone Oracle WebLogic Server."

> **Note:** Do *not* deploy both `ohw-rcf-demo-thin.ear` and `ohw-rcf-demo-thick.ear` on Oracle WebLogic Server as they would conflict when you run the demo EAR files.

Both OHW demo EAR files, `ohw-rcf-demo-thin.ear` and `ohw-rcf-demo-thick.ear`, contain two OHW sample helpsets along with their help topics, helpset file, and control files. They also contain `ohwconfig.xml` which is needed to configure OHW.

When extracted into a directory, the OHW demo file extracts files into their respective name directories, `ohw-rcf-demo-thin` and `ohw-rcf-demo-thick`. The EAR file contains a WAR file and a `META-INF` directory, which contains the `application.xml` file. The WAR file contains all helpset directories along with their help topics, helpset file, and control files.

Table 16–1 describes the files and directories in OHW demo EAR files.

Table 16–1 OHW Demo Files and Directories

File	Description
`application.xml`	Java EE application file. The file is available in `META-INF` directory.
`helpsets` directory	Web module containing two helpsets: `ohguide` and the sample `shakespeare`, in their respective directories. The helpsets directory exists in the respective WAR files (`ohw-rcf-demo-thick.war` or `ohw-rcf-demo-thin.war`).

`trinidad-config.xml`
`faces-config.xml`
`web.xml`
`weblogic.xml`
	Contains configuration and deployment information that affects OHW thin and thick clients.
	The files are available in `WEB-INF` directory of WAR files.

`help-share.jar`
`ohw-rcf.jar`
`ohw-share.jar`
	OHW library files required for deployment.
	The files are available in `WEB-INF\lib` directory of WAR files.

The following library files are available for thick clients only: ▪ `adf-richclient-api-11.jar` ▪ `adf-richclient-impl-11.jar` ▪ `trinidad-api.jar` ▪ `trinidad-impl.jar`	

Table 16–1 (Cont.) OHW Demo Files and Directories

File	Description
`ohwconfig.xml`	OHW configuration file. The file is available in `helpsets` directory.

16.3 Deploying the OHW Demo EAR File on Standalone Oracle WebLogic Server

Before you start the demo file deployment, verify that ADF, JSF, and JSTL libraries are installed in Oracle WebLogic Server. For more information, see Section 17.3, "Verifying OHW Library Files".

After the library verification, deploying the demo EAR file is a very simple process:

1. Download the OHW demo EAR file from OTN. The name of the demo file is `ohw-rcf-demo-thick.ear`. This file includes OHW library files and sample helpsets.
2. Start the Oracle WebLogic Administration Console and navigate to **Deployments** page.
3. In the Deployments page, click **Install** to start the deployment wizard.
4. In the **Path** field, enter the location where you saved the `ohw-rcf-demo-thick.ear` file, or in the Current Location, browse and select the EAR file.

   Click **Next** to continue.
5. In the Choose targeting style page, select **Install this deployment as an application**, and click **Next**.
6. In the Optional Settings page, verify the settings. It is recommended that you leave the settings as default. Click **Next** to continue, or click **Finish** to complete the deployment.
7. In the Additional Configuration page, select **Yes, take me to the deployment's configuration screen** and click **Finish** to complete the deployment.

   The deployment wizard, after successful deployment, returns you to the Settings page of `ohw-rcf-demo-thick.ear`. If there are errors while deploying the file, you are navigated to Deployment home page where the errors are listed in red text.

16.4 Deploying the OHW Demo EAR File on JDeveloper Integrated Oracle WebLogic Server

To deploy the demo file on Oracle JDeveloper, follow these steps:

1. Download the OHW demo EAR file from OTN. The name of the demo file is `ohw-rcf-demo-thin.ear`. This file includes OHW and sample helpsets.
2. Start JDeveloper.
3. Create a new application from the `ohw-rcf-demo-thin.ear` file. From the **File** menu, select **New**. In the New Gallery dialog, select **Applications** under General category, and then select **Application from EAR File** from the Items list.
4. In the Location page of Create Application from EAR File wizard, browse and select the `ohw-rcf-demo-thin.ear` file. You may also change the application name and the location of application. Click **Next** to continue.

5. The EAR Modules page of the wizard shows the project name and the module name. Click **Next** to continue, or click **Finish** to create the application from EAR file.
6. The Finish page of the wizard shows a summary of your settings for the application. Click **Finish** to create the application from EAR file. JDeveloper extracts all files from the EAR file and creates an application which is ready to edit and deploy.
7. In the Applications window, open the project and edit the desired files.
8. To deploy the application, start the integrated Oracle WebLogic Server instance. From **Run** menu, choose **Start Server Instance** to start the integrated Oracle WebLogic Server.
9. In the Applications window, select the **ohw-rcf-demo-thin** project. From the **Build** menu, select **Deploy**.

   If you are deploying the demo application for the first time, select **ohw-rcf-demo-thin** from the submenu, and follow these steps:

   a. On the Deployment Action page of Deploy ohw-rcf-demo-thin dialog box, select **Deploy to Application Server** as the deployment action, and then click **Next**.
   b. On the Select Server page, select **IntegratedWebLogicServer** as the application server, and then click **Next**.
   c. On the Weblogic Options page, verify the settings. It is recommended that you leave the settings as default. Click **Next** to continue, or click **Finish** to complete the deployment.
   d. On the **Summary** page, verify your settings, and click **Finish**.

   If you have deployed the demo application before and want to continue using the same settings, you can choose **ohw-rcf-demo-thin to IntegratedWebLogicServer** from the submenu.

   JDeveloper starts the deployment process and the status of the deployment is reflected in the Log window. When the application is successfully deployed, JDeveloper prompts with `Deployment finished` message in the Log window.

16.5 Running the OHW Demo EAR File

After successful deployment of demo file, open your browser and navigate to the following URL:

```
http://<yourHost>:<yourPort>/<jdevProjectName>/ohguide/
```

If you have installed the OHW demo EAR file using Oracle WebLogic Administration Console, use the following URL:

```
http://localhost:7101/ohw-rcf-demo/ohguide
```

If you have installed the OHW demo EAR file using Oracle JDeveloper, use the following URL:

```
http://localhost:7101/ohw-rcf-demo-thin/ohguide
```

For more information on user interface of OHW, see Chapter 3, "Oracle Help for the Web User Interface".

17

Understanding OHW Deployment

This chapter describes the OHW deployment process and configuration files, how to install OHW artifacts, how to configure OHW to display custom helpsets, how to change the access URL, and how to deploy OHW as a standalone web application and a part of another web application.

This chapter includes the following sections:

17.1 About OHW Deployment

After help authors have finished creating the help contents, then OHW administrator must modify the Web configuration to deploy those help contents using OHW.

You can deploy OHW in multiple ways. However, there are certain tasks that are common to all deployment modes. This chapter describes those common tasks that are a prerequisite for further steps.

If you are new to OHW, you may start with deploying the demo file. For more information, see Chapter 16, "Deploying OHW Demo File". The demo EAR file includes all files needed to deploy the sample helpsets immediately.

If you are creating a new OHW helpset, the following sections help you understand the OHW deployment process and describe the steps required to create and deploy your own OHW help system.

Unless configuration files have already been created and the application server configured, the OHW administrator must perform the tasks described in the subsequent sections to deploy an OHW help system.

17.2 Verifying Requirements and Dependencies

The following requirements must be verified for OHW:

Table 17–1 OHW Deployment Minimum Requirements

Requirement	Description
Application Server	The OHW requires a Java EE 1.5 compatible application server that could support Java Servlet, JSP and JSF. Oracle WebLogic Server, standalone or integrated with JDeveloper, is recommended as it requires minimal configuration effort. For more information about supported application servers, see Application Servers section in "Certification Information" on OTN.
Client	The client receives only HTML, and all it requires is a web browser to display and view the OHW help content. The web browser must have JavaScript support enabled. OHW is supported in Microsoft Internet Explorer 7, Microsoft Internet Explorer 8, Mozilla FireFox 2, Mozilla FireFox 3, Apple Safari, and Google Chrome. For more information about supported browsers, see the ADF Faces and Data Visualizations section in "Release Notes - JDeveloper 11*g*" on OTN.
Rich ADF Faces	OHW needs Rich ADF Faces libraries and their dependencies to be available. The application server should also be configured for ADF-based applications by installing the correct JAR files or by running the ADF Runtime Installer using Oracle JDeveloper. For more information, see the online help of Oracle WebLogic Application Console.

17.3 Verifying OHW Library Files

The application server where you deploy the OHW help files, must be configured to support Rich ADF Faces, because OHW depends on that technology.

If you are using Oracle WebLogic Server, review your Oracle WebLogic Application Console and confirm that following libraries are also deployed:

- ADF (`adf.oracle.domain(1.0,11.1.1.2.0)`)
- Java Server Faces (`jsf(1.2,1.2.9.0)`)
- JavaServer Pages Standard Tag Library (`jstl(1.2,1.2.0.1)`)

The libraries are listed on Deployment page of Oracle WebLogic Application Console. If the libraries are not installed, extend your WebLogic domain using Oracle WebLogic Configuration Wizard to include Oracle JRF libraries. After including Oracle JRF libraries, restart your Oracle WebLogic Server and the libraries would be listed in Deployments page of Oracle WebLogic Application Console. For more information about extending a domain, see *Administering Oracle ADF Applications*.

If you are not using Oracle WebLogic Server, ensure that all jar files of `ohw-rcf-demo-thick.ear` are copied in `\WEB-INF\lib` directory of WAR deployment file. You can download the library files from OTN, or copy them from the demo file. The libraries are available in `ohw-rcf-demo-thick\ohw-rcf-demo-thick\WEB-INF\lib` directory of `ohw-rcf-demo-thick.ear` file.

17.4 Installing OHW Artifacts

There are some files needed to be installed on the server to make OHW working. The details about this will be shown in different possible deployment of OHW topics.

The OHW distributable components consist of JAR files like `ohw-rcf.jar`, `ohw-share.jar`, `help-share.jar`, and `ohw-rcf-webapp.zip`. The `ohw-rcf-webapp.zip` contains the `helppages` directory, which contains installable files like jspx (XML style of a JSF page) that are needed to run OHW properly.

All artifacts are available on OTN for download.

17.5 Understanding OHW Configuration Files

Before you start deploying the OHW helpset, there are some files that must be created or modified to configure OHW correctly. The following information helps you understand the XML configuration files:

- **`application.xml`**: A manifest of all web modules that run under a given Java EE application. It points to each web module of each product that is deployed.

  The name and location of `application.xml` is fixed by the Java EE standard. The file is available in `<OHW_HOME>\META-INF\` directory.

- **`web.xml`**: Sets the initialization parameters for the OHW components, including the location of the OHW configuration file. There is one instance of `web.xml` for each web module. The file is available in `<OHW_HOME>\<OHW_deployment_name>\WEB-INF\` directory.

  There is a minimum set of parameters must be set to assure all prerequisites for OHW are loaded and initialized correctly:

  – JSF servlet and servlet mapping

  – Trinidad resource servlet, servlet mapping, filter and filter mapping

  – OHW filter and filter mapping

  – help instance servlet and servlet mapping

  The following example shows a sample `web.xml` file:

```
<context-param>
  <param-name>org.apache.myfaces.trinidad.CHANGE_PERSISTENCE</param-name>

<param-value>oracle.adf.view.rich.change.FilteredPersistenceChangeManager</para
m-value>
</context-param>
<filter>
  <filter-name>trinidad</filter-name>

<filter-class>org.apache.myfaces.trinidad.webapp.TrinidadFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>trinidad</filter-name>
  <servlet-name>Faces Servlet</servlet-name>
</filter-mapping>
<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>/faces/*</url-pattern>
</servlet-mapping>
<servlet>
  <servlet-name>resources</servlet-name>
```

```
<servlet-class>org.apache.myfaces.trinidad.webapp.ResourceServlet</servlet-clas
s>
</servlet>
<servlet-mapping>
  <servlet-name>resources</servlet-name>
  <url-pattern>/adf/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>resources</servlet-name>
  <url-pattern>/afr/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>resources</servlet-name>
    <url-pattern>/ohr/*</url-pattern>
  </servlet-mapping>

<!-- configuration for product1 help front servlet -->
<servlet>
  <servlet-name>product1</servlet-name>
  <servlet-class>oracle.help.web.rich.OHWServlet</servlet-class>
  <init-param>
    <param-name>ohwConfigFileURL</param-name>
    <param-value>/helpsets/product1/ohwconfig.xml</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>

<!-- configuration for product2 help front servlet -->
<servlet>
  <servlet-name>product2</servlet-name>
  <servlet-class>oracle.help.web.rich.OHWServlet</servlet-class>
  <init-param>
    <param-name>ohwConfigFileURL</param-name>
    <param-value>/helpsets/product2/ohwconfig.xml</param-value>
  </init-param>
  <load-on-startup>3</load-on-startup>
</servlet>

<!-- servlet mappings for the OHW-RC front servlets -->
<servlet-mapping>
  <servlet-name>product1</servlet-name>
  <url-pattern>/product1/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>product2</servlet-name>
  <url-pattern>/product2/*</url-pattern>
</servlet-mapping>

<!-- OHW-RC servlet filter definition and mappings -->
<filter>
  <filter-name>OHWRCFRequestFilter</filter-name>
  <filter-class>oracle.help.web.rich.OHWFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>OHWRCFRequestFilter</filter-name>
  <servlet-name>Faces Servlet</servlet-name>
</filter-mapping>
```

- **ohwconfig.xml** (default file name): Specify which helpsets to display and how to present them. You can also specify locales, branding information, and various other settings. For information about this configuration file, see Chapter 8, "Oracle Help for the Web Configuration File". The name and location of this file is set as a OHW front servlet initialization parameter, which is handled differently for each servlet container. The `ohwConfigFileURL` initialization parameter could take a path that contains `{%some_parameter}` tokens. The token's value is resolved from Java `System.getProperty` calls. Always consult the servlet container documentation for current and complete information.

  The file is available in `<OHW-RC_HOME>\<OHW-RC_deployment_name>\helpsets` directory. Note that you can also specify `ohwConfigURL` using the `prop` system property. For example, in `web.xml`, the `ohwConfigURL` would be configured as:

  ```
  <param-name>ohwConfigFileURL</param-name>
  <param-value>file:///{%prop}/help/ohwconfig.xml</param-value>
  ```

  In `adf-settings.xml`, the `ohwConfigURL` would be configured as:

  ```
  <property>
      <property-name>ohwConfigFileURL</property-name>
      <value>file:///{%prop}/help/ohwconfig.xml</value>
  </property>
  ```

- **trinidad-config.xml**: Specify the configuration for the Trinidad and ADF Rich components. It specifies the skinning option to be used in OHW. The file is available in `<OHW_HOME>\<OHW_deployment_name>\WEB-INF\` directory.
- **faces-config.xml**: This is the JSF configuration file. The file is available in `<OHW_HOME>\<OHW_deployment_name>\WEB-INF\` directory.

  You must add the ADF Faces render kit information in this file:

  ```
  <application>
    <default-render-kit-id>oracle.adf.rich</default-render-kit-id>
    <locale-config>
      <default-locale>en</default-locale>
    </locale-config>
  </application>
  ```

17.6 Configuring OHW to Display Custom Helpsets

The instructions in this section helps you create the directory structure required for OHW help system, add your custom helpset files in the correct location, create or modify the configuration files, and deploy the help system on application server.

The instructions in this section also assume that you have installed the OHW demo EAR file and you have a knowledge of the demo EAR file's directory structure. If you have not installed the demo file, install it following instructions in Chapter 16, "Deploying OHW Demo File".

Follow these steps to set up your OHW help system:

1. Set up the directory structure as following:

   ```
   <OHW-RC_HOME>
     |
     — <OHW-RC_deployment_name>
         |
   ```

```
      – helppages
      – helpsets
          |
          – <custom_helpset_directory>
      – META-INF
      – WEB-INF
          |
          – lib
  – META-INF
```

For example:

```
my_module
  |
  – my_module_help
      |
      – helppages
      – helpsets
          |
          – my_ModuleHelpset
      – META-INF
      – WEB-INF
          |
          – lib
  – META-INF
```

2. Create your own helpset directory. Place all your help files in or under `<OHW_HOME>\<OHW_deployment_name>\helpsets\<custom_helpset_directory>` directory, including the helpset file, topic files, and the other control files (index, table of contents, and so on). Also, place any JAR files here, if you are using JAR files for your helpset. You can use JARred and unJARred helpsets together in the same deployment.
3. Update and configure the configuration file. Copy the demo EAR's `ohwconfig.xml` from `ohw-rc-thick-demo\ohw-rc-thick-demo\helpsets` directory and save it in your `<OHW_HOME>\<OHW_deployment_name>\helpsets` directory. Edit the file according to your requirement:
   a. Modify the `<books></books>` section to direct it to your helpset. For example:

```
<books>
  <helpSet id="myModule" location="my_ModuleHelpset/my_ModuleHelpset.hs" />
</books>
```

   b. Remove the helpsets which you do not want to provide from the `<books></books>` section. If removed, the helpsets would not appear in the helpset switcher dropdown list of the OHW user interface. If you have only one `<helpSet>` element in the `<books></books>` section, the helpset switcher is not available.
   c. Update the `<brandings></brandings>` section to display your own brand. For example:

```
<brandings>
  <branding text="My Module" />
</brandings>
```

For more information about `ohwconfig.xml` file behaviors you can configure, see Chapter 8, "Oracle Help for the Web Configuration File".

4. Copy the ohw-rc-thick-demo\ohw-rc-thick-demo\helppages directory to your <OHW_HOME>\<OHW_deployment_name> directory.

5. Copy the following library files from JDeveloper to <OHW_HOME>\<OHW_deployment_name>\WEB-INF\lib directory.

File Name	Location
JSF library file (jsf-1.2.war)	<JDEV_HOME>\wlserver_10.3\common\deployable-libraries
JSTL library file (jstl-1.2.war)	<JDEV_HOME>\wlserver_10.3\common\deployable-libraries
ADF library files (adf-richclient-api-11.jar, adf-richclient-impl-11.jar)	<JDEV_HOME>\oracle_common\modules\oracle.adf.view_11.1.1

6. Configure faces-config.xml and trinidad-config.xmlto configure JSF and JSTL support in OHW. Copy the XML files from ohw-rc-thick-demo\ohw-rc-thick-demo\WEB-INF directory and save them in <OHW_HOME>\<OHW_deployment_name>\WEB-INF directory.

   The faces-config.xml is the JSF configuration file where you register a JSF application's resources and define the page-to-page navigation rules. The trinidad-config.xml enables you to configure ADF Faces features. Like faces-config.xml, the trinidad-config.xml file has a simple XML structure that enables you to define element properties using the JSF Expression Language (EL) or static values.

7. If you are using Oracle WebLogic Server, copy the weblogic.xml from ohw-rc-thick-demo\ohw-rc-thick-demo\WEB-INF directory and save it in your <OHW_HOME>\<OHW_deployment_name>\WEB-INF directory. Then, edit the file to your requirements.

8. Copy the web.xml from ohw-rc-thick-demo\ohw-rc-thick-demo\WEB-INF directory and save it in <OHW_HOME>\<OHW_deployment_name>\WEB-INF directory. Then, edit it to your requirements:

   a. Modify the <display-name></<display-name> and <description></description> section to display your custom helpset name. For example:

   ```
   <web-app>
     <display-name>My Module</display-name>
     <description>My module help</description>
   </web-app>
   ```

   b. Optionally, you may want to edit the <servlet-name> element under <servlet> element to change your URL used to access OHW. For more information about changing the access URL, see Section 17.7, "Changing the OHW Access URL".

9. Compress the <OHW-RC_deployment_name> directory into a WAR file.

10. Copy the application.xml from ohw-rc-thick-demo\META-INF directory and save it in <OHW-RC_HOME>\META-INF directory. Then, edit the file to your requirements. In this file, provide the web module name of each product that you want to deploy.

    You may also specify the WAR file name, created in step 9, in <web-uri></web-uri> element. If you want to change the access URL of the

application, update the <context-root><context-root> element. For more information, see Section 17.7.2, "Changing the access URL to another name".

11. Compress the <OHW_HOME> directory into a EAR file.
12. Start the Oracle WebLogic Server and deploy the EAR file. If Oracle WebLogic Server is running, you must shut it down and then restart it before the changes made since you last started the servlet are available. For more information about deploying an EAR file, see the "Install an Enterprise application" section in *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.
13. Direct the browser to http://<hostname>:<port>/<OHW-RC_deployment_name>/ohguide/, where <hostname> is the name of the system on which Oracle WebLogic Server is installed.

    The first page of the demo help system displays in the browser. If there are multiple helpsets, use the dropdown list in the toolbar to select a helpset, then click the helpset switcher to display the TOC and index from the selected helpset only. The text search searches only for items in the selected helpset.

17.7 Changing the OHW Access URL

The URL to access OHW is http://<hostname>:<port>/mymodule/ohguide/, where <hostname> is the name of the system on which OHW and Oracle WebLogic Server are installed.

You can change this URL in the following ways:

- Changing the final URL element of the access URL
- Changing the access URL to another name

17.7.1 Changing the final URL element of the access URL

To change the help at the end of the URL, edit web.xml in <OHW_HOME>\<OHW_deployment_name>\WEB-INF.

The <servlet-mapping> parameter <url-pattern> specifies the URL used to access OHW. For example, if you change <url-pattern> from the default /help/* to /onlinereference/*, the URL used to access OHW would become http://<hostname>:<port>/mymodule/onlinereference/.

For example:

```
<servlet-mapping>
   <servlet-name>mymodule</servlet-name>
   <url-pattern>/onlinereference/*</url-pattern>
</servlet-mapping>
```

17.7.2 Changing the access URL to another name

To change the access URL for your application, edit the <context-root> element entry under <web> element in application.xml, located in <OHW_HOME>\META-INF:

```
<web>
   <web-uri>my_module.war</web-uri>
   <context-root>my_module</context-root>
</web>
```

For example, if you want the OHW access URL to be `http://<hostname>:<port>/jdeveloper/help/`, modify the root element:

```
<web>
   <web-uri>my_module.war</web-uri>
   <context-root>jdeveloper</context-root>
</web>
```

17.8 Deploying OHW as a Standalone Web Application

One of the ways that OHW can be deployed is to have it as a standalone Web application. To deploy OHW as a standalone application, an OHW WAR file, containing all files needed to run the OHW, must be copied into a separate deployment directory in the application server that has a dedicated context path.

The OHW administrator must perform some primary tasks, and then go on to deploy the OHW help system as a standalone Web application, as follows. To know more about the tasks, see Chapter 17, "Understanding OHW Deployment".

17.8.1 Installing the OHW Artifacts

The Oracle WebLogic Server and other servlet containers allow OHW modules to be compressed as WAR (Web ARchive) files, which are then deployed as an EAR (Enterprise ARchive) file, which wraps any WAR and JAR (Java ARchive) files and the OHW installable files. One way to do this is to create WAR or EAR files using the standard Java JAR utility.

Then the OHW WAR or EAR file must be extracted by the application server so that the Web client can access the OHW pages. You may consult the relevant application server guidelines on how to deploy WAR or EAR files.

Another way is to manually create the Web application using a web developer studio like Oracle JDeveloper Studio, include the `ohw-rcf.jar`, `ohw-share.jar`, `help-share.jar` in the library path, and extract the `ohw-rcf-webapp.zip` to the `public html` directory.

17.8.2 Configuring OHW as Standalone Web Application

After all files have been put in the right locations, the OHW administrator still must modify some configuration files to make OHW work:

- Modify `web.xml` file to include JSF and Trinidad parameters.

  For example:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ... >
  <context-param>
    <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
    <param-value>client</param-value>
  </context-param>
  <context-param>
    <param-name>org.apache.myfaces.trinidad.CHECK_FILE_
MODIFICATION</param-name>
    <param-value>false</param-value>
  </context-param>
  ...
  ...
</web-app>
```

- Modify `web.xml` to support OHW front servlets and JSF filter.

  For example:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ... >
  ...
  ...
  <filter>
    <filter-name>trinidad</filter-name>
    <filter-class>
        org.apache.myfaces.trinidad.webapp.TrinidadFilter
    </filter-class>
  </filter>
  <filter-mapping>
    <filter-name>trinidad</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>REQUEST</dispatcher>
  </filter-mapping>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>OHW Servlet 1</servlet-name>
    <servlet-class>oracle.help.web.rich.OHWServlet</servlet-class>
    <init-param>
      <param-name>ohwConfigFileURL</param-name>
      <param-value>/helpsets/ohwconfig.xml</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>OHW Servlet 1</servlet-name>
    <url-pattern>/ohguide/*</url-pattern>
  </servlet-mapping>
  ...
  ...
</web-app>
```

- If you have not created ohwconfig.xml and `helpsets` directory, create the file and directory as described in steps 1, 2, and 3 of Section 17.6, "Configuring OHW to Display Custom Helpsets".

  If created, modify `ohwconfig.xml`, and edit the help content as required. It specifies which helpsets to display and how to present them. You can also specify locales, branding information, and various other settings. The name and location of this file is set as the `ohwConfigFileURL` servlet initialization parameter, which is handled differently for each servlet container. The ohwConfigFileURL parameter is defined in `web.xml` to specify `param-value`.

  For information about this configuration file, see Chapter 8, "Oracle Help for the Web Configuration File".

If you want to provide the help content outside of the application's EAR file, you must configure the `web.xml` file. In the `<param-value>` element, you can use a variable to define the path of `ohwconfig.xml` using the following syntax:

```
<init-param>
```

```
  <param-name>ohwConfigFileURL</param-name>
  <param-value>
    file:///{%yourVariableName}/help/ohwconfig.xml
  </param-value>
</init-param>
```

When OHW finds a variable (for example, `{%yourVariableName}`) in the path, it looks for the Java system property of the same name (`yourVariableName`), and then replaces the value of the variable with the value defined in Java system property. You can define Java system property in your Oracle WebLogic Server startup scripts.

Support for Ctrl+N Shortcut to Open a New Help Window

You can configure `web.xml` to open a new browser window when a users Ctrl+N shortcut. Add the following code in `web.xml` to enable the shortcut support:

```
<context-param>
  <param-name>oracle.adf.view.rich.newWindowDetect.OPTIONS</param-name>
  <param-value>on</param-value>
</context-param>
```

Support for Partial Page Navigation

To improve performance, you can enable partial page navigation support in OHW. By default, the support is disabled in ADF Faces application, but you can enable it in your helpset by adding the following code in `web.xml`:

```
<context-param>
  <param-name>oracle.adf.view.rich.pprNavigation.OPTIONS</param-name>
  <param-value>onWithForcePPR</param-value>
</context-param>
```

17.9 Deploying OHW as part of a Web Application

One way to deploy the OHW is to make it co-exist with your Web application. The Web application could be a JSF, ADF, or JSP application or any Java EE Web application. OHW then could be deployed as one of the Web projects within the existing application.

When you deploy OHW as part of an existing web application, the web application and OHW help system, both, share the same `web.xml`. This could limit the fine tuning of OHW help system and may cause conflict with your application. It is recommended that you deploy OHW separately from your web application, and then link the help system with your application. For more information, see Section 17.8, "Deploying OHW as a Standalone Web Application". If your application is using ADF Faces, you may use `helpTopicId` attribute on the ADF Faces components for an ADF application. For more information, see Section 19.2, "Integrating Online Help With ADF Faces Application".

17.9.1 Installing the OHW Artifacts

Extract the `ohw-rcf-webapp.zip` to the `public_html` folder (or the web application root directory) of the existing Web application.

Copy the `ohw-rcf.jar`, `ohw-share.jar`, `help-share.jar` files to the application's `WEB-INF/lib` folder, or to the defined library folder. If you are developing in JDeveloper, remember to add these jars to your project (**Project Properties** > **Libraries and Classpath**).

17.9.2 Configuring OHW as Part of Web Application

After all files have been put in the right locations, the OHW administrator must modify some configuration files to make OHW work:

- Modify `web.xml` file to include JSF and Trinidad parameters if it does not exist.

  For example:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ... >
  <context-param>
    <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
    <param-value>client</param-value>
  </context-param>
  <context-param>
    <param-name>org.apache.myfaces.trinidad.CHECK_FILE_
MODIFICATION</param-name>
    <param-value>true</param-value>
  </context-param>
  ...
  ...
</web-app>
```

- Modify `web.xml` to support OHW front servlets and JSF filter.

  Since the OHW is part of existing application, the OHW administrator must ensure that the `load-on-startup` ordering is maintained in the right sequence.

  For example:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ... >
  ...
  ...
  <filter>
    <filter-name>trinidad</filter-name>
    <filter-class>
        org.apache.myfaces.trinidad.webapp.TrinidadFilter
    </filter-class>
  </filter>
  <filter-mapping>
    <filter-name>trinidad</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>REQUEST</dispatcher>
  </filter-mapping>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>product1</servlet-name>
    <servlet-class>oracle.help.web.rich.OHWServlet</servlet-class>
    <init-param>
      <param-name>ohwConfigFileURL</param-name>
      <param-value>/helpsets/product1/ohwconfig.xml</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
  </servlet>
  ...
  ...
```

```
</web-app>
```

- Modify `ohwconfig.xml`, and edit the help content as needed. If you have not created helpsets or `ohwconfig.xml` file, create them as described in steps 1, 2, and 3.

  The `ohwconfig.xml` file specifies which helpsets to display and how to present them. You can also specify locales, branding information, and various other settings. The name and location of this file is set as the `ohwConfigFileURL` servlet initialization parameter (defined in `web.xml`), which is handled differently for each servlet container. For information about this configuration file, see Chapter 8, "Oracle Help for the Web Configuration File".

17.10 Deploying Multiple Help Instances in a Web Application

OHW supports the deployment of multiple help instances (a single help instance may contain multiple helpsets) in a single Web application or enterprise application. One of the main reasons for providing this support is to minimize the changes needed when upgrading from a OHW configuration. The deployment of multiple help instances for OHW is achieved by providing an OHW front servlet that forwards the request to the JSF servlet.

17.10.1 Application and OHW Configuration Files and Setup

You must modify the `web.xml` file of your application to add servlet mapping to the OHW front servlets.

Here is an example of the changes that must be done to the `web.xml` file, to support the deployment of multiple help instances for OHW:

```
<!-- configuration for product1 help front servlet -->
<servlet>
  <servlet-name>product1</servlet-name>
  <servlet-class>oracle.help.web.rich.OHWServlet</servlet-class>
  <init-param>
    <param-name>ohwConfigFileURL</param-name>
    <param-value>/helpsets/product1/ohwconfig.xml</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>

<!-- configuration for product2 help front servlet -->
<servlet>
  <servlet-name>product2</servlet-name>
  <servlet-class>oracle.help.web.rich.OHWServlet</servlet-class>
  <init-param>
    <param-name>ohwConfigFileURL</param-name>
    <param-value>/helpsets/product2/ohwconfig.xml</param-value>
  </init-param>
  <load-on-startup>3</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>product1</servlet-name>
  <url-pattern>/product1/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>product2</servlet-name>
```

```
  <url-pattern>/product2/*</url-pattern>
</servlet-mapping>
```

In the above sample code, there are two OHW front servlets called `product1` and `product2`. Each servlet can load a different OHW configuration file that determines the set of books and views shown in the user interface. The `product1` servlet is mapped to URL pattern of `/product1/*`. So, if you specified a URL mapped to the rich OHW context root that has `product1` at the end portion of the URL, it is re-routed to this servlet. Similarly, the `product2` is mapped to the URL pattern of `/product2/*`.

17.10.2 Running the Application

Once you have successfully deployed OHW in the application server, you can connect to any OHW front servlet you have configured, using a URL similar to the following:

```
http://www.myhelpserver.com:<port>/docs/product1
```

The above URL calls the servlet `product1` that loads the OHW configuration file from `/helpsets/product1/ohwconfig.xml` and is redirected to a URL like this:

```
http://www.myhelpserver.com:<port>/docs/faces/helppages/main.jspx?config=product1
```

OHW can also process locale and group information appended to the URL (similar to OHW).

18

Implementing Context-Sensitive Help in a Web Application

This chapter describes how to enable context-sensitive help in your web application and how to map topic IDs with the help topics.

This chapter includes the following sections:

18.1 About Implementing Context-Sensitive Help In a Web Application

Oracle Help for the Web (OHW) provides a context-sensitive help mechanism that launches help topics that are associated with some context in the Web application user interface. Typically, help topics are written to describe the function of a particular page, table, or input field in a Web application. When a user requests help for a user interface control—for example, by clicking a Help button—the appropriate topic for that context (or control) is displayed.

To provide context-sensitive help for a Web application, the help system must include one or more map files, and the appropriate help code must be added to the application code.

18.2 Mapping Topic IDs to Help Topics

OHW context-sensitive help systems rely on one or more map files that map topic IDs to help topic HTML files. In a helpset, the map file is saved in XML file format as `map.xml`.

The map file is usually created by the help author. As a Web application developer, when associating Web application controls with context-sensitive topics you must use the topic IDs specified in the author's map file. Thus, you will have to coordinate your efforts with the help author.

Here is a sample map file in XML format:

```
<?xml version='1.0' ?>
<map version="1.0">
  <mapID target="topic_1" url="file_1.html" />
  <mapID target="topic_2" url="file_2.html#a1" />
  <mapID target="topic_3" url="file_3.html" wintype="intro" />
</map>
```

The `target` attribute specifies a unique ID for the associated HTML file within a helpset. The `url` attribute specifies the location of the file to associate with the ID. The `wintype` attribute is optional; it specifies the name of a window type that the topic will be displayed in. For more information about the elements used in the map file, see Section 5.3, "Map Files".

18.3 Creating Context-Sensitive Links to the Help System

Applications that rely on OHW for context-sensitive help request the context-sensitive topics via specially formulated URLs to the OHW servlet. Any user interface control with a URL destination (links, images, etc.) can be associated with a context-sensitive topic.

When creating a link to OHW for context-sensitive help, you can either use the URL destination for the front main page, which is a tripane-layout UI with the **Contents**, **Index**, and **Search** navigators on the left side, or you can create a URL destination for displaying a topic in tripane-layout UI using the topic ID. You can also specify a locale and a group in the URL destination.

- Section 18.3.1, "Linking to the Front Main Page"
- Section 18.3.2, "Linking to a Topic"
- Section 18.3.3, "Specifying the Locale and Group"

18.3.1 Linking to the Front Main Page

The URL to the front main page is simply the URL to the OHW servlet:

```
http://<server>:<port>/<servlet mapping>
```

where, `<server>` is the name of your server running the servlet container, `<port>` is the port used by the servlet container, and `<servlet mapping>` is the servlet mapping set up in the `web.xml` file for the OHW servlet (`oracle.help.web.rich.OHWServlet`).

For example, in the `web.xml`, it has the following servlet definition and servlet mapping:

```
<!-- configuration for product1 help front servlet -->
<servlet>
   <servlet-name>product1</servlet-name>
   <servlet-class>oracle.help.web.rich.OHWServlet</servlet-class>
   <init-param>
      <param-name>ohwConfigFileURL</param-name>
      <param-value>/helpsets/product1/ohwconfig.xml</param-value>
   </init-param>
<load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
   <servlet-name>product1</servlet-name>
   <url-pattern>/product1/*</url-pattern>
</servlet-mapping>
```

In this example, the URL to the front main page is:

```
http://www.myhelpserver.com:8888/docs/product1/
```

When a user requests help for a user interface control that is linked to the front main page, the OHW tripane-layout main page will be displayed in the user's browser, showing the **Contents**, **Index**, and **Search** navigators on the left side.

18.3.2 Linking to a Topic

To create the URL for linking to a topic, add a topic parameter to the URL of the OHW servlet. The value of the topic parameter is the topic ID of the help topic:

```
http://<server>:<port>/<servlet mapping>/?topic=<topic-id>
```

For example, the following URL requests the topic associated with the topic ID `topic_1`:

```
http://www.myhelpserver.com:8888/docs/product1/?topic=topic_1
```

When implementing context-sensitive links to OHW, you may also wish to use JavaScript to open the link in a secondary window rather than replace the main application page.

When a user requests help for a user interface control that is linked to a topic ID, OHW displays the tripane-layout UI with the topic file shown in the Topic Navigator (the right side), and Contents Navigator is shown in the left side with the topic highlighted in the TOC tree.

18.3.3 Specifying the Locale and Group

When you link to any OHW page, including topic pages or front pages, you can include a locale and a group in the URL of the OHW servlet with the locale and group query parameter.

The topic syntax is:

```
http://<server>:<port>/<servlet-mapping>/?topic=<topic-id>&locale=<ISO-code>
http://<server>:<port>/<servlet-mapping>/?topic=<topic-id>&locale=<ISO-code>&group
=<aGroup>
```

and the front page syntax is:

```
http://<server>:<port>/<servlet-mapping>/?locale=<ISO-code>
http://<server>:<port>/<servlet-mapping>/?locale=<ISO-code>&group=<aGroup>
```

If you specify the locale, OHW switches to the localized helpset if it is available, and will keep using the specified locale until it is overridden or removed. If the specified localized helpset is not available, the parameter is ignored.

For example:

```
http://www.myhelpserver.com:7101/docs/product1/?topic=topic_1&locale=sp
```

For more information about locale and group, see Chapter 8, "Oracle Help for the Web Configuration File".

19

ADF Rich Client Help Provider

This chapter describes how to integrate online help with the ADF Faces application and how to register OHW as a help provider.

This chapter contains the following sections:

19.1 About ADF Rich Client Help Provider

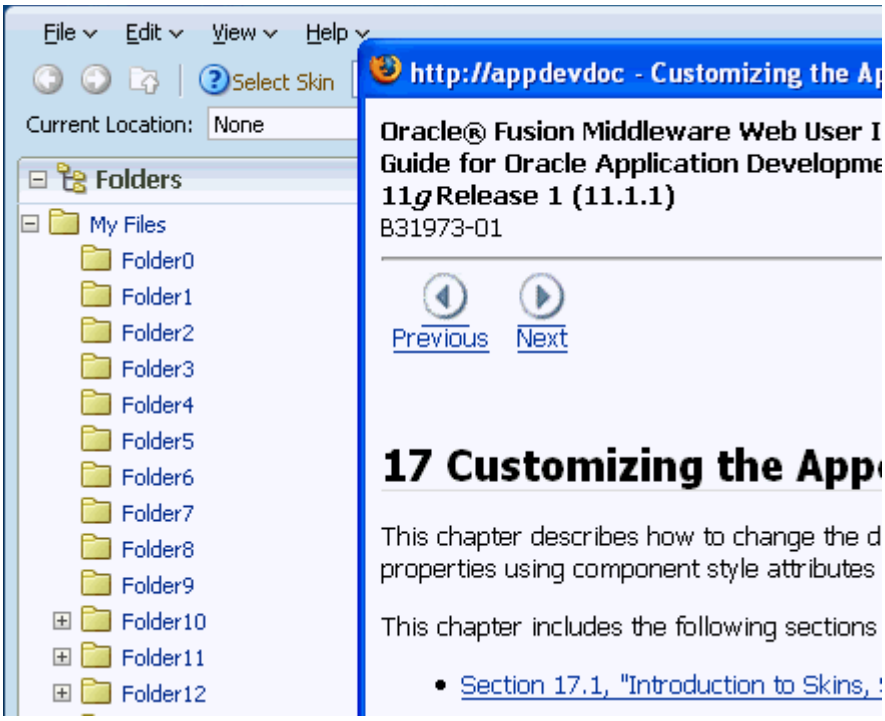
If you have an ADF Faces application and wish to incorporate online help into your application, ADF Faces, also known as ADF Rich Client, provides an easy way to do that. You can create an online help system with help topics which are integrated with components of the application.

This chapter describes how to configure OHW as a help provider, lists other ADF Rich Client help providers, and also describes how to use **HelpTopicId** attribute to associate help topics with your application.

19.2 Integrating Online Help With ADF Faces Application

When you integrate a help topic with an ADF Faces component, a help icon (a blue circle with a question mark) appears with the component. When you click the help icon, the related help topic appears in a new window, as shown in Figure 19–1.

Figure 19–1 Help Icon in an ADF Faces Application



Integrating an online help with an application is an easy process, described in the following steps:

- Register a help provider with your application
- Create a properties file that contains the topic ID and help text for each help topic
- Associate the UI components with help topics by using the **HelpTopicId** attribute

For more information about integrating online help with an application, see "Displaying Help for Components" section in *Developing Web User Interfaces with Oracle ADF Faces*.

19.3 Registering OHW as an ADF Rich Client Help Provider

ADF Rich Client Faces includes a variety of help providers. You can use a combination of the different help providers or create your own help provider class. You can also use OHW as a help provider.

You need to perform the following tasks in order to set up OHW as a Help Provider in an ADF Rich Client application:

1. Deploy OHW as a web application:
   a. Deploy an ADF application to web application server.
   b. Note down the context-root of this deployment.
   c. Open the `web.xml` file in this deployment.
   d. Find the OHW servlet instance.
2. Develop the jspx Web pages.
   a. Create an ADF Faces-based Web application.

b. Create a jspx Web page; for some components (input*, select*, etc), where you can find an attribute named `helpTopicId`, specify the ID you want to display with your OHW instance.

3. Copy the helpsets into a directory under your `<application_root>/public_html` folder. For example, let's name this directory `helpsets`.

4. Set up the `adf-settings.xml` file.

a. Navigate to your application's directory, and look for the `.adf/META-INF/` directory. Under that directory you can find the `adf-settings.xml` file. If the `.adf/META-INF` directory is not present, create the `META-INF` directory under the `ViewController/src` directory.

b. Enter code into `adf-settings.xml`; for example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<adf-settings xmlns="http://xmlns.oracle.com/adf/settings">
  <adf-faces-config xmlns="http://xmlns.oracle.com/adf/faces/settings">
    <help-provider>
      <help-provider-class>
            oracle.help.web.rich.helpProvider.OHWHelpProvider
      </help-provider-class>
      <property>
        <property-name>ohwConfigFileURL</property-name>
        <value>/helpsets/ohwconfig.xml</value>
      </property>
      <!--property>
      <property-name>group</property-name>
      <value>null</value>
    </property-->
      <property>
        <property-name>baseURI</property-name>
        <value>
            http://localhost:8989/help-ohw-rcf-context-root/ohguide/
        </value>
      </property>
    </help-provider>
  </adf-faces-config>
</adf-settings>
```

In this `adf-settings.xml` file:

* If you want to use OHW as your help provider, then, in this `adf-settings.xml` file, the class has to be `OHWHelpProvider`.
* Set the `ohwConfigFileURL` property to point to your `/helpsets/ohwconfig.xml`. Note that you created the `helpsets` directory in Step 3.

Note: You can also specify `ohwConfigURL` using the `prop` system property. For example, in `web.xml`, the `ohwConfigURL` would be configured as:

```
<param-name>ohwConfigFileURL</param-name>
<param-value>file:///{%prop}/help/ohwconfig.xml</param-value>
```

In `adf-settings.xml`, the `ohwConfigURL` would be configured as:

```
<property>
    <property-name>ohwConfigFileURL</property-name>
    <value>file:///{%prop}/help/ohwconfig.xml</value>
</property>
```

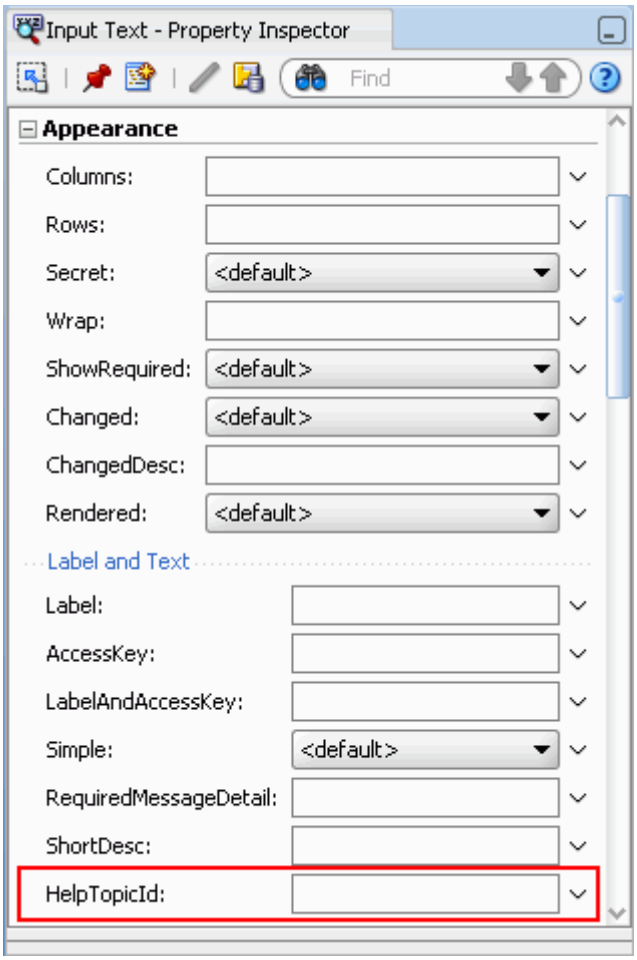
The `prop` property is predefined, or specified, by starting Oracle WebLogic Server with `-Dprop` option. For example, `-Dprop=/Oracle/help`.

* The group property specifies the group name (which you want to use in the help provider) in your `ohwconfig.xml` file.
* The `baseURI` property specifies the server host, the context root, and the OHW servlet instance of the RCF application that you deployed in Step 1.

19.4 Using HelpTopicId Attribute

To associate a help topic with an ADF component, you must assign the help topic's unique id as the **HelpTopicId** attribute's value. The **HelpTopicId** attribute is available in the Appearance section of Properties window.

Figure 19–2 HelpTopicId Attribute



Before associating a help topic with a component, ensure that you have registered a help provider with your application. To register OHW as a help provider, see Section 19.3, "Registering OHW as an ADF Rich Client Help Provider".

For more information about **HelpTopicId** attribute, see "Displaying Help for Components" section in *Developing Web User Interfaces with Oracle ADF Faces*.

19.5 Using Other Help Providers

Two common ADF Rich Client help providers are `ResourceBundleHelpProvider` and `ELHelpProvider`. The `ResourceBundleHelpProvider` help provider allows you to display help in your ADF Faces application that is defined in resource bundles. These resource bundles are containers of your help files, control files, graphics, and other related files. The `ELHelpProvider` help provider allows you to display help text in your ADF Faces application that is defined in the XLIFF files. The XLIFF files get converted into maps, or create a managed bean that contains a map of help text strings. Note that `ELHelpProvider` does not create help files, but helps you connect those files to the ADF Faces application.

For more information about these and other help providers, see "Displaying Help for Components" section in *Developing Web User Interfaces with Oracle ADF Faces*.

A

Oracle Help and JavaHelp File Formats

This appendix describes the Oracle Help file formats based on the JavaHelp specification. Oracle Help extends the JavaHelp file definitions to support additional features.

This appendix includes the following sections:

These conventions apply to the following tables:

- In the **Element** column, child elements are indented under parent elements.
- The items in the **Attribute** column are attributes of the element listed directly above the attribute(s).
- In the **Origin** column, "JH" refers to the JavaHelp 1.0 file format specification. "OH" refers to Oracle Help. Oracle's extensions to JavaHelp were originally developed for Oracle Help for Java (OHJ). However, most of the formats now apply to both OHJ and Oracle Help for the Web (OHW). Therefore, "OH" alone is used.
- In the **Supported By** column, "OH" refers both to OHJ and OHW.

A.1 Helpset File

For more information about this file, see Section 5.2, "Helpset File".

Table A–1 Helpset Elements

Element	Attribute	Origin	Supported By
`<helpset>`	-	JH	OH & JH - Differences as shown below
-	`xml:lang`	JH	JH only
-	`version`	JH	JH only
`<title>`	-	JH	OH & JH

Table A–1 (Cont.) Helpset Elements

Element	Attribute	Origin	Supported By
<maps>	-	JH	OH & JH
<homeID>	-	JH	JH & OH (in some circumstances)
<mapref>	-	JH	OH & JH - Differences as shown below
-	location	JH	OH & JH
-	class	OH	OH only
<wintype>	-	OH	OH only
-	default	OH	OH only
<name>	-	OH	OH only
<height>	-	OH	OH only
<width>	-	OH	OH only
<x>	-	OH	OH only
<y>	-	OH	OH only
<textfg>	-	OH	OH only
<linkfg>	-	OH	OH only
<bg>	-	OH	OH only
<title>	-	OH	OH only
<toolbar>	-	OH	OH only
<links>	-	OH	OH only
<linkref>	-	OH	OH only
-	location	OH	OH only
<view>	-	JH	OH & JH - Differences as shown below
<name>	-	JH	JH only
<label>	-	JH	OH & JH
-	image	OH	OH only
<title>	-	OH	OH only
-	image	OH	OH only (TOC view only)
<type>	-	JH	OH & JH
<data>	-	JH	OH & JH - Differences as shown below
-	class	OH	OH only
-	engine	JH	OH & JH
<subhelpset>	-	OH	OH only
-	location	OH	OH only
-	class	OH	OH only

A.2 Map File

For more information about this file, see Section 5.3, "Map Files".

Table A–2 Map File Elements

Element	Attribute	Origin	Supported By
<map>	-	JH	OH & JH - Differences as shown below
-	xml:lang	JH	JH only
-	version	JH	JH only
<mapID>	-	JH	Differences as listed below
-	target	JH	OH & JH
-	url	JH	OH & JH
-	wintype	OH	OH only
-	xml:lang	JH	JH only

A.3 Table of Contents File

For more information about this file, see Section 6.2, "Table of Contents File".

Table A–3 Table of Contents Elements

Element	Attribute	Origin	Supported By
<toc>	-	JH	OH & JH - Differences as shown below
-	xml:lang	JH	JH only
-	version	JH	JH only
<tocitem>	-	JH	Differences as listed below
-	target	JH	OH & JH
-	text	JH	OH & JH
-	image	JH	OH & JH
-	xml:lang	JH	JH only

A.4 Index File

For more information about this file, see Section 6.4, "Index File".

Table A–4 Index File Elements

Element	Attribute	Origin	Supported By
<index>	-	JH	OH & JH - Differences as shown below
-	xml:lang	JH	JH only
-	version	JH	JH only
<indexitem>	-	JH	OH & JH - Differences as shown below
-	target	JH	OH & JH

Table A–4 (Cont.) Index File Elements

Element	Attribute	Origin	Supported By
-	`text`	JH	OH & JH
`<indexentry>`	-	OH	OH only
-	`target`	OH	OH only
-	`text`	OH	OH only

A.5 Search Index File

This file is unique to Oracle Help. For more information, see Section 6.5, "Search Index File".

A.6 Link File

This file is unique to Oracle Help. For more information, see Section 6.6, "Link File".

A.7 OHW Configuration File

This file is unique to Oracle Help. For more information, see Chapter 8, "Oracle Help for the Web Configuration File".

B

Working Around the Java Modal Window Problem

This appendix describes how Java handles modal windows that causes a problem when trying to display a context-sensitive help topic for a modal window. A modal window is one that does not allow focus to be shifted away from it. A nonmodal window is one that allows focus to be switched to another window.

This appendix includes the following sections:

B.1 About the Java Modal Window Problem

If a user requests help from a nonmodal window, it is possible to switch back and forth between the help window and the window requesting help. However, this is not possible when requesting help from a modal window. In Java, a modal window blocks access to all other windows created by the Java Virtual Machine, except yet another modal window. Thus, if help is requested from a modal window, OHJ must display help in a modal help window. Then, because OHJ is itself shown in a modal window, the user must close the help window to return to the application.

When help is requested, OHJ determines whether the active window is modal. If it is, then it re-parents the normal OHJ topic windows and the OHJ navigator window into a new modal window. That new window appears in the foreground of the user's display, and the user can interact with it; in fact, they must interact with it if only to close the modal help window. Given the coarse implementation of modality in Java, this is the only solution that will work for all of the Java Virtual Machines currently supported by OHJ.

B.2 Registering a Window

In order for the OHJ workaround to work, OHJ must be able to track the currently active window. Use the `registerClientWindow()` method to register each window (Frame or Dialog) you create with the Help object.

Table B–1 registerClientWindow() Method

Constructor	Description
`registerClientWindow(Window aWindow)`	Window instances registered with the Help object are tracked. If the active window is a modal dialog and help is requested, the Help object will take special action so that the help windows are not blocked by the active modal dialog. Parameters: ■ `aWindow` - The Window instance to register.

B.3 Unregistering a Window

If you registered your Window objects using `Help.registerClientWindow()`, you must also unregister them. When you know that a Window will no longer be active, you should unregister the window with the Help object using the `unregisterClientWindow()` method. It is important to note that failure to unregister Window instances may result in the window not being garbage collected.

Table B–2 unregisterClientWindow() Method

Method	Description
`unregisterClientWindow(Window aWindow)`	Clients should unregister each Window instance they registered with the `registerClientWindow()` method once the window will no longer be active. Failure to unregister Window instances may result in the window not being garbage collected. Parameters: ■ `aWindow` - The Window instance to register.

C

Working With HelpBooks

This appendix describes how older versions of OHJ used the *helpbook* file and the `HelpBook` object. Helpbooks are still supported in current versions of OHJ; however, it is preferable in current versions to use helpsets (helpset file and `HelpSet` object).

If you are still using helpbook file and object, and do not wish to use helpsets, the following sections provide helpbook specific information.

This appendix includes the following sections:

C.1 HelpBook File Name Extensions

If you are using helpbooks, you must ensure that file name and extensions are correct. OHJ and OHW look in a specified directory for files with file name extensions that correspond to the supported file formats, including TOC, TOK, HHC, HHK, OHT, and IDX. For more information about correct file name and extensions, see Section 4.2, "File Name Extensions"

When you use helpsets, you don't have to use specific extensions for the names of the associated control files.

C.2 Adding the Help Data in OHJ

If you want to add a helpbook in OHJ, you must first create a `Help` object. For more information, see Section 13.2, "Constructing the Help Object".

> **Note:** When an OHJ system is implemented using a HelpBook, the map file must use the OHT file format. HelpBooks and OHT files are legacies from early versions of OHJ and are no longer recommended, but they are still supported by OHJ. For more information about the map file, see Section 5.3, "Map Files".

After creating a `Help` object, you must add one or more `Book` objects to it. A `Book` object encapsulates a collection, or a book of help content.

This `HelpBook` book implementation handles legacy OHJ file formats. The HelpBook class examines a directory, identifies files with known extensions, and adds them to the help system.

The following sections describe how to add the helpbooks, and other optional features:

- Section C.2.1, "Constructing a HelpBook"
- Section C.2.2, "Adding Books to Help"

For more information, see the API documentation for `oracle.help.library.helpset.HelpSet`.

C.2.1 Constructing a HelpBook

The `HelpBook` format is directory based. Use its constructors to specify the location of the directory containing the help content (HTML topic files).

Table C–1 HelpBook() Constructors

Constructor	Description
`HelpBook(String baseURL, String baseName, String bookTitle)`	Create a `HelpBook` with the specified title, loading the data from the directory location specified by `baseURL`. The `HelpBook` object examines the directory to identify known control files (TOC, TOK, HHC, HHK, OHT, and IDX) named with the given `baseName`. Use this constructor when you know the exact location of the help content directory. Parameters: ■ `baseURL` - A String specifying the exact location of the helpset file in URL format. ■ `baseName` - The base name of the control files found in the specified directory (for example, if your control files are named userguide.hhc, userguide.hhk, etc., the `baseName` is *`userguide`*). ■ `bookTitle` - The book name that will appear in the book selection pull down menu in multiple book help systems. The title is often supplied by the help author.
`HelpBook(Class baseClass, String dirPathExt, String baseName, String bookTitle)`	Create a `HelpBook` with the specified title, loading the data from the directory location specified by `baseClass` and `dirPathExt`. The `HelpBook` object examines the directory to identify known control files (TOC, TOK, HHC, HHK, OHT, and IDX) named with the given `baseName`. Use this constructor when you know only the path to the help content directory relative to your application implementation. Parameters: ■ `baseClass` - One of your application classes. The `HelpBook` object uses the location of this class to determine the location of your help content directory. ■ `dirPathExt` - The path to the help content directory relative to the location of `baseClass`. The value of this parameter is appended to the absolute path to the directory containing the `baseClass`. The resulting path should be the path to your help content directory. ■ `baseName` - The base name of the control files found in the specified directory (i.e. if your control files are named `discoverer.hhc`, `discoverer.hhk`, etc., `the baseName` would be *`discoverer`*). ■ `bookTitle` - The book name that will appear in the book selection pull down menu in multiple book help systems. The title is often supplied by the help author.

C.2.2 Adding Books to Help

Once you have constructed a `Book` instance using the HelpBook or `HelpSet` constructors, you must add the `Book` to your `Help` instance. This is accomplished by calling the following method on the `Help` instance:

Table C–2 addBook() Constructors

Constructor	Description
`addBook(Book book)`	This method adds a `Book` instance to the help system. Author-defined views contained in the `Book` are displayed in the navigator window, and topics from the `Book` are available to display. Parameters: ■ `book` - The `Book` instance to add to the help system.

C.3 Locales in Oracle Help for the Web Configuration File

While configuring Oracle Help for the Web configuration file (`ohwconfig.xml`) for locale, you can specify a single locale or multiple locales in the `<locales>` section.

For more information on configuration file, see Chapter 8, "Oracle Help for the Web Configuration File".

C.3.1 The <locale> Child Element <books>

The `<books>` element specifies the content to be displayed in Oracle Help for the Web. The `<books>` element can contain any number of helpbooks, or a combination of helpsets and helpbooks. Helpsets and helpbooks are also called as books.

Table C–3 describes the `<helpbook>` child element:

Table C–3 <books> Child Elements

Element	Description
`<helpBook>`	The helpbook to include in this instance of OHW. This element has the following attributes: ■ `id` – Unique id to this book. This attribute is required. ■ `basename` – The name, without the file name extension, of the control files in the directory. For example, if the control files are named `myproject.hhc`, `myproject.hhk`, `myproject.idx`, and so on, the basename is `myproject`. ■ `location` – The location of the helpbook file. The path can be either absolute or relative to the location of the OHW configuration file. ■ `title` – The title to use when if helpbooks or helpsets are merged. The helpbook is displayed in the merged book list in the OHW user interface. By contrast, this title is set in the `<title>` element in the helpset file for a helpset. ■ `controlFileencoding` – A Java-supported encoding name. The set of supported encodings varies with JDK version. For a list of supported encodings for Java SE, see `http://download.oracle.com/javase/1.5.0/docs/guide/intl/encoding.doc.html`.

For example:

```
<books>
```

```
 <helpBook id="disco" baseName="discoverer" location="discoverer/"
title="Discoverer Help" />
</books>
```

The <helpbook> elements can contain zero or more <contentLocation> elements for situation when help topic files are located in locations other than those expected by Oracle Help.

For more information about other child elements of <books>, see Section 8–3, " <books> Child Elements".

C.3.2 The <contentLocation> Element

By default OHW automatically processes help topic files that are located in the same locations as helpbook base directories.

In helpbooks, OHW processes help topic HTML files:

- in the helpbook base directory and subdirectories under that location
- or if the helpbook is in a JAR file, all help topic files in the same JAR

If you have help topic files in some other location, you must use the <contentLocation> element to point to that location.

The <contentLocation> element has the attribute baseURI. It represents a URI to the root location of a set of help content, using a path that is either absolute or relative.

The <contentLocation> element can be a child of <helpbook>. A <helpbook> can contain zero or more <contentLocation> elements.

This element is needed because, unlike plain HTML files, Oracle Help help topic files must be processed by the servlet in order to be displayed. Therefore, it is necessary to explicitly list the locations where help topic files referenced in the helpbook reside, if they are not in the default locations. This can happen if your helpset includes a subhelpset in another location or even on another web server or if your context sensitive map file contains references to help topic files located elsewhere on the same server or on a different server.

For more information, see Section 8.4.2, "The <contentLocation> Element".

D

Oracle Help for the Web – UIX

This appendix describes Oracle Help for the Web – UIX (OHW-UIX). OHW-UIX is a Java servlet and a file formats specification for developing and delivering HTML-based help content in a web environment.

OHW-UIX can be used to provide context-sensitive help for web applications or as means for processing and displaying structured views of independent HTML content on the web. With OHW-UIX, a user needs only a web browser to navigate and view help content. The processing takes place on the server, through the OHW-UIX servlet. Because the help content is managed on a server and displayed in any number of web browsers, many users have access to a single installation of the help.

This appendix includes the following sections:

D.1 About Oracle Help for the Web – UIX

OHW-UIX is the previous version of Oracle Help for the Web. If you are creating a new help system, it is recommended that you create the help system using Oracle Help for the Web, instead of choosing OHW-UIX. You should use OHW-UIX if you are building applications with Oracle's ADF UIX technology.

OHW-UIX includes the following:

- **The OHW-UIX servlet**: The OHW-UIX servlet is installed on a web server to provide help to multiple users who access the help system through a web browser. Among other tasks, the OHW-UIX servlet does the following:
  - Parses and merges helpsets
  - Processes searches
  - Generates the OHW-UIX user interface and delivers it to users' web browsers
  - Delivers the help content for display through OHW-UIX in users' web browsers

The OHW-UIX user interface includes all features available in OHJ's Java user interface, but they are rendered as HTML in users' browsers. Features include a table of contents, index, and text search.

The help content files and control files (the same HTML and XML files that are used in OHJ) can be stored on the same server as the servlet or can be spread out over multiple servers in different locations.

- **Documentation**: Documentation includes this Guide.

D.2 OHW-UIX User Interface

The Oracle Help for the Web – UIX (OHW-UIX) user interface provides the same features as that of Oracle Help for Java (OHJ). However, since OHW-UIX is a web application, there are some differences in appearance and behavior.

OHW-UIX help system is recommended if you are building applications with Oracle ADF UIX technology. If you are not using ADF UIX technology, you must use Oracle Help for the Web – Rich Client help system. For more information, see Chapter 3, "Oracle Help for the Web User Interface".

Figure D–1 OHW-UIX User Interface

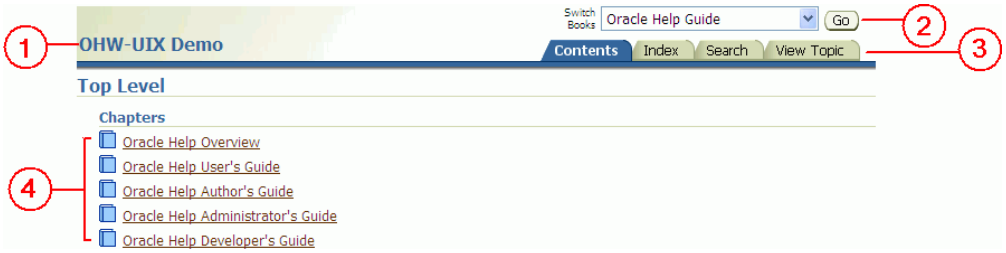


Figure D–1 numbered callouts identify the following user interface components:

1. **Branding area**: This area can contain text, an image, or both. Typically, this area identifies the help content or provides company information, such as the name and logo.
2. **Helpset switcher**: If you have more than one helpset, you can specify whether to merge them or to enlist them separately in this dropdown.
3. **Tab bar**: The default tabs are **Contents**, **Index**, **Search**, and **View Topic**.
4. **Content area**: When a tab is selected, the content associated with that tab appears in this area.

With the exception of the branding area, these elements are configured in the helpset file. OHW-UIX and OHJ use the exact same file formats, including the helpset file. That means that you can take an existing OHJ help system and deploy it as an OHW-UIX system, without changing any of your existing control files. OHW-UIX uses the same directives from the helpset file to construct its user interface as are used by OHJ to configure its user interface. To deploy a help system as an OHW-UIX system, you must configure and deploy a servlet container, and you must add an OHW-UIX configuration file. The branding information is specified in this file, among other configuration parameters.

For more information about deploying OHW-UIX system, see Section D.4, "Understanding OHW-UIX Deployment".

Comparing OHW-UIX Tabs with OHJ Tabs

The standard tabs (also called Navigators) in OHW-UIX are **Contents**, **Index**, **Search**, and **View Topic**. The **Contents**, **Index**, and **Search** tabs correspond to the same tabs that appear in an OHJ navigation window for a helpset. The **View Topic** tab takes the place of the topic window in OHJ.The OHW-UIX navigators remember their state for the current user. That means that if you switch from one tab to another, or follow a series of links from a topic, the previously visited tabs retain their contents. For example, if you perform a search in the Search tab and then follow several links from one of the topics found in your search, when you return to the **Search** tab, your most recent search criteria and results are displayed. This is not a surprising feature for an application that resides on a local system (such as OHJ), but it is an important feature for a web application, where the application runs on a server and can be accessed by many remote users at the same time.The following topics describe the OHW-UIX user interface elements in more detail, including comparisons to OHJ:

- OHW-UIX Table of Contents
- OHW-UIX Index
- OHW-UIX Search
- OHW-UIX Topics

D.2.1 OHW-UIX Table of Contents

In contrast to OHJ, which shows the Table of Contents (TOC) as a tree, the OHW-UIX TOC displays the hierarchy of a help system as a sequence of pages. Each page shows one node in the hierarchy, and that page can list both, topics and child nodes (that is, nodes that fall under the current node). When you select a topic title, OHW-UIX switches to the View Topic tab, where the contents of that topic are displayed. When you click the title of a node in the **Contents** tab, the page is refreshed to show the listing for that node. When you navigate through a hierarchy, the navigation trail, or breadcrumbs, is shown as a set of links at the top of the page. This provides a quick way to navigate back to a previous level in the hierarchy.

Figure D–2 Table of Contents View in OHW-UIX

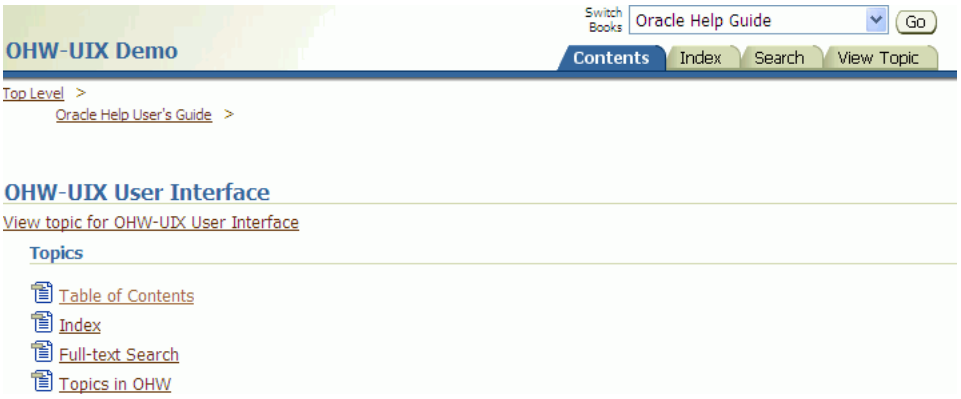


Figure D–3 Navigation Links

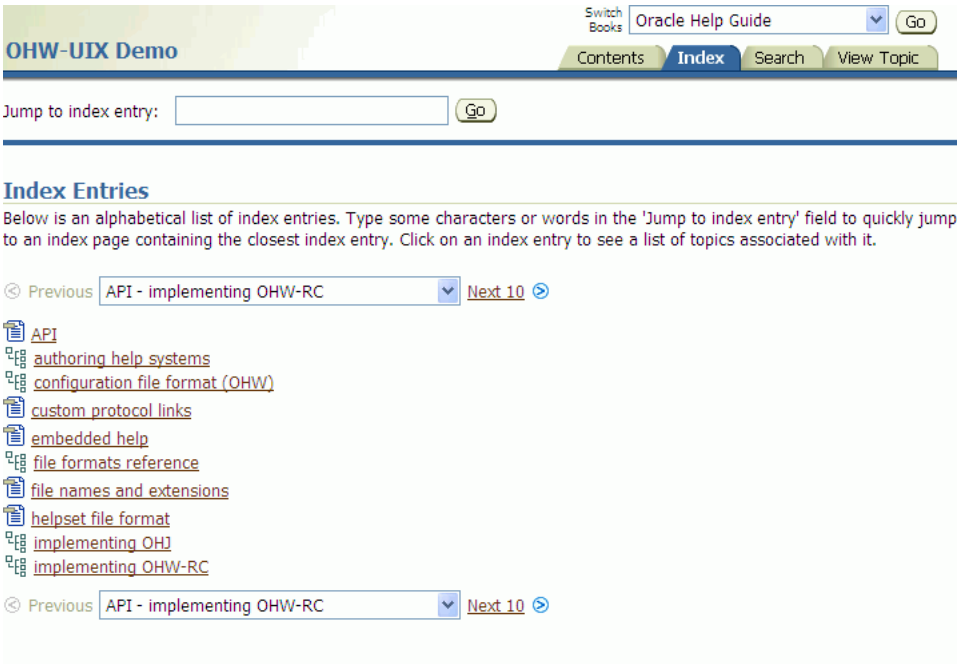
D.2.2 OHW-UIX Index

The OHW-UIX keyword index is also slightly different from the one in OHJ. As in OHJ, the OHW-UIX index has a text field labelled **Go to** where you can enter words you want to find in the index. In contrast to OHJ, you must select **Go** or press the Enter key before the associated words are displayed. OHW-UIX displays only 10 items at a time, but it also has controls for navigating through the entire list of items that match what was entered in the **Go To** field. If there is no entry in the **Go To** field, you can navigate through the entire keyword list.

When you select an item from the list of keywords, the page is refreshed with a list of topics associated with the selected keyword. This is equivalent to the list of found topics at the bottom of the OHJ Index tab. When you select an item from the list, OHW-UIX switches to the **View Topic** tab, where the topic is displayed.

The **Topics for** page in the index has navigation links at the top of the page, similar to the one described for the table of contents. However, this link always takes you back to the list of keywords.

Figure D–4 Index Views in OHW-UIX



D.2.3 OHW-UIX Search

The OHW-UIX text search is similar to the index. You enter a word or phrase in the **Search** text field, press Enter, and OHW-UIX displays a list of associated topics. The first ten items are displayed (the list is sorted by rank), and you can navigate through further items in the list. In Advanced Search mode, you can also specify options for your search, like case sensitivity, match all words, match any words, or use a boolean expression.

As in the index, when you select an item from the list of topics found, the topic is displayed in the **View Topic** tab.

Figure D–5 Search Navigator in OHW-UIX

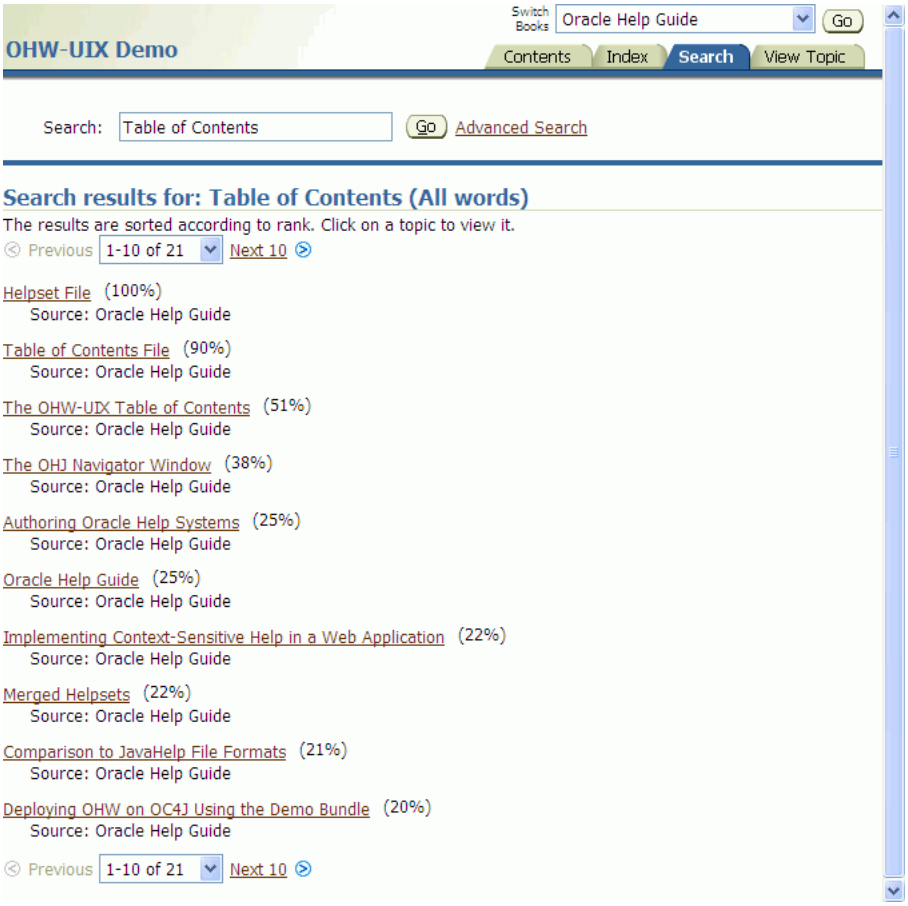
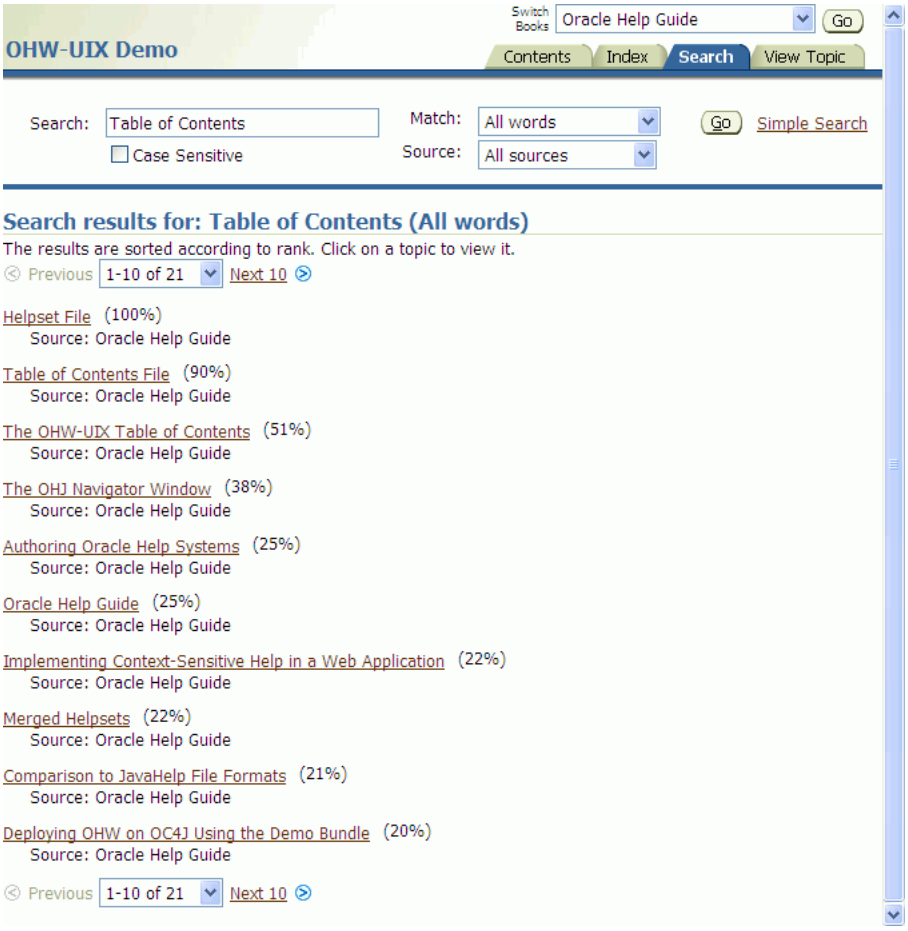


Figure D–6 Advanced Search Navigator in OHW-UIX



D.2.4 OHW-UIX Topics

Topics in OHW-UIX are displayed in the **View Topics** tab. This navigator supports several special features from OHJ, including topic ID links, associative links, and popups. In OHJ, these features were processed by the ICEbrowser display engine. In OHW-UIX they work in any browser. For more information about these features, see Chapter 7, "Topic Files".

Figure D–7 Content (Topic) Views in OHW-UIX

To locate the topic in Table of Contents, click **Locate in 'Contents'**. To take a print of the topic, click **Printable Page**. A new printable page is created without navigation tabs, help title, and helpset switcher.

D.3 Deploying OHW-UIX Demo File

This chapter describes how to deploy OHW-UIX demo file on Oracle WebLogic Server.

The OHW-UIX demo EAR file contains the class libraries that you must view the demo and try out the release. It includes sample helpsets, OHW-UIX servlet file, and XML configuration files. You can deploy the demo file to experience the OHW-UIX interface, or replace the existing helpsets and add your own.

The following sections help you understand the demo file, deploy and test the sample helpset in your environment, and optionally add your own helpset for testing and deployment:

- Section D.3.1, "Understanding the OHW-UIX Demo File"
- Section D.3.2, "Installing the OHW-UIX Demo EAR File on Oracle WebLogic Server"
- Section D.3.3, "Installing the OHW-UIX Demo EAR File using Oracle JDeveloper"
- Section D.3.4, "Running the OHW-UIX Demo EAR File"

D.3.1 Understanding the OHW-UIX Demo File

The OHW-UIX demo file, `ohw-uix-demo.ear`, contains three OHW-UIX sample helpsets along with their help topics, helpset file, and control files. It also contains `ohwconfig.xml` which is needed to configure OHW-UIX.

The OHW demo EAR file extracts files into the `ohw-uix-demo` directory. The EAR file, when extracted, contains an `ohw-uix-demo.war` file and creates the `META-INF` directory, which contains the `application.xml` file. The `ohw-uix-demo.war` when extracted into `ohw-uix-demo` directory contains the helpsets directories along with their help topics, helpset file, and control files.

Table D–1 describes the files and directories in OHW-UIX demo file.

Table D–1 OHW-UIX Demo Files

Directory Location	Description
`application.xml`	Java EE application: a simple OHW-UIX application. The file is available in `META-INF` directory.
`helpsets`	A web module containing three helpsets: `blafdoc`, `ohguide`, and the sample `shakespeare`, in their respective directories. The helpsets directory exists in the `ohw-uix-demo.war` file.
`cabo`	The UIX installable resource files. The `cabo` directory exists in the `ohw-uix-demo.war` file.
`web.xml`	Contains configuration and deployment information that affects OHW-UIX. The file ise available in `WEB-INF` directory of `ohw-uix-demo.war` file.
`ohwconfig.xml`	OHW-UIX configuration file. The file is available in `helpsets` directory.

D.3.2 Installing the OHW-UIX Demo EAR File on Oracle WebLogic Server

Installing the demo EAR file is a very simple process:

1. Download the OHW-UIX demo EAR file from OTN. The name of the demo file is `ohw-uix-demo.ear`. This file includes OHW-UIX and sample helpsets.
2. Start the Oracle WebLogic Administration Console and navigate to **Deployments** page.
3. In the Deployments page, click **Install** to start the deployment wizard.
4. In the **Path** field, enter the location where you saved the `ohw-uix-demo.ear` file, and click **Next**.
5. In the Choose targeting style page, select **Install this deployment as an application**, and click **Next**.
6. In the Optional Settings page, verify the settings. It is recommended that you leave the settings as default. Click **Next** to continue, or click **Finish** to complete the deployment.
7. In the Additional Configuration page, select **Yes, take me to the deployment's configuration screen** and click **Finish** to complete the deployment.

   The deployment wizard, after successful deployment, returns you to the Settings page of `ohw-uix-demo.ear`. If there are errors while deploying the file, you are navigated to Deployment home page where the errors are listed in red text.

D.3.3 Installing the OHW-UIX Demo EAR File using Oracle JDeveloper

To install the demo file on Oracle JDeveloper, follow these steps:

1. Download the OHW-UIX demo EAR file from OTN. The name of the demo file is `ohw-uix-demo.ear`. This file includes OHW-UIX and sample helpsets.
2. Start JDeveloper.

3. Create a new application from the `ohw-uix-demo.ear` file. From the **File** menu, select **New**. In the New Gallery dialog, select **Applications** under General category, and then select **Application from EAR File** from the Items list.
4. In the Location page of Create Application from EAR File wizard, browse and select the `ohw-uix-demo.ear` file. You may also change the application name and the location of application. Click **Next** to continue.
5. The EAR Modules page of the wizard shows the project name and the module name. Click **Next** to continue, or click **Finish** to create the application from EAR file.
6. The Finish page of the wizard shows a summary of your settings for the application. Click **Finish** to create the application from EAR file. JDeveloper extracts all files from the EAR file and creates an application which is ready to edit and deploy.
7. In the Applications window, open the project and edit desired files.
8. To deploy the application, start the integrated Oracle WebLogic Server instance. From **Run** menu, choose **Start Server Instance** to start the integrated Oracle WebLogic Server.
9. In the Applications window, select the `ohw-uix-demo` project. From the **Application** menu, select **Deploy**. Then from the submenu, select **ohw-uix-demo**, **to**, and then select **IntegratedWLSConnection**. JDeveloper starts the deployment process and the status of the deployment is reflected in the Log window.

   When the application is successfully deployed, JDeveloper prompts with `Deployment finished` message in the Log window.

D.3.4 Running the OHW-UIX Demo EAR File

After successful deployment of `ohw-uix-demo.ear`, open your browser and navigate to the following URL:

```
http://<yourHost>:<yourPort>/ohw-uix-demo/help/
```

For example:

If you have installed Oracle WebLogic Server on your local system, you can open the demo help file with the following URL:

```
http://localhost:7101/ohw-uix-demo/help/
```

The URL automatically changes to `http://localhost:7101/ohw-uix-demo/help/state?navSetId=ohguide&navId=0&destination=` and the help opens with Table of Contents view. For more information on user interface of OHW-UIX, see Section D.2, "OHW-UIX User Interface".

D.4 Understanding OHW-UIX Deployment

Help authors create help content using the authoring tools of their choice. Help authors usually also create the Oracle Help control files that are needed for deploying the help content as OHW-UIX help systems. OHW-UIX administrators typically perform all tasks necessary to deploy a helpset.

Because both help authors and OHW-UIX administrators may want to perform deployments for testing or production, demo deployment files are provided. You can download the latest demo files from OTN.

If you are new to OHW-UIX, you may start with deploying the demo `ohw-uix-demo.ear` file. For more information, see Section D.3.1, "Understanding the OHW-UIX Demo File". The demo EAR file includes the files needed to deploy the sample helpsets immediately.

If you are creating a new OHW-UIX helpset, the following sections help you understand the OHW-UIX deployment process and describe the steps required to create and deploy your own OHW-UIX help system.

Unless configuration files have already been created and the application server configured, the OHW-UIX administrator needs to perform these tasks to deploy an OHW-UIX help system:

- Section D.4.1, "Verifying Requirements and Dependencies"
- Section D.4.2, "Understanding OHW-UIX Configuration Files"
- Section D.4.3, "Configuring OHW-UIX to Display Custom Helpsets"
- Section D.4.4, "Changing the OHW-UIX Access URL"
- Section D.4.5, "Upgrading OHW-UIX and UIX"

D.4.1 Verifying Requirements and Dependencies

Verify all requirements and dependencies before beginning any deployment:

Table D–2 OHW-UIX Deployment Minimum Requirements

Requirement	Description
Servlet Container	OHW-UIX requires a JavaEE 1.5 compatible application server. Oracle WebLogic Server, standalone or integrated with JDeveloper, is recommended as it requires minimal configuration effort.
Client	The client receives only HTML, and all it requires is a web browser to display and view the OHW-UIX help content. The web browser must have JavaScript support enabled. OHW-UIX is supported on Microsoft Internet Explorer 7, Microsoft Internet Explorer 8, Mozilla FireFox 2, Mozilla FireFox 3, Apple Safari, and Google Chrome.
UIX	OHW-UIX is a UIX application.
UNIX Only:X Server	On Unix, the servlet container must be configured to connect to an X server in order for dynamic image generation to succeed.

D.4.2 Understanding OHW-UIX Configuration Files

Before you start deploying the OHW-UIX helpset, there are some files that must be modified to configure OHW-UIX correctly. The following information helps you understand the XML configuration files:

- **`application.xml`**: A manifest of all web modules that run under a given Java EE application. It points to each web module of each product that is deployed. Oracle recommends using two instances of `application.xml`:
  - A relatively stable version for the UIX application (optional).
  - A version for the OHW-UIX application that changes frequently as web modules are added or reconfigured.

  The name and location of `application.xml` is fixed by the Java EE standard. In OHW-UIX, the file must be located `<OHW-UIX_HOME>\META-INF` directory.

- **`web.xml`**: Sets the initialization parameters for the servlet, including the location of the OHW-UIX configuration file. There is one instance of `web.xml` for each web module. If OHW-UIX configuration files are located and named in a uniform manner, then this file should be the same for all OHW-UIX web modules. The file must be located in `<OHW-UIX_HOME>\<OHW-UIX_deployment_name>\WEB-INF\`directory.
- **`ohwconfig.xml`** (default file name): Specify which helpsets to display and how to present them. You can also specify locales, branding information, and various other settings. For information about the configuration file, see Chapter 8, "Oracle Help for the Web Configuration File". The name and location of this file is set as a servlet initialization parameter, which is handled differently for each servlet container. The file must be located in `<OHW-UIX_HOME>\<OHW-UIX_deployment_name>` directory.

D.4.3 Configuring OHW-UIX to Display Custom Helpsets

The instructions in this section helps you create the directory structure required for OHW-UIX help system, add your custom helpset files in the correct location, create or modify the configuration files, and deploy the help system on application server.

The instructions in this section also assume that you have installed the OHW-UIX demo EAR file and you have a knowledge of the demo EAR file's directory structure. If you have not installed the demo file, install it following instructions in Section D.3.1, "Understanding the OHW-UIX Demo File".

Follow these steps to set up OHW-UIX help system:

1. Set up the directory structure as following:

```
<OHW-UIX_HOME>
  |
  – <OHW-UIX_deployment_name>
      |
      – cabo
      – helpsets
          |
          – <custom_helpset_directory>
      – META-INF
      – WEB-INF
          |
          – lib
  – META-INF
```

For example:

```
my_module
  |
  – my_module_help
      |
      – cabo
      – helpsets
          |
          – my_ModuleHelpset
      – META-INF
      – WEB-INF
          |
          – lib
  – META-INF
```

2. Create your own helpset directory. Place all your help files in or under <OHW-UIX_HOME>\<OHW-UIX_deployment_name>\helpsets\<custom_helpset_directory> directory, including the helpset file, topic files, and the other control files (index, table of contents, and so on). Also, place any JAR files here, if you are using JAR files for your helpset. You can use JARred and unJARred helpsets together in the same deployment.
3. Copy the ohwconfig.xml from ohw-uix-demo\ohw-uix-demo\helpsets directory and save it in <OHW-UIX_HOME>\<OHW-UIX_deployment_name>\helpsets directory. Then, update and configure the configuration file:
   a. Modify the <books></books> section to direct it to your helpset. For example:

      ```
      <books>
        <helpSet id="myModule" location="my_ModuleHelpset/my_ModuleHelpset.hs" />
      </books>
      ```

   b. Remove the helpsets which you do not want to provide from the <books></books> section. If removed, the helpsets would not appear in the helpset switcher dropdown list of the OHW-UIX user interface. If you have only one <helpSet> element in the <books></books> section, the helpset switcher is not available.
   c. Update the <brandings></brandings> section to display your own brand. For example:

      ```
      <brandings>  <branding text="My Module" /></brandings>
      ```

4. Copy the cabo files from ohw-uix-demo\ohw-uix-demo\cabo directory to <OHW-UIX_HOME>\<OHW-UIX_deployment_name>\cabo, and library files from ohw-uix-demo\ohw-uix-demo\WEB-INF\lib directory to <OHW-UIX_HOME>\<OHW-UIX_deployment_name>\WEB-INF\lib directory.
5. Copy the uix-config.xml from ohw-uix-demo\ohw-uix-demo\WEB-INF directory and save it in <OHW-UIX_HOME>\<OHW-UIX_deployment_name>\WEB-INF directory. Then, edit the file to your requirements.
6. Copy the web.xml from ohw-uix-demo\ohw-uix-demo\WEB-INF directory and save it in <OHW-UIX_HOME>\<OHW-UIX_deployment_name>\WEB-INF directory. Then, edit it to your requirements:
   a. Modify the <display-name></<display-name> and <description></description> section to display your custom helpset name. For example:

      ```
      <web-app>
        <display-name>My Module</display-name>
        <description>My module help</description>
      </web-app>
      ```

   b. Optionally, you may want to edit the <servlet-name> element under <servlet> element to change your URL used to access OHW-UIX. For more information about changing the access URL, see Changing the OHW-UIX Access URL.
7. Compress the <OHW-UIX_deployment_name> directory into a WAR file.
8. Copy the application.xml from ohw-uix-demo\META-INF directory and save it in <OHW-UIX_HOME>\META-INF directory. Then, edit it to your requirements. In this file, you provide the web module name of each product that you deploy. Specify the WAR file name, created in step 7, in <web-uri></web-uri> element. If you want to change the access URL of the application, update the

<context-root></context-root> element. For more information, see Section D.4.4, "Changing the OHW-UIX Access URL".

9. Compress the <OHW-UIX_HOME> directory into a EAR file.
10. Start the Oracle WebLogic Server and deploy the EAR file. If Oracle WebLogic Server is already running, you must shut it down and then restart it before the changes made since you last started the servlet is available.
11. Direct the browser to http://<hostname>:<port>/<OHW-UIX_deployment_name>/help/, where <hostname> is the name of the system on which OHW-UIX and Oracle WebLogic Server are installed.

    The first page of the demo help system displays in the browser. If there is more than one helpset, use the dropdown list in the toolbar to select a helpset, then click the helpset switcher to display the TOC and index from the selected helpset only. The text search searches only for items in the selected helpset.

D.4.4 Changing the OHW-UIX Access URL

The URL to access OHW-UIX is http://<hostname>:<port>/mymodule/help/, where <hostname> is the name of the system on which OHW-UIX and Oracle WebLogic Server are installed.

You can change this URL in the following ways:

- Changing the final URL element of the access URL
- Changing the access URL to another name

D.4.4.1 Changing the final URL element of the access URL

To change the help at the end of the URL, edit web.xml in <OHW-UIX_HOME>\<OHW-UIX_deployment_name>\WEB-INF.

The <servlet-mapping> parameter <url-pattern> specifies the URL used to access OHW-UIX. For example, if you change <url-pattern> from the default /help/* to /onlinereference/*, the URL used to access OHW-UIX would become http://<hostname>:<port>/mymodule/onlinereference/.

For example:

```
<servlet-mapping>
   <servlet-name>mymodule</servlet-name>
   <url-pattern>/onlinereference/*</url-pattern>
</servlet-mapping>
```

D.4.4.2 Changing the access URL to another name

To change the access URL for your application, edit the <context-root> element entry under <web> element in application.xml, located in <OHW-UIX_HOME>\META-INF:

```
<web>
   <web-uri>my_module.war</web-uri>
   <context-root>my_module</context-root>
</web>
```

For example, if you want the OHW-UIX access URL to be http://<hostname>:<port>/jdeveloper/help/, modify the root element:

```
<web>
   <web-uri>my_module.war</web-uri>
```

```
   <context-root>jdeveloper</context-root>
</web>
```

D.4.5 Upgrading OHW-UIX and UIX

When new versions of OHW-UIX and UIX are released, be sure to check the OHW-UIX and UIX download pages for the latest download and install instructions before upgrading your OHW-UIX installation.

- To upgrade OHW-UIX to a newer version, you must replace the OHW-UIX JAR file located in the `WEB-INF/lib` directory.
- To upgrade UIX to a newer version, you must replace the UIX JAR file located in the `WEB-INF/lib` directory, and also replace the UIX installable resource files (distributed in `uix2-install.zip`) by unpacking them into the `cabo` directory.

To test your upgrade, restart the servlet container and point your browser to `http://<hostname>:<port>/ohw-uix-demo/help/`, or wherever you have mapped the OHW-UIX application.

D.5 Implementing Context-Sensitive Help in a Web Application

Oracle Help for the Web – UIX (OHW-UIX) provides a context-sensitive help mechanism that launches help topics that are associated with some context in the web application user interface. Typically, help topics are written to describe the function of a particular page, table, or input field in a web application. When a user requests help for a user interface control—for example, by selecting a Help button—the appropriate topic for that context, or control, is displayed.

To provide context-sensitive help for a web application, the help system must include one or more map files and the appropriate help code must be added to the application code.

The following sections describe how to implement context-sensitive help using OHW-UIX:

- Section D.5.1, "Mapping Topic IDs to Help Topics"
- Section D.5.2, "Creating Context-Sensitive Links to the Help System"
- Section D.5.3, "Implementing Context-Sensitive Help in Oracle UIX-based Applications"

D.5.1 Mapping Topic IDs to Help Topics

OHW-UIX context-sensitive help systems rely on one or more map files that map topic IDs to help topic HTML files. In a helpset, the map file is in XML file format.

The map file is usually created by the help author. As a web application developer, when associating web application controls with context-sensitive topics you must use the topic IDs specified in the author's map file. Thus, you must coordinate your efforts with the help author.

Here's a sample map XML file:

```
<?xml version='1.0' ?>
   <map version="1.0">
   <mapID target="topic_1" url="file_1.html" />
   <mapID target="topic_2" url="file_2.html#a1" />
   <mapID target="topic_3" url="file_3.html" wintype="intro" />
```

```
</map>
```

The attribute target specifies a unique ID for the associated HTML file within a helpset. The attribute url specifies the location of the file to associate with the ID. The wintype attribute is optional; it specifies the name of a window type that the topic would be displayed in. For more information about the elements used in the map file, see Section 12.3.2, "Contents of an OHJDK Release".

D.5.2 Creating Context-Sensitive Links to the Help System

Applications that rely on OHW-UIX for context-sensitive help request the context-sensitive topics through specially formulated URLs to the OHW-UIX servlet. Any user interface control with a URL destination (links, images, and so on) can be associated with a context-sensitive topic.

When creating a link to OHW-UIX for context-sensitive help, you can either use the URL destination for the front page (with the **Contents**, **Index**, and **Search** navigators), or you can create a URL destination for a topic using the topic ID. You can also specify a locale in the URL destination.

- Section D.5.2.1, "Linking to the Front Page"
- Section D.5.2.2, "Linking to a Topic"
- Section D.5.2.3, "Specifying the Locale"

D.5.2.1 Linking to the Front Page

The URL to the front page is simply the URL to the OHW-UIX servlet:

```
http://<server>:<port>/<servlet mapping>
```

where `<server>` is the name of your server running the servlet container, `<port>` is the port used by the servlet container, and `<servlet mapping>` is the servlet mapping set up in the `web.xml` file for the OHW-UIX servlet (by default this is `ohw-uix/help/`). For example:

```
http://www.yourcompany.com:7101/ohw-uix/help/
```

When a user requests help for a user interface control that is linked to the front page, OHW-UIX is displayed in the user's browser, showing the first page of the help system (usually a table of contents).

D.5.2.2 Linking to a Topic

To create the URL for linking to a topic, add a topic parameter to the URL of the OHW-UIX servlet. The value of the `topic` parameter is the topic ID of the help topic:

```
http://<server>:<port>/<servlet mapping>/?topic=<topic-id>
```

For example, the following URL requests the topic associated with the topic ID `topic_1`:

```
http://www.yourcompany.com:7101/ohw-uix/help/?topic=topic_1
```

When implementing context-sensitive links to OHW-UIX, you may also want to use JavaScript to open the link in a secondary window rather than replace the main application page.

When a user requests help for a user interface control that is linked to a topic ID, OHW-UIX displays the file associated with the topic ID in a window page that does

not include the OHW-UIX navigators (tabs). However, the topic page has a link to the front page of the help system should the user want to access the main help.

D.5.2.3 Specifying the Locale

When you link to any OHW-UIX page, including topic pages or front pages, you can include a locale in the URL of the OHW-UIX servlet with the `locale` query parameter.

The topic syntax is:

```
http://<server>:<port>/<servlet-mapping>/?topic=<topic-id>&locale=<ISO-code>
```

and the front page syntax is:

```
http://<server>:<port>/<servlet-mapping>/help?locale=<ISO-code>
```

If you specify the locale, OHW-UIX switches to the localized helpset if it is available, and keeps using the specified locale until it is overridden or removed. If the specified localized helpset is not available, the parameter is ignored.

For example:

```
http://www.yourcompany.com:7101/ohw-uix/help/?topic=topic_1&locale=sp
```

For more information about locale, see Chapter 8, "Oracle Help for the Web Configuration File".

D.5.3 Implementing Context-Sensitive Help in Oracle UIX-based Applications

UIX is an Oracle technology for creating web applications. UIX provides mechanisms that make it easy to provide context-sensitive help through OHW-UIX. With UIX, you can implement context-sensitive help programmatically using the UIX Java API, or declaratively using the UIX language (an XML language).

Note: UIX is not shipped with current release of JDeveloper, but if you want to use UIX, download JDeveloper 10.1.2 or any earlier release from OTN archives. More information about the `uix-config.xml` file is available in the JDeveloper online help.

The `HelpProvider` architecture in UIX provides a generic context-sensitive help mechanism. OHW-UIX provides context-sensitive help for UIX applications through a specific implementation of `HelpProvider` called the `OracleHelpProvider`.

To use the `OracleHelpProvider`, you must register OHW-UIX with the application, then specify the context-sensitive help links through databinding.

D.5.3.1 Registering OHW-UIX in the OracleHelpProvider

The first step in using the `OracleHelpProvider` is to register your `OracleHelpProvider` instance (i.e., OHW-UIX) with the UIX Configuration object. In UIX, the `HelpProvider` appears as a special UIX DataProvider that can be used for databinding. It is special in that you do not require to declare it in your UIX page, it is available in all pages once you register your `HelpProvider` with the Configuration object.

In UIX, you can use the `uix-config.xml` file and `ApplicationConfiguration` API to create a set of configuration objects without writing a line of code, and update configuration properties in the field without recompiling code.

To register OHW-UIX with your application, modify the `uix-config.xml` file to point UIX to an instance of the OHW-UIX servlet.

Here is a sample `uix-config.xml` file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

 <configurations xmlns="http://xmlns.oracle.com/uix/config">
   ...

   <default-configuration>

     <help-provider>
       <ohw-servlet-url>http://www.yourcompany.com:7101/ohw-uix</ohw-servlet-url>
     </help-provider>

   </default-configuration>

   ...
 </configurations>
```

The `<help-provider>` element allows configuration of a help provider. The only supported syntax at this time is a contained `<ohw-servlet-url>` element. The `ohw-servlet-url` must contain an URL that points to an installation of OHW-UIX. Once you've set this property, all uiXML and UIX Java pages have access to two data providers: `ui:helpTopics` and `ui:helpSystem`.

In UIX, if you want to use Java code to create your Configuration object but want to use the default properties defined in the `uix-config.xml` file, you would use the following code:

```
 ApplicationConfiguration appConfig =
   ApplicationConfiguration.getInstance(servletContext);
 configurationImpl impl =
   new ConfigurationImpl ("someName", appConfig.getDefault());
 impl.register(servletContext);
```

To register OHW with your application programmatically in UIX, see the following sample code.

```
protected ConfigurationImpl createDefaultConfiguration()
{
   ConfigurationImpl cfg = super.createDefaultConfiguration();
   //For your application you'd likely pull the location of the
   //OHW servlet out of a servlet init parameter

   OracleHelpProvider provider = new
OracleHelpProvider("http://www.yourcompany.com:7101/ohw-uix/help/")
   cfg.setProperty(Configuration.HELP_PROVIDER, provider);
   return cfg;
}
```

The `HelpProvider` sets up two special data objects (`helpTopics` and `helpSystem` in the UIX UI Components namespace).

D.5.3.2 Databinding a Destination

The `HelpProvider` sets up two data providers—`ui:helpTopics` and `ui:helpSystem`. Here, `ui` is used as the prefix for the UIX UI namespace. They are used for databinding the destination attribute of links or buttons (or any control that has a destination) from which you want to connect to the help system.

After registering OHW-UIX with your UIX-based application, you can then specify context-sensitive help declaratively using the data objects `ui:helpTopics` and `ui:helpSystem`.

Databinding a Destination to the Front Page

Using declarative UIX, a destination can be created for the front page by using the special frontPage key for the `ui:helpSystem` data object. For example:

```
<globalButton icon="globalhelp.gif" text="Help"
data:destination="frontPage@ui:helpSystem"/>
```

When a user requests help for a user interface control that is linked to the front page, OHW-UIX is displayed in the user's browser, showing the first page of the help system.

> **Note:** The first page of the help system is defined as the first navigator declared in the `.hs` file and the first book defined in the OHW-UIX configuration file (`ohwconfig.xml` or another name specified by the `configFileName` initialization parameter for the servlet). Typically the first navigator of the first book is a table of contents.

Databinding a Destination to a Topic

To show a topic, use the unique topic ID as the key for the `ui:helpTopics` data object. For example:

```
<button text="Button To Help" data:destination="myTopicID@ui:helpTopics" />
<link text="Link To Help" data:destination="someOtherTopicID@ui:helpTopics" />
```

At runtime, UIX uses the `OracleHelpProvider` instance to resolve the value of these destinations. The `OracleHelpProvider` automatically returns a destination that includes JavaScript to launch help in a separate, smaller browser window. This window has a link to the front page of the help system should the user want to access the main help.

D.6 Upgrading OHW-UIX Help System to OHW Help System

When upgrading from OHW-UIX to OHW, the helpsets contents may remain unchanged, however, there are two things you must do to use OHW:

- Copy the OHW jar files into the `WEB-INF/lib` directory.
- Modify the `web.xml` file under the `WEB-INF/` directory. The best way to do this is to get the `web.xml` from the OHW demo bundle and copy the servlet and resource definitions and mappings.

There are three major parts required in a `web.xml` for OHW:

- Basic configuration for JSF Trinidad and ADF RC (Faces Servlet, Trinidad Servlet and Filter, and so on):

```
<context-param>
  <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
  <param-value>client</param-value>
</context-param>

<filter>
  <filter-name>trinidad</filter-name>
```

```
<filter-class>org.apache.myfaces.trinidad.webapp.TrinidadFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>trinidad</filter-name>
  <servlet-name>Faces Servlet</servlet-name>
</filter-mapping>

<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>/faces/*</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>resources</servlet-name>

<servlet-class>org.apache.myfaces.trinidad.webapp.ResourceServlet</servlet-clas
s>
</servlet>

<servlet-mapping>
  <servlet-name>resources</servlet-name>
  <url-pattern>/adf/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>resources</servlet-name>
  <url-pattern>/afr/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>resources</servlet-name>
  <url-pattern>/ohr/*</url-pattern>
</servlet-mapping>
```

- Basic configuration for OHW:

  You must define an OHW Filter and map it to the Faces servlet.

  ```
  <filter>
    <filter-name>OHWRCFRequestFilter</filter-name>
    <filter-class>oracle.help.web.rich.OHWFilter</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>OHWRCFRequestFilter</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
  </filter-mapping>
  ```

- OHW-UIX Servlet definition and servlet mapping:

  The existing OHW-UIX servlet definitions must be changed to use the OHW class `oracle.help.web.rich.OHWServlet`. The following steps describe how to change existing OHW servlet definitions:

1. To specify the location of the `ohw-config.xml` for a OHW-UIX servlet instance, define OHW_UIX servlets with `<init-param>` named `ohwConfigFileURL`.
2. Define the URL mapping for OHW-UIX servlets. You do not have to change the existing servlet mappings.
3. Specify the `<load-on-startup>` parameter.

OHW supports multiple OHW instances in one Web application. Here is an example which deploys two OHW instances in a `web.xml`:

```
<!-- configuration for product1 help front servlet -->
<servlet>
  <servlet-name>product1</servlet-name>
  <servlet-class>oracle.help.web.rich.OHWServlet</servlet-class>
  <init-param>
    <param-name>ohwConfigFileURL</param-name>
    <param-value>/helpsets/product1/ohwconfig.xml</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>resources</servlet-name>
  <url-pattern>/ohr/*</url-pattern>
</servlet-mapping>

<!-- configuration for product1 help front servlet -->
<servlet>
  <servlet-name>product2</servlet-name>
  <servlet-class>oracle.help.web.rich.OHWServlet</servlet-class>
  <init-param>
    <param-name>ohwConfigFileURL</param-name>
    <param-value>/helpsets/product2/ohwconfig.xml</param-value>
  </init-param>
  <load-on-startup>3</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>product1</servlet-name>
  <url-pattern>/product1/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>product2</servlet-name>
  <url-pattern>/product2/*</url-pattern>
</servlet-mapping>
```