4

Preparing Help Topics

The two most important features of any online documentation system are well-written, meaningful topics and a good navigation facility. This chapter discusses the former.

JavaHelp topics are written in HTML. This chapter doesn't teach you HTML, but it explains how to apply HTML to certain parts of JavaHelp topics. If you are not familiar with HTML, you may want to consult O'Reilly's *HTML: The Definitive Guide*.

While most of the concepts in this chapter apply to all online documentation systems, I discuss them as they apply to JavaHelp. To help you create well-written and meaningful JavaHelp topics, this chapter provides the following guidelines for accomplishing the following tasks:

- Planning your help topics
- Creating help topics and applying appropriate HTML tags
- Writing effective and meaningful help topics
- Using preexisting HTML topic files

Planning Your Help Topics

In the previous chapter I outlined the entire JavaHelp project. I also explained that you should determine general help-topic areas based on your research of the audience and their needs. In this section I examine the following basic tasks that break down the general topic subject areas into specific online help topics:

- Assigning work to help authors
- Organizing information into help topics

- Obtaining approval from team members

Assigning Work to Help Authors

If you are working with a team of help authors, there are various methods for dividing the work among the team members. One method is to make each help author responsible for a different type of topic. For example, if three help authors were working on a project, one could work on field-level help, another on conceptual help, and the other on procedural help.

If the help authors have limited time for learning the application for which they are writing online help, you might try another method, in which each writer works in a specific subject area (such as creating new documents or formatting text). This method has each writer creating topics of all types (field-level, conceptual, and procedural) for his or her assigned subject area. The benefit to this approach is that each author spends time learning only a small portion of the application.

For large projects, you should assign the help-topic types (field-level, conceptual, procedural, etc.) to teams. Then have each team allocate help topics to the individual help authors. For example, if a team of help authors is working on field-level help, have each author work on the topics for different screens. Then, you won't have help authors duplicating efforts, and each author can remain focused on a specific type of help topic. Again, depending on your resources, you might have help authors focus on particular subject areas instead of topic types, to minimize the time required for learning the subject matter.

If you are working alone on the project, it's still wise to think of the different topic types and subject areas as separate components. Although you are responsible for everything, breaking things down can help you approach the project in an organized manner.

Organizing Information into Help Topics

Once you know the specific topic types and subject areas for which you are responsible, you can begin organizing specific information into individual help topics—a process known as *chunking*. For example, if you want to create procedural help that explains how to use the word processor's formatting tools, determine specific procedures, such as how to embolden text, how to italicize text, and how to color text. You then place each of these procedures in a separate help topic instead of combining them in one long topic.

At this point in the project you should have an idea of the information the help topics will contain, so you can give each topic a title. Keep your project organized as you create the topic titles. In the last chapter, I used the word-processor-applica-

tion example to show how to organize the general help types and subject areas into a directory and file structure. When you decide on the topic titles, start creating the actual files within your directory structure. You won't write the topic content yet, but you can accomplish two tasks at once if you create the topic's HTML file at the same time you outline your topics and create topic titles. For example, you can create actual HTML files for the help topics on emboldening, italicizing, and coloring text. You aren't yet ready to write the actual content or procedures within the help topics, but you can determine their titles based on the information the topics provide. Later, when you write the content of each help topic, you can simply use the appropriate HTML file you have already created.

Based on the word processor example, Figure 4-1 shows how you might set up a directory and file structure for help topics on formatting text.

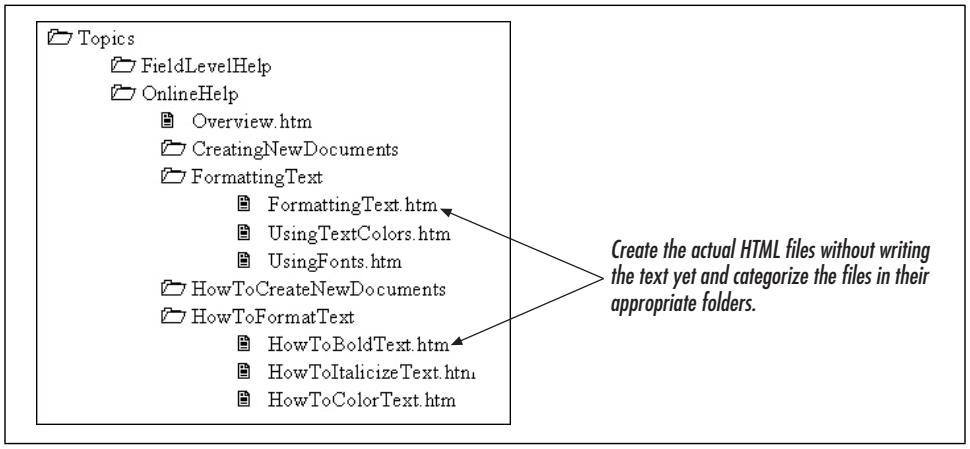

Figure 4-1. Organizing help topic files

When I outline topics for a new help project, I usually create and categorize the topic files with their actual topic titles as shown in Figure 4-1. Then, when I want to print out or show someone my outline, I simply take screen shots of the directory and file structure and use the screen shots as my printed outline.

Later in this chapter I provide some tips on chunking the individual help topics, designing topic titles, and naming topic files.

Obtaining Approval from Team Members

When you have determined the specific help topics you will prepare, you should present them to other members on your development team to get general approval. Depending on the company for whom you work, this approval could be a verbal "sounds good to us," or it could be a formal proposal, requiring signa-

tures from team members, verifying that your help topics are appropriate for the project. Make sure that everyone agrees that the topics you have outlined can be prepared by the project deadline.

If you present the report to managers, or if you are working under particular deadlines, you should include anticipated completion dates to show that you can complete the help topics in time for the project deadline. In Chapter 3, *Planning the JavaHelp Project*, I discussed determining the time required for you to create help topics.

Creating Help Topics and Applying Appropriate HTML Tags

Before you start writing the text for your help topics, you should understand the concepts behind naming the topic's HTML file, formatting text in the topic, and using links to different topics and web sites.

Naming the HTML Files

When you create your HTML files, consider the tips in this section for using meaningful filenames and keeping your project well organized. If you use consistent, logical filenames, you should be able to recognize the subject of the help topic just by looking at its filename.

When naming files, consider the following related HelpSet items:

- Filenames
- Map IDs
- Topic titles

Name the help-topic files in a manner such that anyone can easily match them to their corresponding map ID and topic title. However, you don't want to give the files ridiculously long names that become hard to manage. I tend to use nearly identical names for all three related items. So, for a word-processor help topic on formatting text, I would use the map ID "FormattingText," the file name "FormattingText.htm," and the topic title "Formatting Text."

NOTE If you use a third-party help-authoring tool to create your HelpSet, you might not have a choice in assigning the map ID: the third-party tool might automatically create it. However, third-party tools that automatically assign the map ID typically allow you to work at the topic-title level, so that you don't have to pay attention to map IDs.

Applying Formats to Text

Before writing text for your help topics, you should know how to use HTML formatting tags in JavaHelp topics. As you write each topic, you will want to apply formatting tags to the topic title heading, subheadings, body text, bulleted lists, and numbered lists.

In Chapter 2, *Creating Your First HelpSet*, you used HTML tags to format text within help topics. You probably remember that it was as simple as writing a basic web page, since the HelpSet Viewer is based on HTML. Now take a closer look at applying available formatting tags to JavaHelp topics. Figure 4-2 shows the HelpSet Viewer with a help topic that contains a variety of text formatting.

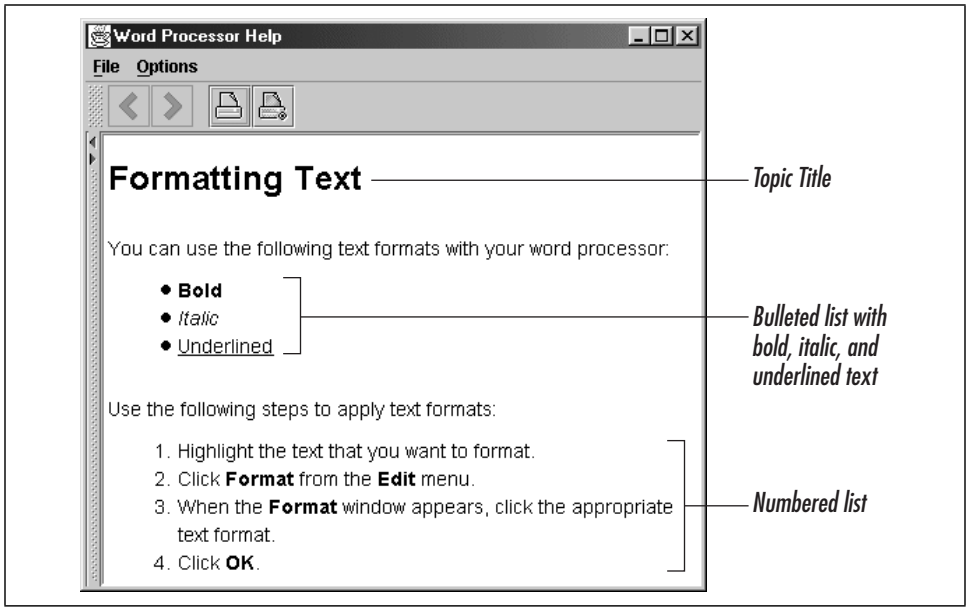

Figure 4-2. Displaying formatted text

Here is the source text for the topic shown in Figure 4-2; it uses standard HTML formatting tags.

```
<html>
<head>
<title>Formatting Text</title>
</head>
<body>
<h1>Formatting Text</h1>
<p>You can use the following text formats with the word processor:
<ul>
  <li><strong>bold</strong></li>
```

```
    <li><em>italic</em></li>
    <li><u>underlined</u> </li>
  </ul>
  <p>Use the following steps to apply text formats:
  <ol>
    <li>Highlight the text that you want to format.</li>
    <li>Click <strong>Format</strong> from the <strong>Edit</strong> menu.</li>
    <li>When the <strong>Format</strong> window appears, click the appropriate
        text format.</li>
    <li>Click <strong>OK</strong>.</li>
  </ol>
  </body>
  </html>
```

This text uses the following HTML formatting tags:

- `<h1>` identifies the topic title. You can use `<h2>` and additional heading levels throughout the help topic if you want to break it down into subsections.
- `<ul>` and `<li>` identify the bulleted list.
- `<strong>` identifies bold text, `<em>` identifies italicized text, and `<u>` identifies underlined text.
- `<ol>` and `<li>` identify the numbered list.

At the time of this writing, the HelpSet Viewer doesn't support the following HTML tags or attributes:

- `<font>` tag is not reliable; you can use basic fonts, such as Times or Arial, but the HelpSet Viewer doesn't interpret all fonts.

  If you don't specify a font with the `<font>` tag, JavaHelp uses a default sans-serif font.
- `<map>` and `<img usemap>` tags (for client-side image maps).
- `<tt>` and `<code>` tags (for moonscape typeface).

You should refer to JavaHelp's documentation or Sun's web site for ongoing updates to this information.

Keep in mind that you should not use a lot of text formatting in the same help topic just because you can. Too much formatting can be a strain on the reader's eyes. Reserve text formats such as bold, italics, underlines, and color for special conditions. These text formats will be more effective if you use them sparingly.

Using Links

As you create your help topics, you should consider how topics relate to one another. You might have a help topic that explains the basic concepts behind

using word-processor templates and another topic that provides step-by-step procedures for using templates, but the user might not know that both of these topics exist.

You can help the user locate related topics by using HTML hyperlinks. You created a hyperlink in Chapter 2 to enable the user to click selected text and jump to the related help topic. Creating a hyperlink is easy; you use HTML code within your help topic as demonstrated in the following example:

```
This online help system also provides conceptual information on
<a href="../UsingTemplates.htm">using templates</a>.
```

In this example, `<a href="../UsingTemplates.htm">` sets up the hyperlink so that the user can simply click the text "using templates" to jump to the related topic.

When using this kind of link, be careful selecting text for which you apply the hyperlink; you don't want to select more text than necessary. In the previous example I applied the link only to "using templates" instead of to the entire phrase "conceptual information on using templates." The latter would have disturbed the smooth flow for readers not interested in the hyperlink.

Don't use too many hyperlinks in one help topic. By following too many hyperlinks, users risk getting lost in your help system and become frustrated enough to stop using it. Also, make sure the hyperlink actually links to a topic that expands on the subject. I have seen many cases where authors include hyperlinks that send the user to another topic, only for the user to find that the new topic doesn't actually provide additional information on the subject.

Another way to help your users is to have a "Related Topics" or "See Also" section at the bottom of the help topic. This section provides a list of other topics in the current HelpSet (or in a wider scope) that are related to the topic the user is currently reading. To make effective use of related topics, make a note of which help topics closely relate to others while you are organizing and writing the help topics.

Your audience uses related topics in a different manner than they use other hyperlinks. They turn to related topics specifically to see what other help topics exist that might help them find more information on the current topic. Since you don't force the related topics feature on users (you place it at the bottom of the help topic) you can include all topics that relate to the current one—even if you already included the topic earlier in a hyperlink.

Writing Effective and Meaningful Help Topics

While no book can make you an expert on information design, and while the purpose of this book is to teach you how to develop JavaHelp projects, I provide some tips in this section to help you prepare better JavaHelp topics:

TIP If you are already experienced with designing help topics, you might want to skip the rest of this chapter.

Understanding How Users Read Online Help

The best way to start this discussion is to explain the way users don't read online help. They don't read online help the way you are reading this book. With online help, the user doesn't read the document from start to finish. There is neither a beginning nor an end to the document.

Generally, users enter a help system with a question for which they want an answer or a task for which they need instructions. Users are typically in the middle of working with the application and don't want to spend time reading through a lot of information. They tend to skip over sections and skim through sections to find the specific information they need.

The troubling fact about online help is that you don't have many chances to give users what they want. Typically, users will try only a few times to find the information they want before giving up on finding the topic. After several futile attempts with the same help system, users start to lose faith in the information the help system provides. If you don't carefully craft the help topics, as well as the navigational facility that gets the users to the topics, you could very well waste time developing a help system that no one uses. The solution, however, is *not* to omit the help system, afraid of wasting valuable time. Instead, the solution is to take the time to create well-written and meaningful help topics through which users can find the information they need.

Chunking Information

As mentioned earlier in this chapter, when you chunk information, you are breaking it down into separate topics so that each chunk of information treats only one subject. For examples of chunking, look through this book. You'll see that information is chunked into sections within each chapter. This section provides only information about chunking. The previous section provides only information about how users read online help.

You must be more specific when chunking information for online help than for hardcopy documents. Using the word processor example, it's not enough to provide one topic on formatting text. Instead, you need one topic that lists the types of formatting options, one topic that describes how to use bolding, one topic that explains how to use italics, and as many other topics needed to cover all formatting options.

Thus, Figure 4-2 may not be a good model to follow for designing a help topic. While the entire help topic discusses only text formatting, it presents multiple topic types and attempts to answer more than one question. It provides concepts on the word processor's text-formatting options and also gives procedures for formatting the text. A good help system would have the conceptual information in one topic and the procedural information in another.

Writing the Topic Content

After you have chunked information into separate subjects, you can begin writing the topic content. Approach writing each topic as an individual document. Never assume the user has read any other help topic prior to the one you are writing. This approach means you can't assume users have any previous knowledge of the subject. Good information chunking will keep you from writing topics that require prerequisite information and therefore eliminate the tendency to write introductory material in every help topic.

Most online-help users skim through information instead of reading every word and sentence. To help accommodate users' reading habits, you must write short, clear, concise sentences and use short paragraphs. You should also use small, simple words. As you write each sentence and are thinking about the right words to use, remember that one important JavaHelp navigation component, the word search index, finds topics based on the words you use within the help topic. Users type a word, and the JavaHelp index presents a list of the help topics containing that word. Therefore, try to use words you anticipate someone will use in a word search.

In addition to sentence and word structure, consider the following tips while writing your help topics:

- Avoid jargon and define all technical words and abbreviations. In Chapter 5, *Creating HelpSet Data and Navigation Files*, I discuss pop-up windows and show you how you can use them to define new words.
- If you must write a long help topic, break the topic into subheadings. Since users scan through the help topic, subheadings help them locate particular sections.

- Don't avoid writing a help topic because the subject is too confusing. Spend the time to make the subject clear and easy to understand. If a procedure in the application is difficult to document, it is probably difficult to perform. You should look at such procedures to see if you should change the actual application interface to make it easier for users.

  This tip might seem like a lot of additional work, but your users will appreciate it. The documentation phase presents an excellent opportunity to test the application's interface. Take advantage of it. Instead of trying to cover up interface flaws with awkward documentation, fix the interface so that both the application and its documentation will be friendlier to your users.

- Be consistent with your writing style and choice of words. If you refer to the software application as the "system," use this word consistently. Don't use "system" in one place, "application" in another, and "software" in another. The user will waste time trying to find the difference in meaning among these words when in fact you use them all to mean the same thing.

- Place cautions and important notes before the passage they modify. For example, if you want to tell the users that performing Step 3 will delete the contents on their hard drive, tell them before they read Step 3. Don't wait until after the user performs the action. This tip might sound obvious, but many writers place cautions like this one at the bottom of the topic or after the particular step. People won't read everything before performing a step, so make sure they know important consequences before they take action.

- Users can process about three to seven online items. This characteristic means you should limit procedures to no more than seven steps. If you must use more then seven steps, try to break the procedure into multiple procedures. However, sometimes breaking procedures apart just to avoid using more than seven steps is not feasible. You are better off giving the user nine steps for a task than creating multiple procedures and making users jump through the help system to finish the task.

- Avoid using the future tense by eliminating the word "will." For example, don't say, "When you select the formatting option, the system *will* display the formatting screen." Instead say, "When you select the formatting option, the system displays the formatting screen." Users are taking action in the present tense. Making a sentence future tense by using the word "will" not only is inconsistent with the user's present action, but also makes sentences unnecessarily long and awkward.

- Write in the active voice instead of the passive voice. When you write in the active voice, you make something take action on something else instead of the "something else" being acted upon. For example, if you say, "When you select the formatting option, the system displays the formatting screen," you are

using the active voice because the user takes action on the system, and the system takes action on the formatting screen. However, if you say, "The formatting screen is displayed when the formatting option is selected by the user," you are using the passive voice because the formatting screen and the formatting option are being acted upon.

Another problem with using the passive voice is that it generally forces using a phrase such as "the user" as I did in the previous example. Since the users are the people for whom you are writing, write to them directly instead of writing about them. Use the word "you" as I did in the example of an active sentence instead of the phrase "the user" as I did in the example of a passive sentence.

Setting the Topic Length

When you are writing the help content, keep in mind the length of the help topic. There is no magical number of words per topic that will work best for your users. Setting such a number stops you from providing important information in an attempt to avoid writing longer topics. It also forces you to write longer topics in place of shorter ones just to fill up space. You must provide as much information as is required to discuss only the given subject—nothing more, nothing less.

Take a common-sense approach to setting topic length. When was the last time you enjoyed reading a lengthy topic while trying to operate a software application? Your users are most likely no different from you when it comes to reading software documentation. Give them only the information they are looking for. If doing so means writing a longer topic, your users will probably be happy that all the information they want is there. Remember, though, to break down longer help topics into subsections with their own subheadings.

Designing the Topic Title

Once you have written the topic's content, you must give the topic a title. Users should know if they want to read a help topic simply by reading its title. The topic title should tell users whether or not the help topic contains the answer to their question. For example, "Applying Bold Formatting to Text" says more to the user than "Bold Formatting."

In designing a topic title, be sure to use a short phrase that concisely describes the topic's contents. Like the topic itself, topic titles should be self-contained. Don't make a help topic title depend on text or titles from other topics. Users will see this title in JavaHelp's table of contents and must decide from there what the topic title means.

Using the <title> tags to specify the topic title

As discussed in Chapter 1, *Understanding JavaHelp*, the word-search index uses the information you place in the `<title>` tags of your help-topic files. For this reason, you should use the topic title recorded in the TOC file in your help files' `<title>` tags. For our word-processing help example, you would use the following line in the topic file to specify its title:

```
<title>Formatting text</title>
```

NOTE It's up to you to keep the title you specify with `<title>` consistent with the topic title you specify in the TOC file.

Using Preexisting HTML Topic Files

It is quite possible that you might start a JavaHelp project with all of your HTML topic files already created. For example, you might be converting a preexisting HTML-based help system to JavaHelp, or you might simply have created all of your HTML topic files before beginning your JavaHelp-level work.

If you are using preexisting HTML files, rest assured that your workload for creating the HelpSet will be nearly the same as if you created the HTML files specifically for your HelpSet. Remember, JavaHelp uses HTML for its topic files. All you have to do is point to the HTML files in your map file. You should, however, perform a quick check to make sure the HTML files meet the standards I discussed in this chapter. For example, you should check that your HTML files don't contain markup tags or script that the HelpSet Viewer can't interpret. You should also check that each file contains the topic title between the `<title>` tags (so that the topic titles appear in a JavaHelp search).