8

Deploying the Help System to Your Users

This chapter discusses issues involved in deploying a JavaHelp-based help system to users—delivering the files in the HelpSet(s) you've developed, along with the Java class files that implement the JavaHelp system itself. This chapter covers the following topics:

- Encapsulating the HelpSet
- Delivering all the required files
- Ensuring basic Java support

Encapsulating the HelpSet

A HelpSet can comprise hundreds or even thousands of files. When you deliver your JavaHelp system to users, you probably want to encapsulate each HelpSet into a single Java archive (JAR) file. (Some people call this "JARing the HelpSet.") This is usually preferable for you and for your users, avoiding the need to handle vast multitudes of files. It saves space, too; a JAR file is stored in compressed format.

This decision has no effect on the authoring process or on the logical functioning of the JavaHelp system. A HelpSet's directory and file structure works the same way whether it's encapsulated in a JAR file or it's stored in the regular filesystem.

Before deciding whether or not to JAR your HelpSet, consider whether users (or you) will need to customize the help topics after they are installed at the user's site? If users need to customize the help topics in your HelpSet, you shouldn't place the HelpSet into a JAR file. Depending on the background of your users, they might not know how to retrieve files from the JAR file or how to JAR the HelpSet files again after they make the updates. If you don't JAR the HelpSet, users can simply use an HTML editor to make changes to individual topic files; the changes are instantly accessible.

This analysis also applies to you, the HelpSet author. Consider whether you'll be making any updates to the HelpSet after initial installation. Will the HelpSet be merged with other HelpSets or have cross links with them? In such cases, it is most reliable to deliver the entire HelpSet in a single, consistent JAR file.

Creating the JAR File

If you have followed my suggestions in this book for organizing the directory and file structure of your HelpSet, then creating a JAR file for the HelpSet should be easy. Use the following steps to create a JAR file for the Aviation HelpSet:

1. At a command prompt, go to the Aviation JavaHelp sample master project directory (*Aviation*).
2. Enter this `jar` command:

   ```
   jar -cvf aviation.jar *
   ```

   The option flags `-cvf` specify the options for creating JAR file named *aviation.jar*. The wildcard character `*` specifies that all files in the current directory (*Aviation*) should be included in the JAR file.

This command creates the JAR file *aviation.jar* in the current directory, *Aviation*. Table 8-1 describes some of the common options you can use with the `jar` command; for creating JAR files, `-cvf` is typical.

Table 8-1. Option Flags for Creating JAR files

Option Flag	Description
c	Creates a new JAR file
t	Lists the contents of the specified JAR file
x	Extracts the contents of the specified JAR file
f	Signifies that you are specifying the name of the JAR file
v	Lists names of the files placed in, or extracted from, the JAR file

NOTE Before creating a JAR file for Windows systems, double-check the names of the files and directories specified in your map file (for example, *Map.jhm*). The names in the map file must match the actual file and directory names *exactly*, including upper- and lowercase letters. Windows filesystems look up names in a case-insensitive manner, but JAR files are *case-sensitive* on all platforms. The main indication that you have case problems is when your help topics don't display properly (for example, missing icons or missing content) when you access it through a JAR file, but everything works fine when the HelpSet is accessed from the regular filesystem.

Delivering All the Required Files

This book doesn't discuss particular software installation tools, but here's an informal checklist that helps ensure that you include all the files required for your help system in your software distribution package. For simplicity, assume that you're deploying a software application and its online documentation, implemented as a single HelpSet.

The application itself

Obviously, you need to include all the files that make up the software application itself.

The HelpSet

If you've encapsulated the HelpSet as a JAR file, just that one file goes into the distribution package. (Exception: if you've excluded the HelpSet file and/or map file from the JAR file, be sure to include the individual files in the distribution package.)

If you're not JARing the HelpSet, you need to include all its files individually.

The Java classes that implement the JavaHelp system

At this time, JavaHelp is an extension to Java, not part of Sun Microsystems' Java software distribution. Thus, you should include the JavaHelp JAR file *jh.jar* or *jhall.jar*, which implements JavaHelp support, in the distribution package. Users must include this JAR file on their Java *class path* when starting a Java application or the HelpSet Viewer, in order to make JavaHelp features available.

A sophisticated installation program might check the user's site, to determine whether the JavaHelp JAR file is already installed. (The user might be installing your software on a computer that already has JavaHelp support, perhaps as part of another one of your company's software distributions.) If so, it may not be necessary to install the JavaHelp JAR file at all.

Alternatively, you can completely separate the installation of the JavaHelp JAR file from the installation of the application and its HelpSet.

Ensuring Basic Java Support

Since JavaHelp itself is implemented in Java, the user's computer must support Java program execution. That is, the computer must have the Java runtime system and class libraries installed. It's beyond the scope of this book to discuss installation of Java itself, but this section discusses some version-related issues.

If you are deploying a Java application and help system based on Java SDK 1.1, you must install support for Java Swing (GUI components) separately. This support is implemented in file *swing.jar*. As with the JavaHelp JAR file, users must include file *swing.jar* on their Java class path. If you are deploying a Java application and help system based on Java 2 (SDK 1.2 and higher), the basic Java distribution includes the Swing classes. Thus, there is no need to provide Swing support separately. Table 8-2 provides more information on version matching.

Table 8-2. Compatible Versions of Java and Java Swing

Java SDK	**Java Swing**	**Notes**
1.1	1.1 or 1.1.1	Swing 1.1.1 is included in the JavaHelp distribution. With SDK 1.1, you can't encapsulate HelpSets in JAR files, and you can't print help topics.
1.2	1.1	This version of Swing is built into the SDK. To use Swing 1.1.1, upgrade to SDK 1.2.2: overlaying Swing 1. 1.1 on SDK 1.2 can be difficult.
1.2.2	1.1.1	This version of Swing is built into the SDK.
1.3 (currently at "release candidate" stage)	New version	A new, unnumbered version of Swing is built into the SDK. See *http://java.sun.com/products/jfc/tsc*.