

Providing Help for Web Applications

Austin Avrashow



As your ASP.NET applications become more complex, the need increases to provide user Help. The good news is that you can use the same techniques—and even use the same Help file format—as Windows desktop applications. Austin Avrashow explains.

OVER the course of the past few releases, Windows has refined and improved the features and usability of desktop application Help.

But what about Web applications? Web applications need to provide assistance to users for the same reasons that standard desktop applications do. By observing the trends and conventions in Windows Help, you can apply the same techniques to Internet-based applications.

After all, Web applications are looking more and more like desktop applications every day. You have a nearly identical set of controls available. On the row of navigation buttons of many Web pages, Help is often the rightmost menu item, just as it is for standard Windows applications. You can even hook up a standard HTML Help file to open when a user clicks Help in your ASP.NET application. There are, however, some limitations and considerations you need to be aware of.

HTML Help files

HTML Help, the current version of Windows Help, is the standard way of presenting Help topics in a browser-based

viewer for a Windows desktop application. An HTML Help file (.CHM extension) is an updated version of the Windows Help (.HLP) format.

It's actually pretty easy to allow your Web users to view an HTML Help file. You make the CHM file available from a standard HTML anchor link:

```
<a href= "foo.chm"> Click for help </a>
```

When a user clicks the link, a dialog box asks whether to open the file or save it to the user's hard disk (see **Figure 1**).

If the user chooses Open, a Help Viewer window appears displaying the Help system as if the CHM file were on the user's hard disk and the user double-clicked it (see **Figure 2**).

Note that only Windows users with Internet Explorer 4.01 and above can open CHM files. If needed, you can detect the type of browser and redirect users without the minimum requirements to another type of Help page.

Although it does have limitations when viewed over the

Technology Toolbox	
<input checked="" type="checkbox"/>	ASP.NET
<input checked="" type="checkbox"/>	Online Help



Figure 1. Downloading an HTML Help file.



Figure 2. The HTML Help file opened.

Internet (which I'll discuss later), the CHM file format offers some advantages:

- CHM is a standard format for Windows Help. The viewer is built into all recent Windows computers.
- It's easy for a user to download or view a Help file from a Web page, as you've seen.
- The CHM format provides book-like conventions such as a table of contents, index, and full text search.
- A CHM file acts as a container for all the documentation, so Help can be developed and maintained independently. Modifying its content doesn't require any changes to Web application pages.

Alternatives to CHM Help files

A CHM file may be a good choice for Help information if:

- the Help is for an intranet application where you know the operating system and browser of your application's users;
- you want to provide a simple, standard solution the majority of users will be able to use (and deal with the exceptions in some alternative way); or
- your company wants to make the most current version of a CHM file for a product or application available for anyone to download over the Internet.

There are alternative formats for providing Help information. You could use the same type of <a> link to let Web site users download a PDF file. Both PDF and CHM files can have a table of contents and full text search. CHM has some extra capabilities for navigating (index) and grouping information (using keywords for generating lists of related topics). PDF is more printable.

There are also companies that offer pure HTML documentation products that mimic standard Windows Help. These include WebHelp (eHelp Corporation) and WebWorks Publisher (Quadralay Corporation). These products have the contents, index, and search features available in their familiar tabbed leftmost frame.

Posting Help as HTML files

If you want your Help to be viewable by the widest audience, you can post the Help as HTML files. If your Help began life as a CHM file, you can simply post the source HTML files to the Web server (or to a network location for intranet applications). You then have the flexibility of being able to specify a particular HTML page to display from a hyperlink, rather than always opening the HTML Help at its main topic.

You can decompile CHM files back into their constituent HTML files if the source files aren't available. One reason that the CHM format was developed (it stands for Compiled Help) was to allow a developer to ship one Help file with an application rather than ship and maintain dozens or hundreds of separate HTML files and images.

Since all browsers and operating systems can see HTML pages, you don't have to worry about possible

incompatibilities as with a format such as CHM. On the downside, displaying a single HTML page doesn't offer comprehensive search capabilities or book-like features such as table of contents and index that give a user the ability to navigate around the Help information.

Using ToolTips and validators

A trend in Help systems is to integrate Help into the application. Just as Microsoft Office and Visual Studio provide an internal window or area for Help information, you can also provide assistance to your users from within a Web application using standard features Visual Studio .NET provides.

Adding ToolTips for forms controls lets you provide a quick few words of helpful information. They can display up front what a Validator is checking for behind the scenes.

The Tooltip property for a control or image is translated into a Title attribute in the HTML rendered to the user's browser. The text you choose becomes like a floating caption that gives users a hint about using the control or some added textual description for any of your graphics. Note that list boxes and dropdown lists don't have a Tooltip property.

Tooltip properties are read/write, so you can change a Tooltip programmatically as the user's context changes and the new value will be displayed after the next page postback.

ToolTips for graphic images are beneficial for users with disabilities who use screen readers to access Internet pages. And as an additional benefit, many search engines use the words found in Title attributes as a factor in ranking Web pages for matching search terms.

Validators save roundtrips to the server and make it easy to create custom error messages for your users, but you shouldn't use them to hide the rules and requirements of form fields. Web sites generally indicate required fields using some convention such as an asterisk or a label with bold text. Knowing what is and is not required encourages users to fill out a form knowing they won't get an error if they leave some fields blank.

For some sites, passwords must be at least eight characters. Sometimes sites require a special format for telephone numbers. It's better to indicate these requirements up front than to have users get an error to deal with later.

ASP.NET developers don't have to write complex JavaScript validation code, but we should try to *prevent* errors instead of just catching them. It provides a more pleasant user experience.

How to display Help

One way of integrating Help information is to display an additional window. If you use a standard hyperlink, the user is taken away from her context to view a Help page of information. You can instead direct a hyperlink to pop open a simple JavaScript window to open the link destination in a new window.

A new window appears when a user clicks the hyperlink. Note the link in the status bar (see **Figure 3**).

```
<asp:HyperLink id="HyperLink3" ...
NavigateUrl="javascript:myopen();">
Open New Window using Javascript
</asp:HyperLink>
```

The myopen function is defined in the Webform1.aspx HTML <Head> area:

```
<HEAD>
<title>WebForm1</title>
<meta content="Microsoft Visual Studio.NET 7.0" ...>
...
<script language="javascript">
function myopen(){
    open('help2.aspx',null,'width=300,height=300');
}
</script>
</HEAD>
```

You can have the window open to a page that has a different background color (for example, the yellow background used in the preceding example) and specify window size and other attributes, although you can't specify its exact location—that's up to the browser.

One benefit of setting the height and width when calling the window is that features you don't specifically request (browser buttons, status bar, and so forth) won't be shown. This gives you a nice minimalist window of supplementary information that doesn't disturb the user's task at hand. You could add a Close Window graphic or text in the top right-hand corner, or just let the user click the close box.

You can also use JavaScript alerts, which are like MessageBoxes. One drawback is that they only have some text and an OK button, so they're good for quick errors or warnings.

showHelp

The Javascript showHelp function allows users who have Internet Explorer to open a page in the HTML Help Viewer (see Figure 4). This function is only useful to Windows users who have Internet Explorer.

```
<asp:HyperLink id="HyperLink8" ...
NavigateUrl="javascript:showHelp("foo.aspx");">
Open a Web form in the HTML Help viewer
</asp:HyperLink>
```

```
<asp:HyperLink id="HyperLink9" ...
NavigateUrl="javascript:showHelp("bar.htm");">
Open an HTML file in the HTML Help viewer
</asp:HyperLink>
```

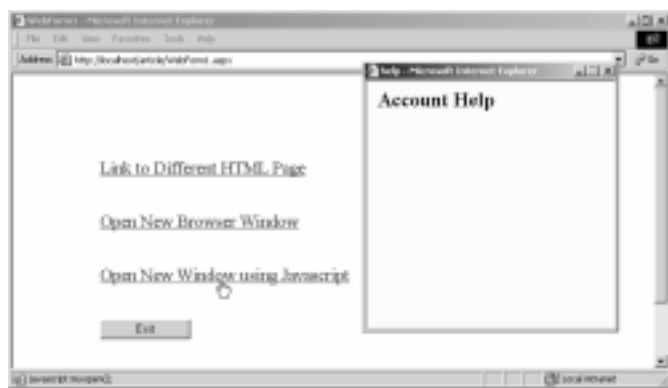


Figure 3. A new window opened.

showHelp offers several advantages over the open function. The Help Viewer window is always on top (it won't get lost behind the main browser window). Also, sending subsequent pages to the Help Viewer displays them inside the same viewer window at the current location and size (that is, the viewer remains in place while you change the content with each showHelp call).

Most importantly, since you can send any .aspx or HTML file types to the Help Viewer, you can invent your own Help format and make it as sophisticated or as simple as you want. Figure 5 shows an HTML page with two frames.

Of course, the downside is the limitation of only working with Internet Explorer. But if your Web application is for your company intranet or you can accept that users won't be able to view your Help if they use alternate browsers, showHelp may be a great choice for displaying Help topics.

Opening a CHM file with showHelp

Under certain conditions, you can also use the showHelp function to open a CHM file, but this can only be done if the CHM file is available on the user's computer in a known location (or at a known location accessible on the network

Continues on page 16

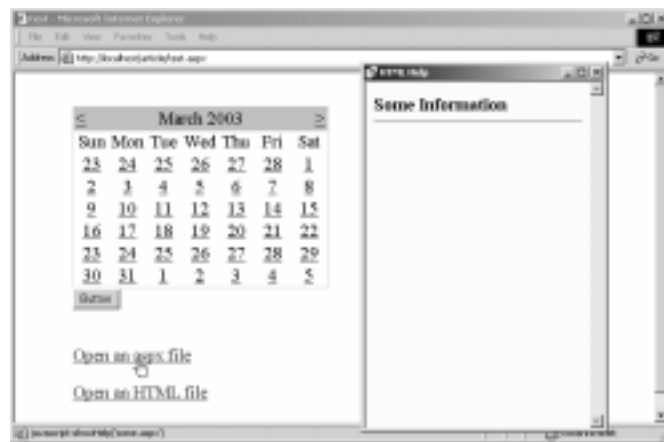


Figure 4. showHelp displays the Help Viewer window.

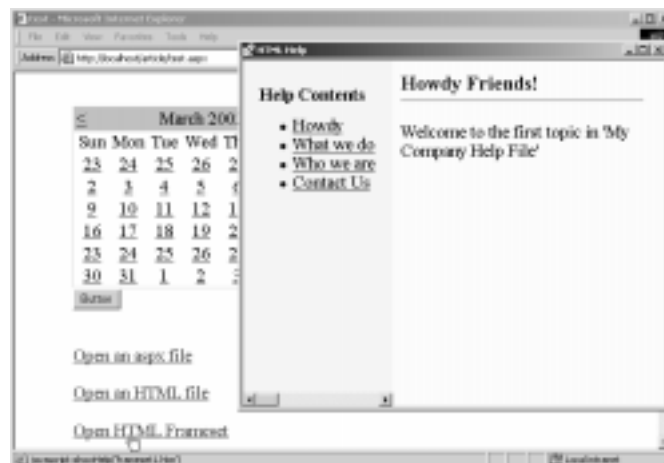


Figure 5. Help Viewer showing a document with frames.

Help for Web Applications...

Continued from page 10

for intranet applications).

```
<a href="javascript:showHelp('c:\\foo\\help.chm');">  
Open the CHM file I've downloaded  
</a>
```

Assuming the CHM file is at a known location on the user's computer, you can also specify the particular topic in the CHM file to display.

```
<a href="javascript:showHelp('c:\\foo\\help.chm', 7);">  
Show a specific help topic  
</a>
```

The second parameter is the identifier for the Help topic to display (this parameter is optional).

The showHelp function is limited in its usefulness for displaying CHM files. Unless your users already have the CHM file on their hard drive at a specific location or download it to be there (C:\foo), you can't use showHelp to

display a CHM Help file.

Your choices for Web application Help

Microsoft is working on a new Help format. Visual Studio .NET is the first commercially available HTML Help 2.0 system. HTML Help 2.0 will be available in a year or so (the word from Microsoft is "2003 at the soonest").

Until then, we can saddle up the standard Windows HTML Help (1.x) CHM format, work with raw HTML files, or use a third-party Help product. Whatever your choice, you should look for ways to guide the Web application user using the tools and techniques Visual Studio .NET offers.

Remember that users move quickly and have little patience. They abandon shopping carts and half-complete forms. Make your users' job easier and they'll tend to visit and use your site more often. ▲



AVRASHOW.ZIP at www.hardcorevdotnet.com

Austin Avrashow is a technical writer and .NET developer in Portland, OR. aavrashow@yahoo.com



April 2003 Downloads

- **TERRELL.ZIP**—Code for Eric Bergman-Terrell's article.
- **AVRASHOW.ZIP**—Code for Austin Avrashow's article.
- **GUNDERLOY.ZIP**—Code for Mike Gunderloy's article.
- **WESTCOTT.ZIP**—Code for Mike Westcott's article.

For access to all current and archive content and source code, log in at www.hardcorevdotnet.com with your unique subscriber user name and password. For access to this issue's Downloads only, click on the "Source Code" button, select the file(s) you want from this issue, and enter the User name and Password at right when prompted.

User name

Password

Editor: Bill Hatfield (billhatfield@edgequest.com)
CEO & Publisher: Mark Ragan
Group Publisher: Connie Austin
Executive Editor: Farion Grove
Production Editor: Andrew McMillan

Questions?

Customer Service:

Phone: 800-493-4867 x.4209 or 312-960-4100
Fax: 312-960-4106
Email: PinPub@Ragan.com

Editorial: FarionG@Ragan.com

Advertising: HowardF@Ragan.com

Pinnacle Web Site: www.pinnaclepublishing.com

Subscription rates

United States: One year (12 issues): \$219; two years (24 issues): \$372
Other:* One year: \$244; two years: \$415

Single issue rate:

\$27.50 (\$32.50 outside United States)*

* Funds must be in U.S. currency.

Hardcore Visual Studio .NET (ISSN 1543-0987) is published monthly (12 times per year) by:

Pinnacle
A division of Lawrence Ragan Communications, Inc.
316 N. Michigan Ave., Suite 400
Chicago, IL 60601

POSTMASTER: Send address changes to Lawrence Ragan Communications, Inc., 316 N. Michigan Ave., Suite 400, Chicago, IL 60601.

Copyright © 2003 by Lawrence Ragan Communications, Inc. All rights reserved. No part of this periodical may be used or reproduced in any fashion whatsoever (except in the case of brief quotations embodied in critical articles and reviews) without the prior written consent of Lawrence Ragan Communications, Inc. Printed in the United States of America.

Hardcore Visual Studio .NET is a trademark of Lawrence Ragan Communications, Inc. Visual Studio .NET, .NET Framework, Visual C#, Microsoft SQL Server, Microsoft Visual Basic, Microsoft Visual C++, Visual C++ .NET, Microsoft Visual Basic .NET, VB.NET, ASP.NET, .NET Enterprise Servers, Microsoft .NET, and Active Server Pages are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other brand and product names are trademarks or registered trademarks of their respective holders. Microsoft Corporation is not responsible in any way for the editorial policy or other contents of the publication.

This publication is intended as a general guide. It covers a highly technical and complex subject and should not be used for making decisions concerning specific products or applications. This publication is sold as is, without warranty of any kind, either express or implied, respecting the contents of this publication, including but not limited to implied warranties for the publication, performance, quality, merchantability, or fitness for any particular purpose. Lawrence Ragan Communications, Inc., shall not be liable to the purchaser or any other person or entity with respect to any liability, loss, or damage caused or alleged to be caused directly or indirectly by this publication. Articles published in *Hardcore Visual Studio .NET* reflect the views of their authors; they may or may not reflect the view of Lawrence Ragan Communications, Inc. Inclusion of advertising inserts does not constitute an endorsement by Lawrence Ragan Communications, Inc., or *Hardcore Visual Studio .NET*.