How to Access Help from HydroDesktop

September 20, 2010

By Tim Whiteaker

# Introduction

HydroDesktop help files are built as HTML files. A given HTML file should provide help for a particular topic, such as “Printing a Map.” These help files can be opened programmatically from the HydroDesktop application as it runs. This document describes how to open a help file using C#.

# Procedure

Each help file has a unique address which can be used to open the file. To open a help file, you’ll need to know the path to the help relative to the main html help folder in HydroDesktop. For example, if I wanted to open the welcome page, I would just need “welcome.html” since that page is directly under the html help folder. If I wanted to open the page for the HydroR plugin, I would look in the extensions folder for HydroR, with a path such as “Extensions\\HydroR\\HydroR.html” (in C#).

As for the path to the main html help folder itself, there’s a static class called **LocalHelp** in the **HydroDesktop.Help** namespace that takes care of that for you. You pass the relative location of the help file you want to open to Localhelp, and it finds the help file and opens it for you with the default program for viewing HTML files.

**Note**  
At the time of this writing, local help files are stored in a folder such as C:\Program Files\CUAHSI HIS\HydroDesktop\Help\html\. However, this may change as the help system evolves, or the user may choose to install the help in a different location. By using the LocalHelp class, you don’t have to worry about these issues. The LocalHelp class will find the help folder for you and open the help file.

The following C# examples show how to open help files using the LocalHelp class. For all examples, you will need to do the following before you use the LocalHelp class:

1. In your project, add a reference to HydroDesktop.Help.dll. This DLL contains the LocalHelp class.
2. Set Copy Local for HydroDesktop.Help.dll to False.
3. Add the top of your class, add the following using statement.

using HydroDesktop.Help;

Now you’re ready to use the LocalHelp class to open help files.

It’s also a good idea to store the paths to your help files in a settings file, rather than hardcode the strings in your event handlers. Assuming you haven’t yet added a settings file to your project:

1. In the Solution Explorer, right-click the name of your project and click **Add | New Item…**
2. In the Add New Item dialog, choose **Settings File**.
3. Give the file a name such as Settings.settings and click **Add**.
4. The settings file should open automatically. If it does not, double click it in the Solution Explorer to open it.
5. Add settings to store paths to your help files, using values such as the ones below.
   1. Name: localHelpUri
   2. Type: string
   3. Scope: User
   4. Value: data-export.html

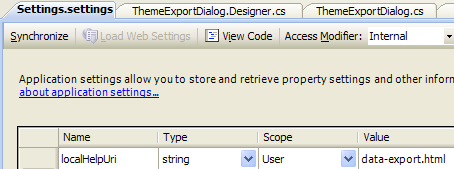


Figure Adding a Setting

You can access these settings by creating member variables in your class, as shown in the code below.

private readonly string \_localHelpUri = Properties.Settings.Default.localHelpUri;

With this preparation completed, accessing the help only requires a single line of code, typically placed in an event handler:

LocalHelp.OpenHelpFile ( \_localHelpUri );

A common way that users access help is to press the F1 key. To enable this on your form, add a handler for your form’s **HelpRequested** event. The code for adding the handler in a form called ThemeExportDialog is shown below.

this.HelpRequested += new System.Windows.Forms.HelpEventHandler ( this.ThemeExportDialog\_HelpRequested );

The code for the handler itself is shown below.

/// <summary>

/// Opens a help topic for the item in context when the user presses F1.

/// </summary>

private void ThemeExportDialog\_HelpRequested ( object sender, HelpEventArgs hlpevent )

{

LocalHelp.OpenHelpFile ( \_localHelpUri );

hlpevent.Handled = true; // Signal that we've handled the help request.

}

You could also help button to your form, along with code to open a help file when the button is clicked. There are several ways of doing this. One way is to simply add a Button control. Another is to set the form’s **HelpButton** property to **True**. When this property is set to true, a small button with a question mark appears in the caption bar to the left of the close button on the form. When the question mark is clicked, the mouse icon changes to include a question mark. Then, the user can then click a control in the form to get help on that control, without actually firing events tied to the control itself. Note that you also have to set the **MaximizeBox** and **MinimizeBox** properties to **False** in order for the question mark to show up.

If you just want the question mark to act like a regular button so that when it is clicked it just opens a help file, you can Cancel the event within your HelpButtonClicked event handler. Continuing the ThemeExportDialog example, the code below shows how to add an event handler for the HelpButtonClicked event.

this.HelpButtonClicked += new System.ComponentModel.CancelEventHandler ( this.ThemeExportDialog\_HelpButtonClicked );

The code for the handler itself is shown below. Note how the event is cancelled to that the mouse cursor does not changed to a question mark.

/// <summary>

/// Opens a help topic when the Help button is clicked.

/// </summary>

private void ThemeExportDialog\_HelpButtonClicked ( object sender, CancelEventArgs e )

{

LocalHelp.OpenHelpFile ( \_localHelpUri );

e.Cancel = true;

}