Programmer's Notepad: Help

Table of Contents

Programmer's Notepad 2

Programmer's Notepad 2

Programmer's Notepad is a free, open source, text editor with special features for coders.

You can donate to support the project.

This documentation is a work in progress. All contributions are welcome, if you know how to do something with PN that is not documented here, please consider spending a few moments of your time writing a quick bit of documentation for it - anyone can help!

How To

Table of Contents

Capture compiler output

This mini-guide will show you how to capture the output of a tool such as a compiler or linker.

Basic Settings

Open up the Options dialog from the Tools menu, and select the Tools tab. The drop-down box at the top of the tools page allows you to select a scheme that the tool should be available with. You can also select "(None - Global Tools)" if you wish the tool to be available everywhere in PN.

Click "Add" to add a new tool. A new window will pop up allowing you to enter the details of the tool. The first thing to choose is the name of the tool, something like "Compile".

Next up is the "Command" for the tool, here you select the program that the tool will run, for example "cl.exe". Don't put any parameters here, they go further down!

The "Folder" field allows you to set the current directory for when the tool runs. It is most often a good idea to set this to the folder of the current file you are working on, so put in `%d`.

Put the parameters for your tool in the "Parameters" field! You can use any of the shortcuts shown in the "Special Symbols" box at the bottom of the page, for example `%f` for the filename of the current file or `$(ProjectPath)` for the path to the current project file. If you just want to pass in the full current filename, you should use `"%d%f"`

You can choose a shortcut key to use to run your tool, just put your cursor in the shorcut box and press the key combination you want, e.g. or .

It's often useful to save the current or all files before running a tool. The Save option allows you to select what files should be saved

when running the tool.

Capturing the tool output

Select the second tab, titled "Console I/O". This tab allows you define how PN will interact with your tool.

To capture the output from your tool, simply select the "Capture Output?" option. You can then choose whether you want the output to be placed in the global output window (the main, docking output window) or in an output pane attached to the current document. Finally, you can choose whether to clear the output window before running the tool.

PN allows you to click on errors and warnings in your tool output and jump to the relevant parts of source file. It does this by parsing the output from the tool and finding text such as `"myfile.cpp:21: Error..."`. PN has built in support for several tools, to try this support leave the "Use the built-in error parser" option checked and run your tool. If you find that PN does not pick up errors from your tool then you will need a custom pattern. To see how to use custom patterns to match errors and warnings see here.

Parse unsupported tool output

This documentation is still to be written, perhaps you can help?

Modify a document using a tool

This documentation is still to be written, perhaps you can help?

Get started with projects

This documentation is still to be written, perhaps you can help?

Use a magic folder

This documentation is still to be written, perhaps you can help?

Associate a file with PN

This documentation is still to be written, perhaps you can help?

Make a filetype always use a certain scheme

This documentation is still to be written, perhaps you can help?

Switch between related .cpp and .h files

This documentation is still to be written, perhaps you can help?

Navigation

Table of Contents

Keyboard Shortcuts

This help describes some of the default keyboard shortcuts used by PN2.

The Escape Key

The escape key can often be used to get you out of dialogs - it generally represents the pressing of the cancel button in these cases. The escape button can also be used to hide any output, find in files or find bar windows that are visible. So if you run a compile which shows you some output, and then you don't want to see the output window any more then just press escape until it goes away!

Window Navigation

To go to the next window, you can use either Ctrl-Tab or Ctrl-F6. Depending on your options, this will either navigate through documents in Windows' own recent window order, or will use a Visual-Studio window stack system. Note that you can hold down the Shift key with either of these combinations to traverse in the opposite direction.

Output Window

There are two types of output window (from 0.4 onwards). The global output window (dockable) can be toggled with F8, and individual output windows can be toggled with Shift-F8.

Indent Unindent

Select a block of text over a line and use the Tab key to indent (by either tab characters or spaces depending on your settings). Use Shift-Tab to unindent.

More Shortcuts

Shortcut	Command
Alt-Enter	Show Document Properties
Alt-G	Jump To
Ctrl-/	Show Find Bar
Ctrl-Tab	Next Window
Ctrl-Shift-Tab	Previous Window
Ctrl-A	Select All
Ctrl-C	Copy
Ctrl-D	Duplicate Line
Ctrl-F	Find Dialog
Ctrl-G	Goto
Ctrl-H	Replace Dialog
Ctrl-L	Cut Line
Ctrl-N	New File
Ctrl-O	Open File
Ctrl-P	Print
Ctrl-R	Replace Dialog
Ctrl-S	Save
Ctrl-T	Transpose Lines
Ctrl-U	Lowercase
Ctrl-V	Paste
Ctrl-W	Close Window
Ctrl-X	Cut
Ctrl-Y	Redo
Ctrl-Z	Undo
Ctrl-Shift-C	Clipboard Swap
Ctrl-Shift-F	Find in Files Dialog
Ctrl-Shift-L	Delete Line

Ctrl-Shift-S	Save All
Ctrl-Shift-T	Copy Line
Ctrl-Shift-U	Uppercase
Shift-Delete	Cut
Shift-Insert	Insert
Ctrl-Insert	Copy
F2	Next Bookmark
Ctrl-F2	Set Bookmark
F3	Find Next
Ctrl-F4	Close Window
Ctrl-F6	Next Window
Ctrl-Shift-F6	Previous Window
F8	Toggle Output Window
Shift-F8	Toggle Individual Output Window

Bookmarks

PN2 supports two types of bookmarking. The first of these is "simple" bookmarks. These are the type found in Visual Studio, where you can set as many bookmarks as you like, and one key jumps from one to the next:

To set a simple bookmark: Ctrl-F2 To jump to the next simple bookmark: F2

PN2 also supports numbered bookmarks from 0 to 9. These are set and reached by pressing a key combination followed by the number of the bookmark:

To set a numbered bookmark: Ctrl-K, x To jump to a numbered bookmark: Ctrl-Q, x

where x is the number of the bookmark you wish to use. Bookmarks are shown in the margin of the text editor, the picture below shows both a simple bookmark and two numbered ones:

Search and Replace

Table of Contents

Regular Expressions

Search Patterns

Pattern	Meaning
.	Matches any character
\(	This marks the start of a region for tagging a match.
\)	This marks the end of a tagged region.
\n	Where n is 1 through 9 refers to the first through ninth tagged region when replacing. For example, if the search string was Fred\([1-9]\)XXX and the replace string was Sam\1YYY, when applied to Fred2XXX this would generate Sam2YYY.
\<	This matches the start of a word using Scintilla's definitions of words.
\>	This matches the end of a word using Scintilla's definition of words.
\x	This allows you to use a character x that would otherwise have a special meaning. For example, \[would be interpreted as [and not as the start of a character set.
[...]	This indicates a set of characters, for example, [abc] means any of the characters a, b or c. You can also use ranges, for example [a-z] for any lower case character.
[^...]	The complement of the characters in the set. For example, [^A-Za-z] means any character except an alphabetic character.
^	This matches the start of a line (unless used inside a set, see above).
$	This matches the end of a line.
*	This matches 0 or more times. For example, Sa*m matches Sm, Sam, Saam, Saaam and so on.
+	This matches 1 or more times. For example, Sa+m matches Sam, Saam, Saaam and so on.

Replacing

To use text from regular expressions in your replace string you need to use tagged expressions. To include the text of the first tagged expression in your replace text simply include `\1`. The next tagged expression is `\2` and you can use nine tagged expressions in total.

Restrictions

Support for regular expressions in PN2 is currently limited, the supported patterns and syntax are a very small subset of the powerful expressions supported by perl. The biggest restriction is that regular expressions match only within a single line, you cannot use multi-line regular expressions.

There are plans to improve this support by using the PCRE library (used elsewhere in PN2) to provide document searching. If you're interested in helping please make yourself known to the pn-discuss mailing list: PN Mailing Lists.

This documentation needs improving, perhaps you can help?

Text Clips

Table of Contents

What are Text Clips?

Text clips are text fragments that are easily inserted into your code. Programmer's Notepad ships with a variety of text clips representing common programming languages such as C or ASP.NET, or common character sets such as the symbols associated with HTML. Text Clips are easily extensible so you may modify any existing Text Clip library or extend the model to create your own Text Clip files.

Note

To create a text clips file use the Text Clip Creator application.

Click the link above to download this useful tool. Note that it requires the Microsoft .NET Framework to run.

Text Clips are stored as .clips files. These files are located in the Clips folder beneath the folder where the Programmer's Notepad executable is installed (generally \Program Files\Programmer's Notepad). The files are standard XML documents and may be manipulated using any text or XML editor.

Any properly formatted XML file with a .clips extension which is located in the Clips folder will load into the Text Clips window whenever Programmer's Notepad is started.

Manually Create a Text Clip File

There are at minimum three steps to create a custom Text Clip.

1. Use Programmer's Notepad to create an XML document.
2. Ensure that the document contains the following XML tags:

   <?xml version="1.0"?> <clips name=”Name for custom clips file”> <clip name=”Name of your first text clip”> <![CDATA[first text clip content]]> </clip> </clips>

3. Save the file and ensure that it has the .clips extension.
4. Restart Programmer's Notepad and test your new Text Clips library.

To Extend an Existing Text Clip File

To add a new clip to an existing library:

1. Open the desired Text Clip file in Programmer's Notepad.

2. Create a new line following one of the </clip> tags. Note that clips appear in the same order in which they exist in the Text Clips file.

3. Insert a new <clip> tag and use the name attribute to identify your new text clip. Example: <clip name=”my first clip”>

4. Although not mandatory, to ensure proper parsing of your clip you should embed your text clip inside a <![CDATA[ ]]> tag. Your text clip is inserted between of the inner square brackets and may include line feeds, tabs, angle brackets, and other formatting characters. Example: <![CDATA[This is my first text clip!]]>

5. Complete the text clip by including the closing </clip> tag.

6. Save the file. You must restart Programmer's Notepad before your edit will take effect.

Advanced

Table of Contents

Creating Text Clips Files

To create a text clips file use the Text Clip Creator application.

Click the link above to download this useful tool. Note that it requires the Microsoft .NET Framework to run.

Creating User Schemes

This documentation is still to be written, perhaps you can help?

Creating Project Templates

This documentation is still to be written, perhaps you can help?

config.xml

config.xml stores some advanced configuration options for PN.

User Settings Path

The <userSettings> element defines where PN should store the settings files it creates to store user options. By default this element is not defined, but it can be used to make PN portable (i.e. not store settings in the normal system "Documents and Settings" folder).

If you wanted, for example, to store settings in the "settings" folder on the memory stick you are running PN from, you can use this element like so:

```
<userSettings path="settings"/>
```

Settings Storage

By default, PN stores a number of settings in the registry for fast access. If you want to use an INI file instead, then you can use the <storeType> element to choose this:

```
<storeType value="Ini"/>
```

Extension Registration

Each extension is registered with PN using an <extension> element. This element can specify either a fully-qualified path or a relative one to point to the dll to be loaded:

```
<extension path="pypn.dll"/>
```

Command Line Arguments

The following command line arguments can be used:

Argument	Short form	Meaning
--line x	-l x	Go to line x
--col x	-c x	Go to column x
--scheme name	-s name	Use the named scheme (look in .scheme files to find the name)
--allowmulti		Override configuration and allow multiple instances
--checkassoc		Check file associations
--exit		Exit (used for spawned processes, listed for completeness)
--reset		Reset user settings and interface settings (if your PN has gone wrong!)
--safemode		Run PN without loading any extensions

Extend PN

You can write extensions for PN using C++ or using one of the scripting interfaces (like Python through PyPN).

To get started visit the web site.