RichEditLog_Demo

2.0

Generated by Doxygen 1.7.4

Thu May 26 2011 11:37:33

Contents

Chapter 1

Main Page

1.1 Introduction

This is a very simple and small project to demonstrate the use of a CRichEditCtrl for displaying formatted messages as a log. It also demonstrated auto-scrolling the control such that the last line of text is shown on the bottom.

Most examples that deal with this issue create their own class by inheriting from CRichEditCtrl. This is a different approach. Displaying formatted messages requires just one or two simple function to add colored text to the "log" and then scroll the log down as much as needed to have the last line of text shown on the bottom of the "log". I would suggest you simly copy the required function or functions into your own application instead of inheriting the whole class. If you require text formatting as well, simply modify the functions to suit your needs.

The applications shows two different approaches:

- the "naive" approach, which simply counts the number of lines being added and then scrolls down by this number.
- the "improved" approach, which is explained in detail below.

1.1.1 Naive approach

The "naive" approach simply counts the number of lines being added to the control and then scrolls down by this number. The naive approach is fine if you are sure that the user does not modify the scolling position at any time, but fails if the user changes the scrolling position.

1.1.2 Improved approach

The "improved" approach dynamically determines the number of visible lines in the control. It first srolls down to the maximum extent, therefore showing the last line of text at the top of the control. Then it scrolls back by the number of visible lines to move the last line of text to the bottom. The improved approach can be used also for controls which can be resized by the user.

1.2 Targeted audience

This demo application is intended for CRichEditCtrl beginners or intermediates having trouble implementing auto-scrolling.

1.3 Some words about CRichEditCtrl

CRichEditCtrl is a very powerful control. The MFC documentation suggests that it is easy to handle as well, but it shows some strange behaviour many people struggled with, so a demo application to get you started without having to spend hours and hours to find out about these features and workarounds should be a good thing.

1.3.1 Auto-scrolling

One of the peculiarities is that - even if you did not set one of the two ES_AUTOxSCROLL styles - CRichEditCtrl does scroll automatically if you are adding text programatically when CRichEditCtrl has the focus.

This is the main flaw that drove many people crazy when trying to implement a robust auto-scrolling feature.

CRichEditCtrl has a member function called LineScroll(nLines), which allows you to scroll up and down by `nLines` (positive: scroll down, negative: scroll up). The documentation states that when scrolling beyond the last line of text, the LineScroll() function would automatically adjust scrolling such that the last line of text would be visible at the top of the control.

So, as a simple example, if you had a fixed-size CRichEditCtrl, you would simply scroll down infinitely, to have LineScroll() adjust scrolling to display the last line of text at the top. Then you would scroll back up by the number of lines your CRichEditCtrl can show, and you're done implementing auto-scrolling. Every thing works fine, as long as CRichEditCtrl does not have the focus (which is true for most but not all dialog-based applications). However, if CRichEditCtrl has the focus, it scrolls automatically by itself. So, if you do scrolling, and the control does some scrolling, your text scrolls anywhere but not where you wanted it to be.

The simple workaround is to check if the control has the focus. If so, we don't do anything, because CRichEditCtrl does the job for us. If not, we do the scrolling as described above. See AddToLogAndScroll() for a working example.

Some people now might think, why not force CRichEditCtrl to get the focus, then add text and then give back the focus to the control which had the focus before. That would also implement auto-scrolling, because then CRichEditCtrl does the job? Good thought, but this does not always work. This may be a good approach for a dialog-based application (everthing is a control), but it does not work for document/view applications. The most robust way - as far as i know - is the one I implemented in AddToLogAndScroll().

1.3.2 Character formatting / Text color

The second thing many people stumbled across was character formatting. If you want to change the text color, it is important to turn off CFE_AUTOCOLOR to enable user-specified colors.

1.4 Version

This is version 2 of the demo application. The first version only implemented the naive way of auto-scrolling.

1.5 Thank you

Thank you for being interested in my little demo application. I hope it helps you implement what you were looking for. I would appreciate if you would rate my work at `http://www.codeproject.com/KB/edit/RichEditLog_Demo.aspx`

With kind regards,

V. Typke

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 CRichEditLog_DemoApp Class Reference

Public Member Functions

- CRichEditLog_DemoApp ()

  Constructor of the class.
- virtual BOOL InitInstance ()

  Initialization of the application.

4.1.1 Detailed Description

Author

V. Typke

Date

2005-10-20 - 2011-05-26 (last modified)

Version

1.0

4.1.2 DESCRIPTION

This is the dialog class implementation for the "RichEditLog_Demo" application.

The documentation for this class was generated from the following files:

- RichEditLog_Demo.h
- RichEditLog_Demo.cpp

4.2 CRichEditLog_DemoDlg Class Reference

Public Types

- enum { **IDD** = IDD_RICHEDITLOG_DEMO_DIALOG }

Public Member Functions

- CRichEditLog_DemoDlg (CWnd ∗pParent=NULL)

  Constructor of the class.
- int AppendToLog (CString str, COLORREF color)

  Add a string to the log window at the current position and scroll by the number of inserted lines (the naive solution for auto-scrolling).
- int AppendToLogAndScroll (CString str, COLORREF color)

  Add a string to the serial log window at the current position, then scroll to the end of the text such that the last line of the text is shown at the bottom of the CRichEditCtrl.
- int GetNumVisibleLines (CRichEditCtrl ∗pCtrl)

  Returns the number of lines that are currently visible in the client area of the given CRichEditCtrl.

Protected Member Functions

- virtual void DoDataExchange (CDataExchange ∗pDX)

  Called by the framework to exchange and validate dialog data.
- virtual BOOL OnInitDialog ()

  Initialize the dialog.
- afx_msg void OnPaint ()

  The framework calls this member function when Windows or an application makes a request to repaint a portion of an application's window.
- afx_msg HCURSOR OnQueryDragIcon ()

  The framework calls this member function by a minimized (iconic) window that does not have an icon defined for its class.
- afx_msg void OnBnClickedInsertBlack ()

  This function is called when the user clicks on the corresponding button.
- afx_msg void OnBnClickedInsertRed ()

  This function is called when the user clicks on the corresponding button.
- afx_msg void OnBnClickedInsertGreen ()

  This function is called when the user clicks on the corresponding button.
- afx_msg void OnBnClickedInsertBlackEx ()

  This function is called when the user clicks on the corresponding button.
- afx_msg void OnBnClickedInsertRedEx ()

  This function is called when the user clicks on the corresponding button.
- afx_msg void OnBnClickedInsertGreenEx ()

  This function is called when the user clicks on the corresponding button.

Protected Attributes

- HICON **m_hIcon**
- CRichEditCtrl **m_ctrlLog**

4.2.1 Detailed Description

Author

V. Typke

Date

2005-10-20 - 2011-05-26 (last modified)

Version

1.0

4.2.2 DESCRIPTION

This is the dialog class implementation for the "RichEditLog_Demo"-Dialog.

4.2.3 Constructor & Destructor Documentation

4.2.3.1 CRichEditLog_DemoDlg::CRichEditLog_DemoDlg (CWnd * *pParent* = `NULL`)

Constructor of the class.

Parameters

`in`	*pParent*	Pointer to parent window. NULL by default.

4.2.4 Member Function Documentation

4.2.4.1 int CRichEditLog_DemoDlg::AppendToLog (CString *str*, COLORREF *color*)

Add a string to the log window at the current position and scroll by the number of inserted lines (the naive solution for auto-scrolling).

The string is added to the log starting at the current position, i.e. without starting a new line. Then the control scrolls down by the number of lines inserted. The string is displayed in the specified text color. The string may be a multiline string using carriage return/line feed (i.e. newline) characters to indicate a line breaks.

The scrolling mechanism used here is kind of naive, because it assumes that the user did not touch the scroll bars and that the scroll position is always the end of the text. However, this is not the general case. In general, we need to assume that the current scrolling position is unkown. A solution for that is shown in the AppendToLogAndScroll()

method.

Parameters

`in`	*str*	The string to add to the message log.
`in`	*color*	The text color of the string. You may use the RGB(r,g,b) macro to specify the color byte-wise.

Returns

An integer indicating sucess or failure:

- 0, if the function succeeded.
- (-1), if the function failed. (This function always returns 0, because no parameter or failure checking is done.)

Remarks

Support for adding multiline strings requires the ES_MULTILINE style to be set. If you are not using the Visual Studio Wizards but create the control indirectly using the Create() method, you should use the following style: WS_CHILD|WS_-VSCROLL|WS_HSCROLL|ES_MULTILINE|ES_READONLY.

See also

AppendToLogAndScroll()

4.2.4.2 int CRichEditLog_DemoDlg::AppendToLogAndScroll (CString *str*, COLORREF *color*)

Add a string to the serial log window at the current position, then scroll to the end of the text such that the last line of the text is shown at the bottom of the CRichEditCtrl.

The string is added to the message log starting at the current position, i.e. without starting a new line. Then the control scrolls down to show as much text as possible, including the last line of text at the very bottom. The string is displayed in the specified text color. The string may be a multiline string using carriage return/line feed (i.e. newline) characters to indicate a line breaks.

Parameters

`in`	*str*	The string to add to the message log.
`in`	*color*	The text color of the string. You may use the RGB(r,g,b) macro to specify the color byte-wise.

Returns

An integer indicating sucess or failure:

- 0, if the function succeeded.
- (-1), if the function failed. (This function always returns 0, because no parameter or failure checking is done.)

Remarks

> The automatic scrolling function would be easy, if the MFC documentation was correct. Unfortunetely, it is not as trivial as one might think. If the CRichEditCtrl has the focus, it scrolls automatically if you insert text programatically. If it does not have the focus, it does not scroll automatically, so in that case you can use the LineScroll() method and you get the results you would expect when reading the MFC docs. This is true even if ES_AUTOxSCROLL style is NOT set.

So the point is to check in the AppendToLogAndScroll() method if the affected CRichEditCtrl has the focus. If so, we must not call LineScroll(). If not, it is safe to call LineSroll() to first scroll to the very end, which means that the last line of text is shown at the top of the CRichEditCtrl. Then we call LineScroll() a second time, this time scrolling back by the number of visible lines. This leads to having the last line of the text being displayed at the bottom of CRichEditCtrl.

Please note that in this sample application, the CRichEditCtrl never has the focus, because we always have to click a button in order to insert text. However, if you are using the code in an application not based on a dialog and that fills up the control where the user could have set focus to the control first, this method would fail to scroll correctly without checking the focus. I used this code in an MDI application, and there the control claims to have the focus if I click into the control before clicking a menu command (whatever the reason might be why in that case the focus is not lost to the menu command).

Please note that the code is written for maximum comprehension / good readability, not for code or execution efficiency.

4.2.4.3 void CRichEditLog_DemoDlg::DoDataExchange (CDataExchange ∗ *pDX*) [protected, virtual]

Called by the framework to exchange and validate dialog data.

Parameters

in,out	*pDX*	A pointer to a CDataExchange object.

Remarks

> Never call this function directly. It is called by the UpdateData member function. Call UpdateData to initialize a dialog box's controls or retrieve data from a dialog box.

4.2.4.4 int CRichEditLog_DemoDlg::GetNumVisibleLines (CRichEditCtrl ∗ *pCtrl*)

Returns the number of lines that are currently visible in the client area of the given CRichEditCtrl.

Parameters

in	*pCtrl*	Pointer to the CRichEditCtrl object to query.

Returns

The number of currently visible lines.

Remarks

The code is written for best comprehension / readability, not for code or execution efficiency.

4.2.4.5 void CRichEditLog_DemoDlg::OnBnClickedInsertBlack ( ) [protected]

This function is called when the user clicks on the corresponding button.

This function calls AppendToLog() and thus inserts the text "This is black text.\\n" into the CRichEditCtrl using black text color. AppendToLog() inserts the text (and the blank new line generated for the 'newline' character) and then scrolls down by two lines from whatever the current scrolling position is. This is the naive auto-scrolling implementation, because it does not care about the current scrolling position.

4.2.4.6 void CRichEditLog_DemoDlg::OnBnClickedInsertBlackEx ( ) [protected]

This function is called when the user clicks on the corresponding button.

This function calls AppendToLogAndScroll() and thus inserts the text "This is black text of improved version.\\n" into the CRichEditCtrl using black text color. AppendToLogAndScroll() inserts the text (and the blank new line generated for the 'newline' character) and then scrolls down such that the last line of text of the CRichEditCtrl will be shown on the bottom of the control. This happens independently of the current scrolling position. Note that there will be a blank line at the bottom, due to the 'newline' character at the end of the inserted text. Note that there can be an additional blank line at the bottom, if the current scrolling position is such that there are only partially visible lines. The helper function GetNumVisibleLines() only returns the number of fully visible lines, hence the scrolling back may vary by one line.

4.2.4.7 void CRichEditLog_DemoDlg::OnBnClickedInsertGreen ( ) [protected]

This function is called when the user clicks on the corresponding button.

This function calls AppendToLog() and thus inserts the text "This is dark green text.\\nThis is the second line of green text with a line break.\\r\\n" into the CRichEditCtrl using green text color. AppendToLog() inserts the text and then scrolls down by three lines from whatever the current scrolling position is. This is the naive auto-scrolling implementation, because it does not care about the current scrolling position.

4.2.4.8 void CRichEditLog_DemoDlg::OnBnClickedInsertGreenEx ( ) [protected]

This function is called when the user clicks on the corresponding button.

This function calls AppendToLogAndScroll() and thus inserts the text "This is darker green text of improved version.\\nThis is its second line\\r\\nand this the third line ending with a line break.\\n" into the CRichEditCtrl using black text color. AppendToLogAndScroll() inserts the text (and the blank new line generated for the 'newline' character) and then scrolls down such that the last line of text of the CRichEditCtrl will be shown on the bottom of the control. This happens independently of the current scrolling position. Note that there will be a blank line at the bottom, due to the 'newline' character at the end of the inserted text. Note that there can be an additional blank line at the bottom, if the current scrolling position is such that there are only partially visible lines. The helper function GetNumVisibleLines() only returns the number of fully visible lines, hence the scrolling back may vary by one line.

4.2.4.9 void CRichEditLog_DemoDlg::OnBnClickedInsertRed ( ) [protected]

This function is called when the user clicks on the corresponding button.

This function calls AppendToLog() and thus inserts the text "This is red text.\\r\\nThis is the second line of red text without a line break." into the CRichEditCtrl using red text color. AppendToLog() inserts the text and then scrolls down by two lines from whatever the current scrolling position is. This is the naive auto-scrolling implementation, because it does not care about the current scrolling position.

4.2.4.10 void CRichEditLog_DemoDlg::OnBnClickedInsertRedEx ( ) [protected]

This function is called when the user clicks on the corresponding button.

This function calls AppendToLogAndScroll() and thus inserts the text "This is darker red text of improved version.\\r\\nThis is its second line without a line break." into the CRichEditCtrl using red text color. AppendToLogAndScroll() inserts the text and then scrolls down such that the last line of text of the CRichEditCtrl will be shown on the bottom of the control. This happens independently of the current scrolling position. Note that there can be a blank line at the bottom, if the current scrolling position is such that there are only partially visible lines. The helper function GetNumVisibleLines() only returns the number of fully visible lines, hence the scrolling back may vary by one line.

4.2.4.11 BOOL CRichEditLog_DemoDlg::OnInitDialog ( ) [protected, virtual]

Initialize the dialog.

Called by the framework in response to the WM_INITDIALOG message. This message is sent to the dialog box during the Create, CreateIndirect, or DoModal calls, which occur immediately before the dialog box is displayed.

Returns

A boolean indicating if this function set the focus to a control other than the default control.

- TRUE, if the focus was not set to another control.
- FALSE, if the focus was set to another control. This specific implementation of OnInitDialog always returns TRUE.

4.2.4.12 void CRichEditLog_DemoDlg::OnPaint ( ) [protected]

The framework calls this member function when Windows or an application makes a request to repaint a portion of an application's window.

This function is required if the dialog has a "Minimize" button. If so, its icon is drawn by this function. For applications using the document/view framework, the icon is drawn automatically.

The documentation for this class was generated from the following files:

- RichEditLog_DemoDlg.h
- RichEditLog_DemoDlg.cpp

Chapter 5

File Documentation

5.1 RichEditLog_Demo.cpp File Reference

Contains the class implementation of CRichEditLog_DemoApp.

```
#include "stdafx.h"
#include "RichEditLog_Demo.h"
#include "RichEditLog_DemoDlg.h"
```

Variables

- CRichEditLog_DemoApp theApp

  The one and only application object.

5.1.1 Detailed Description

Contains the class implementation of CRichEditLog_DemoApp. This file contains the class implementations of CRichEditLog_DemoApp, which is the main application.

This code was written and compiled with Microsoft Visual Studio .NET 2003. You should be able to compile this code with any later version of Visual Studio or Visual C++.

5.1.2 LICENSE

This software is licensed under the terms and condfitions of the Code Project Open License (CPOL), which you can find under `http://www.codeproject.com/info/cpol10.aspx` (if not, go to `http://www.codeproject.com` and search for "The Code Project Open License").

This program is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

5.2 RichEditLog_DemoDlg.cpp File Reference

Contains the class implementation of CRichEditLog_DemoDlg.

```
#include "stdafx.h"
#include "RichEditLog_Demo.h"
#include "RichEditLog_DemoDlg.h"
```

5.2.1 Detailed Description

Contains the class implementation of CRichEditLog_DemoDlg. This file contains the class implementations of CRichEditLog_DemoDlg, which is the main dialog.

This code was written and compiled with Microsoft Visual Studio .NET 2003. You should be able to compile this code with any later version of Visual Studio or Visual C++.

5.2.2 LICENSE

This software is licensed under the terms and condfitions of the Code Project Open License (CPOL), which you can find under `http://www.codeproject.com/info/cpol10.aspx` (if not, go to `http://www.codeproject.com` and search for "The Code Project Open License").

This program is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Index