.NET 1.0 Windows 2003 Win32 .NET 1.1 .NET 3.0 Visual Studio 2005 Windows 2000 Windows XP .NET 2.0 .NET 3.5 C# 2.0 C# 3.0 Intermediate Dev Visual Studio Windows .NET C#

Advanced message box for .NET

Sergiy Stoyan

3.63/5 (20 votes)

Mar 17, 2008 CPOL 5 min read 125815 1816

A replacement of .NET MessageBox class that provides additional features and improvements like possibility to show any number of buttons, 'apply-my-answer-by-default' checkbox, intelligence layout and more.

Download MessageForm - 135.3 KB
Download latest version from SourceForge

Introduction

Some applications need a more advanced message box than `MessageBox` class provided by .NET. This article presents a .NET library called `MessageForm` that can be used instead of .NET `MessageBox` getting more features and improvements.

General Features of MessageForm Comparing with MessageBox

Features	MessageForm	MessageBox
can contain any number of buttons	+	- (provides only up 3 predefined buttons)
text and color of the buttons are specified by parameters	+	-
can display 'apply-my-answer-by-default' checkbox so that user can make his/her answer applied by default in the future	+	-
allows selecting and copying message; it is handy thing when you want to get path or error string from message box	+	-
window's content can be seen independently on its size	+ (`MessageForm`'s size is always less than the display; if its content is too large then the scroll bar is provided)	- (if message is too long, it goes out of the screen so that you cannot read it all)
allows setting icon in the window's title bar	+ (by default it displays icon of the hosting application)	-
thread safe	+ (As an option, `Message` class also provides thread synchronization so that messages of different threads are showed strongly in turn)	+
brings message box to the top of windows on the desktop; it is important feature for alerts	+	+/- (if the application showing message box is windowless, the message box can be from the beginnig "lost" under the rest windows on the desktop)
accepts `Image` type as well as `Icon` type when setting icon beside message	+	-
can be fitted/enhanced further	+ (open source)	-

Description

The `MessageForm` library consists of 2 classes:

The MessageForm library consists of 2 classes:

- MessageForm, that is, in fact, the advanced message box implementation;
- Message, a wrapper for MessageForm

MessageForm Class

MessageForm class implements the message box window basing on System.Windows.Forms.Form. Usually you do not want to use it because of using Message class instead. On the other hand, MessageForm provides more flexible settings than Message does.

Message Class

The goal of Message is to make use of the advanced message box in your code as simple as possible. It provides a collection of predefined MessageForm instances most frequently used. That means you have usually to work with Message class only. If its predefined methods do not meet your needs, you can use MessageForm directly or enhance Message with your own ones.

Message is a thread-safe class so that you can use it from within several threads with no collision. The Message.ShowMessagesInTurn parameter defines how message boxes will be showed in a multithread environment. If it is true then message boxes are showed one after another. If it is false, Message shows message boxes invoked from different threads, simultaneously (that's like .NET MessageBox does).

The Message class has the following default settings:

- the icon dispalyed in the title bar of a message window is an icon of the hosting application;
- the text in the title bar is the hosting application's name;
- buttons are colored with different colors. Often it is handy as it helps a user faster select a desired answer. Only the OK-button message box has uncolored button;
- message window is a top-most and top-level as it is usually needed for alert messages;
- message boxes invoked from different threads are showed in turn one after another;

These settings are the respective attributes of Message and so can be changed from your code that uses MessageForm or even directly within the MessageForm project. For example, coloration of the buttons can be toggled off by the following code:

C#

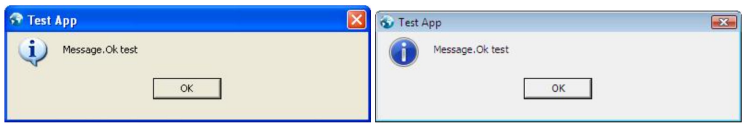
```
//toggle off coloring buttons
Cliver.Message.ButtonColors = null;
```

Usage of MessageForm

Below are several examples of using Message and MessageForm classes. (More examples you can find in Test project within MessageForm solution.) The respective code follows the image.

Using Message Class

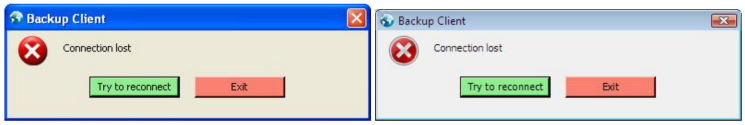
Message box with OK button (XP and Vista style):



C#

```
Cliver.Message.Inform("Message.Ok test");
```

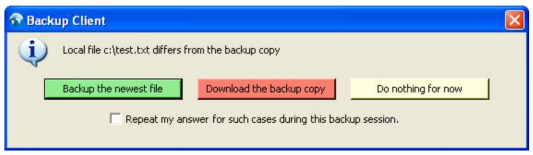
Message box with 2 custom buttons (XP and Vista style):



C#

```
//returns clicked button index
int a = Cliver.Message.Show(SystemIcons.Error, "Connection lost", 0,
    "Try to reconnect", "Exit");
```

Message box with 3 custom buttons and checkbox:



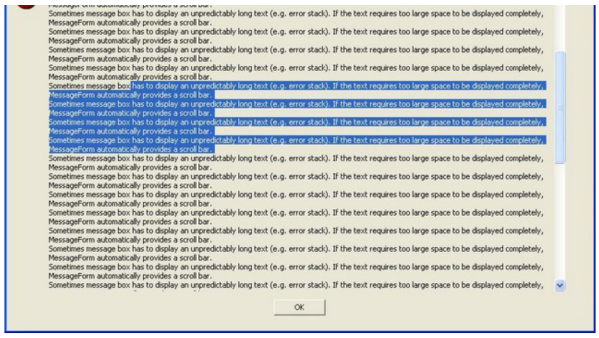
C#

```
//Set message box caption once and forever
Cliver.Message.Caption = "Backup Application";

//returned silent box state
bool r;
//returned clicked button index
int a;

//show message box with 3 buttons
a = Cliver.Message.Show(SystemIcons.Information, out r,
               @"Local file c:\test.txt differs from the backup copy", 0,
               "Backup the newest file",
               "Download the backup copy",
               "Do nothing for now"
               );
```

What if message box has to display unpredictably long text, for instance, an error stack? Sometimes the message can be so long that a .NET MessageBox will go out of the screen making the end of the message invisible. Here is an example how MessageForm treats such 'unsafe' cases:



Using MessageForm Directly

Message box with 6 buttons, checkbox and custom message icon. Notice that `MessageForm` can accept `Image` type in place of `Icon` type.

How it looks in the classic style on screen with less resolution:

C# Shrink ▲

```
Cliver.MessageForm mf = new MessageForm(
    "Using MessageForm directly",//caption
    //message
    "Copying files from: 'c:\\test' to: 'c:\\test2'\nFile 'test.txt' already exists.",
    new Color[6] {
        Color.LightCoral,
        Color.LightYellow,
        Color.Empty,
        Color.LightGreen,
        Color.LightBlue,
        Color.Empty,
    },//button colors
    new string[6] {
        "Overwrite",
        "Overwrite if older",
        "Skip",
        "Rename",
        "Append",
        "Cancel"
    },//array of answers
    1,//default button
    "Apply to all",//silent box text
    new Bitmap("../../copying.jpg")//message icon, can be set from image
);

//set icon in the capture
mf.Icon = new Icon("../../computers308.ico");

//show message form
mf.GetAnswer();

//get state of silent checkbox
bool silent = mf.Silent;
```

Modifying MessageForm

The MessageForm library is written in pure C#. It builds its own window that's not a wrapper of the `MessageBox` class. It inherits `System.Windows.Forms.Form` so you can easily and flexibly modify it as you want by changing its code.

Also be aware that while automatic arranging controls in `MessageForm` the following values are not changed from their initial values:

- minimal size of the window (it can only grow);
- the message icon's location;
- the message label's Y value;
- span between window's left edge and the first button;

Keeping that in your mind, you can tune the look of `MessageForm` in Visual Studio window design mode.

Also, pay attention to several attributes in the `MessageFrom` class that defines its layout as well.

Using the Code

In the attached code you can find:

- library project `MessageForm` that contains the `MessageForm` and `Message` classes. These classes can be compiled as a DLL or be added to your code.
- `Test` project with the usage examples.

The last version of `MessageForm` can be found at SourceForge

Be happy!