

Forum:[Hack Forums](#)

Thread: [NativeClipboard.cs](#) | [Read/Write clipboard text](#) | [64 and 32-bit](#) | [Console or GUI](#)

gigajew - June 27, 2018, 8:40 pm

This is my sweet clipboard class. Enjoy it! Supports both 32- and 64-bit. And it also supports GUI or no GUI.

0xDEAD10CC - June 30, 2018, 11:45 am

Code internal sealed class NativeClipboard

...

public NativeClipboard() : base()

Lol. Also, I think I've seen some implementations of this on HF before. You really should be taking more care with unmanaged resources and functions to free those resources though. try/finally for example. No idea why you need a redundant message loop too, and no reason this couldn't have been a static class, and GlobalAlloc should probably specify the GHND flag rather than relying on memory to be zero'd.

kevy42 - June 30, 2018, 11:45 am

iv been trying to do this myself for ages, finally got a crude version going than saw this. looks 100x better than mine XD, nice share

Washi - June 30, 2018, 1:16 pm

Just a question: What advantages brings your class that the [Clipboard](#) class built into the .NET framework does not provide?

0xDEAD10CC - June 30, 2018, 1:24 pm

(Jun 30, 2018 01:16 PM)Washi Wrote: [\[CITE\]](#) Just a question: What advantages brings your class that the [Clipboard](#) class built into the .NET framework does not provide?

One of them:

Quote:The Clipboard class can only be used in threads set to single thread apartment (STA) mode. To use this class, ensure that your Main method is marked with the STAThreadAttribute attribute.

I wouldn't know whether to call it an "advantage" but it's not a limitation at least. Runs fine in [MTAThread] context.

Here's some of my modifications to the code though:

Code internal static class NativeClipboard

```
{
    private static readonly int CF_UNICODETEXT = 13;

    [DllImport("user32.dll", SetLastError = true)]
    [return: MarshalAs(UnmanagedType.Bool)]
    private static extern bool OpenClipboard([MarshalAs(UnmanagedType.I4)] int window);

    [DllImport("user32.dll")]
    [return: MarshalAs(UnmanagedType.Bool)]
```

```

private static extern bool CloseClipboard();

[DllImport("user32.dll")]
private static extern IntPtr SetClipboardData([MarshalAs(UnmanagedType.I4)] int format,
IntPtr data);

[DllImport("user32.dll")]
[return: MarshalAs(UnmanagedType.LPWStr)]
private static extern string GetClipboardData([MarshalAs(UnmanagedType.I4)] int format);

[DllImport("kernel32.dll")]
private static extern IntPtr GlobalAlloc([MarshalAs(UnmanagedType.I4)] int flags,
[MarshalAs(UnmanagedType.I4)] int size);

[DllImport("kernel32.dll")]
private static extern IntPtr GlobalFree(IntPtr hGlobal);

[DllImport("kernel32.dll")]
private static extern IntPtr GlobalLock(IntPtr hGlobal);

[DllImport("kernel32.dll")]
private static extern void GlobalUnlock(IntPtr hGlobal);

[return: MarshalAs(UnmanagedType.LPWStr)]
public static string GetUnicodeClipboardText()
{
    if (!OpenClipboard(0))
        throw new Win32Exception(Marshal.GetLastWin32Error());

    try
    {
        return GetClipboardData(CF_UNICODETEXT);
    }
    finally
    {
        CloseClipboard();
    }
}

public static bool SetUnicodeClipboardText([MarshalAs(UnmanagedType.LPWStr)] string text)
{
    IntPtr hAlloc = IntPtr.Zero;

    if (!OpenClipboard(0))
        throw new Win32Exception(Marshal.GetLastWin32Error());

    try
    {
        const int GHND = 0x0042;
        byte[] buffer = Encoding.Unicode.GetBytes(text);
        hAlloc = GlobalAlloc(GHND, buffer.Length + 1);
        if (hAlloc != IntPtr.Zero)
        {

```

```

        Marshal.Copy(buffer, 0, GlobalLock(hAlloc), buffer.Length);
        GlobalUnlock(hAlloc);
        return SetClipboardData(CF_UNICODETEXT, hAlloc) != IntPtr.Zero;
    }
}
finally
{
    if (hAlloc != IntPtr.Zero)
        GlobalFree(hAlloc);

    CloseClipboard();
}

return false;
}
}

```

I don't see how the message loop was necessary, but even if it was you could have just thrown the call in the static constructor...

int - July 2, 2018, 6:54 am

(Jun 30, 2018 01:16 PM)Washi Wrote: [\[CITE\]](#) Just a question: What advantages brings your class that the [Clipboard](#) class built into the .NET framework does not provide? was just thinking this.

FryPan - July 2, 2018, 4:07 pm

(Jun 30, 2018 01:24 PM)0xDEAD10CC Wrote: [\[CITE\]](#) One of them:

I wouldn't know whether to call it an "advantage" but it's not a limitation at least. Runs fine in [MTAThread] context.

Who wants to access clipboard using threads? This is a rare occasion

0xDEAD10CC - July 2, 2018, 4:13 pm

(Jul 2, 2018 04:07 PM)FryPan Wrote: [\[CITE\]](#) Who wants to access clipboard using threads? This is a rare occasion

I was just pointing it out, native calls like in the class above are probably going to be faster than the Clipboard class anyways because the .NET framework internally does a lot more checking and stuff behind the scenes than doing what you see above. Still a lot that could be fixed in that class though.

I just finished running a small benchmark against 1000 calls and it's a difference of a factor of 10x... ~30ms vs ~3ms (first run was 100x difference). Not saying you need performance here either, but using the Clipboard class also requires that you reference System.Windows.Forms.dll.

Washi - July 3, 2018, 7:27 am

(Jun 30, 2018 01:24 PM)0xDEAD10CC Wrote: [\[CITE\]](#) I wouldn't know whether to call it an "advantage" but it's not a limitation at least. Runs fine in [MTAThread] context.

Ah okay, the code being independent of the apartment state might actually be a factor I guess.

But yea as you said, I wonder whether this, the speed or the dependency on System.Windows.Forms.dll are that much of an issue to be honest, especially since WinForms is part of the standard .NET framework libraries.

0xDEAD10CC - July 4, 2018, 5:44 pm

(Jul 3, 2018 07:27 AM)Washi Wrote: [\[CITE\]](#) Ah okay, the code being independent of the apartment

state might actually be a factor I guess.

But yea as you said, I wonder whether this, the speed or the dependency on System.Windows.Forms.dll are that much of an issue to be honest, especially since WinForms is part of the standard .NET framework libraries.

I have no side on this, perhaps if I was doing lower level clipboard work I'd use P/Invokes, but for the basic operations there probably isn't much difference.

Interallahu - July 4, 2018, 6:02 pm

(Jul 2, 2018 04:07 PM)FryPan Wrote: [\[CITE\]](#) Who wants to access clipboard using threads? This is a rare occasion

You can use delegates with threads.. or Invoke

Plus, The P/Invoking hes doing requires the message loop, which is usually done in a seperate thread

gigajew - July 4, 2018, 7:33 pm

Revamped code, removed an error dealing with memory and removed some unused imports, added unsafe context

interestingcode - July 5, 2018, 3:51 pm

is this coded in c# or c++?

nvm lol saw just now that it was posted in c# topic