If you are going to call Marshal.GetLastWin32Error, the function whose error you're retrieving had better be the one called most recently

devblogs.microsoft.com/oldnewthing/20150819-00

August 19, 2015

Raymond Chen

Even if you remember to set `SetLastError=true` in your p/invoke signature, you still have to be careful with `Marshal.GetLastWin32Error` because there is only one last-error code, and it gets overwritten each time.

So let's try this program:

```
using System;
using System.Runtime.InteropServices;

class Program
{
  [DllImport("user32.dll", SetLastError=true)]
  public static extern bool OpenIcon(IntPtr hwnd);

  public static void Main()
  {
    // Intentionally pass an invalid parameter.
    var result = OpenIcon(IntPtr.Zero);
    Console.WriteLine("result: {0}", result);
    Console.WriteLine("last error = {0}",
                      Marshal.GetLastWin32Error());
  }
}
```

The expectation is that the call to `OpenIcon` will fail, and the error code will be some form of invalid parameter.

But when you run the program, it prints this:

```
result: False
last error = 0
```

Zero?

Zero means “No error”. But the function failed. Where’s our error code? We printed the result immediately after calling `OpenIcon`. We didn’t call any other p/invoke functions. The last-error code should still be there.

Oh wait, printing the result to the screen involves a function call.

That function call might itself do a p/invoke!

We have to call `Marshal.GetLastWin32Error` immediately after calling `OpenIcon`. Nothing else can sneak in between.

```
using System;
using System.Runtime.InteropServices;

class Program
{
  [DllImport("user32.dll", SetLastError=true)]
  public static extern bool OpenIcon(IntPtr hwnd);

  public static void Main()
  {
    // Intentionally pass an invalid parameter.
    var result = OpenIcon(IntPtr.Zero);
    var lastError = Marshal.GetLastWin32Error();
    Console.WriteLine("result: {0}", result);
    Console.WriteLine("last error = {0}",
                      lstError);
  }
}
```

Okay, now the program reports the error code as 1400: “Invalid window handle.”

This one was pretty straightforward, because the function call that modified the last-error code was right there in front of us. But there are other ways that code can run which are more subtle.

- If you retrieve a property, the property retrieval may involve a p/invoke.
- If you access a class that has a static constructor, the static constructor will secretly run if this is the first time the class is used.

Raymond Chen

Follow