Intercept WinForms Events

You can examine the events WinForms raises before it sends them to subscribers.

by Juval Löwy

Technology Toolbox

- ❑ VB.NET
- ☑ C#
- ❑ SQL Server 2000
- ❑ ASP.NET
- ❑ XML
- ☑ VB6
- ☑ Other: Unmanaged C++

Go Online!

Use these Locator+ codes at **www.visualstudiomagazine.com** to go directly to these related resources.

Download
VS0311QA Download Listing1.

Discuss
VS0311QA_D Discuss this article in the C# forum.

Read More
VS0311QA_T Read this article online.

VS0303QA_T Q&A, "Convert Integers to Byte Arrays," by Mattias Sjögren and Juval Löwy

VS0305QA_T Q&A, "Publish Events Defensively," by Juval Löwy

VS0204JL_T "Tame .NET Events" by Juval Löwy

Q. Intercept WinForms Events

I need to do some precall processing and examine the events WinForms raises before it dispatches them to the subscribers. How do I do this?

A.

WinForms converts Windows messages to delegate-based events. Every control or form has a list of delegate-based events, each of which corresponds to a Windows message. You can place your event- or message-handling code at any point. One end of the spectrum is the raw message level (read the "Handle Windows Messages" topic in the March 2003 Q&A column, "Convert Integers to Byte Arrays"; see the Go Online box for details). At the other end, you can provide a method to handle the event; the visual designer does this by default. The in-between option is to place the code after the raw message processing, but before WinForms converts the message to the event. The Form class's WndProc method contains a switch statement that calls individual protected methods to handle the message. The message-handling method raises the corresponding delegate-based event. For example, consider your form's Load event. WinForms raises this event in response to the WM_SHOWWINDOW Windows message.

The Form class converts Windows messages to delegate-based events (see Listing 1). Listing 1 shows the Form class's implementation in pseudocode. The actual implementation details are somewhat different (Form uses an event list instead of individual event members, but the event accessors present a façade as if these were individual events), yet the essence of how Form converts Windows messages to delegate-based events is the same. The Form class's WndProc method calls the OnLoad method in response to the Windows message. The OnLoad method handles the message, then raises the delegate-based event associated with the message by invoking the Load event. Form includes similar methods—such as OnActivate, OnClick, and so on—for every event type. You must override the matching method if you want to intercept the message before it's converted to a delegate-based event:

```
public class MyForm : Form
{
   protected override void
      OnLoad(EventArgs e)
   {
      /* Do some event processing,
      then: */
      base.OnLoad(e);
   }
}
```

Be sure to call the base-class implementation so the base class raises the delegate-based event.

Q. VB6 and Unmanaged C++ Support to Diverge

What are Microsoft's plans regarding VB6? When (if at all) will Microsoft stop supporting it? What about unmanaged C++?

A.

Microsoft deprecated VB6 with the advent of .NET. Microsoft would like every VB6 developer to adopt .NET and undertake all new development efforts in .NET—ideally. In reality, the migration process will take years, and some companies and developers will never make the move. Microsoft needs to balance expenditures it makes to support VB6 during this transitional time with a commitment to long-term support of its customers—a balance that any reputable vendor maintains regarding its obsolete products.

Microsoft plans three phases for VB6: Mainstream, Extended, and Non-Supported (see Additional Resources). The Mainstream phase is in effect today: Microsoft plans no new releases or new features, but you get full technical support, including free phone support and online incident reports. Microsoft will release hot fixes as needed during this phase on topics such as security. The Mainstream phase will end March 31, 2005. The Extended phase is designed to support the existing VB6 user base and code but reduce incentives for you to stick with the old technology. The main difference between the Mainstream and the Extended phases is that free phone support and incident reports no longer will be available. The Extended phase will end in March 2008. Then, VB6 will move to the Non-Supported phase, when Microsoft will cease all support for the product.

Microsoft has entirely different plans regarding unmanaged C++ support. Microsoft will support its C++ line of products, such as ATL and MFC, fully for the foreseeable future. It also will advance and extend these products as major improvements to version 7 of both MFC and ATL indicate. Future features probably will target improvement of integration with .NET and making the development experience in unmanaged C++ as close as possible to that of .NET.

Conceptually, Microsoft treats managed C++ and unmanaged C++ as the same line of products because it views the C++ capabilities as complementary to .NET. Developers who want ultimate power and control will use C++. As long as .NET runs on top of Windows, the only way to develop products such as hardware drivers is to use VS.NET and C++. VSM

C#, Pcode • Convert Windows Messages to Delegate-Based Events

```
public class Form : ContainerControl
{
   const int WM_SHOWWINDOW = 0x18;
   public event EventHandler Load;
   //Other message and event definitions

   protected override void WndProc(ref Message m)
   {
      switch(m.Msg)
      {
         case WM_SHOWWINDOW:
         {
            OnLoad(EventArgs.Empty);
            break;
         }
         //other cases
      }
      base.WndProc(ref m);
   }
   protected virtual void OnLoad(EventArgs e)
   {
      /* process the event: show window, etc,
      then:  */
      if(Load != null)
      {
         Load(this,e);
      }
   }
}
```

Listing 1 The Form class's WndProc method calls methods in response to Windows messages. These methods handle the event, then raise the delegate-based event associated with this message.

Juval Löwy is a software architect and the principal of IDesign, a consulting and training company focused on .NET design and .NET migration. Juval is Microsoft's regional director for the Silicon Valley, working with Microsoft on helping the industry adopt .NET. His latest book is *Programming .NET Components* (O'Reilly & Associates). Juval speaks frequently at software-development conferences. Contact him at www.idesign.net.

Additional Resources

"Product Family Life-Cycle Guidelines for Visual Basic 6.0": http://msdn.microsoft.com/vbasic/support/vb6.asp

AMYUNI Technologies

We personnalize your needs!

NEW

Document Converter Suite
PDF-RTF-HTML-Excel-JPEG

Amyuni's new Document Converter gives you more document conversion power in a more streamlined product. Our flagship PDFConverter technology has been applied to RTF, HTML, Excel® and JPEG formats and integrated them in one global Amyuni printer driver.

www.amyuni.com

info: sales@amyuni.com

Americas	**Europe**
Toll Free: 1-866-926-9864	Sales: (+33) 1 30 61 07 97
Support: (514) 868-9227	Support: (+33) 1 30 61 07 98