Foreword

The late 1990s brought us the revolution of the Internet. After 15 years of moving from a server-based model of computing to a client/server-based model, the pendulum swung back swiftly toward the server with the rapid growth of Web pages, HTML, and server-based applications.

There is much to like about Web applications. Designers like them, because they have lots of great ways to apply nice-looking style sheets and layouts. Companies like Web applications, because they do away with all the expensive and risky aspects of deploying client applications—all that has to be done is to install the application on a Web server. There is no risk of breaking other applications and no need to physically install the software on every machine in the organization. And for document viewing, HTML is a relatively easy language to learn, so it allows many people to do some software development with few prior skills.

But not everything is perfect. Large-scale Web applications are difficult to write and manage. There are differences among browsers. There aren't very good tools for debugging and developments. The applications don't take advantage of all the power on the client machines: hard drives, video cards, and CPUs. And most important, the user interfaces are generally only well-suited to the most basic data entry. If you need a real-time display or an advanced visualization, things get very difficult.

In early 2002, Windows Forms was released as part of the Microsoft .NET Framework, version 1.0. This changed the landscape in two fundamental ways. First, it gave programmers a consistent, approachable API and tool set with which to build very sophisticated applications for Microsoft Windows without having to know the Win32 SDK forward and backward. And second, the .NET Framework and common language runtime (CLR) allowed client applications to be deployed via a Web server. Once you got the .NET Framework installed on the client machines, you could have true zero-cost or no-touch deployment.

In conjunction with the advantages of Windows applications with .NET 1.0, organizations were beginning to recognize the shortcomings of Web applications in certain scenarios. As a result, they started to deploy client applications once again.

With the release of Version 2.0 of the Microsoft .NET Framework, even more client momentum is building. Windows Forms now allows developers to build applications with the look and feel of not only Windows itself but of Microsoft Office as well. And they can deploy those applications using a much-improved deployment technology called ClickOnce that is integrated directly into the Microsoft Visual Studio 2005 design experience. Gone are the days when organizations had to default to writing Web applications. Now they can choose the technology that is appropriate for the task at hand, which means they can implement their vision without compromising the user experience. Version 1.0 of Windows Forms and the .NET Framework were a good start, but Version 2.0 takes smart client development to the next level!

Matthew MacDonald understands these changes and has created a great resource for developers who want to use the latest version of Windows Forms to create rich applications. Whether your goal is to write components for internal use or a full application, this book will help you deliver great results. Welcome back to the client.

Before Windows Forms, there were application developers, and there were control developers. Even with Visual Basic, controls were usually authored in another language like Visual C++, and authoring them required a specific set of skills. However, with an object-oriented framework like Windows Forms, control behavior can be customized with the same techniques as other application development, which gives developers a powerful new tool to really make their client applications deliver a great user experience that just can't be matched anywhere else. *Pro .NET 2.0 Windows Forms and Custom Controls in VB 2005* does an excellent job of highlighting those possibilities and equipping developers with the techniques to make them a reality. Whether you're creating an owner-drawn TreeView, using the new layout features to build dynamic interfaces, or creating skinned custom controls, this book shows you how.

The practical, task-based approach of *Pro .NET 2.0 Windows Forms and Custom Controls in VB 2005* allows the book to cover a wide range of Windows Forms topics but still provide the technical depth to help developers deliver features. While many other resources read more like technical reference documents, *Pro .NET 2.0 Windows Forms and Custom Controls in VB 2005* does an excellent job of filtering the information down to what developers really need to harness the power and innovations of Windows Forms 2.0 and deliver truly world-class client applications.

Shawn Burke
Development Manager, Windows Forms Team
Microsoft Corporation