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 heavily toward the server with the rapid growth of Web pages, HTML, and server-based applications.

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

But not everything was perfect. Large-scale Web applications were difficult to write and manage. There were differences between browsers. There weren't very good tools for debugging and developments. The applications weren't taking advantage of all the power on the client machines—hard drives, video cards, and CPUs. And most importantly, the user interfaces generally were well-suited only to the most basic data entry. If you needed real-time display or advanced visualization, things got 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 toolset 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 this, organizations were beginning to recognize the aforementioned shortcomings of Web applications in certain scenarios, and started to once again deploy client applications.

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 then 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 this and has built a great resource for developers using Windows Forms to create great, rich applications. Whether the goal is to write components for internal use or a full application, this book will help you get there and 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 required a specific set of skills. However, with an object-oriented framework like Windows Forms, customizing control behavior is done 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 C#* does an excellent job of highlighting those possibilities and equipping developers with the techniques to make them a reality. Whether it's 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 C#* allows it 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 docs, *Pro .NET 2.0 Windows Forms and Custom Controls* 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 to deliver truly world-class client applications.

Shawn Burke
Development Manager, Windows Forms Team
Microsoft Corporation