Foreword

A LONG, LONG TIME AGO when I began programming PC GUIs, there were none of these fancy framework thingies. One wrote a whole lot of C code in a case statement long enough to cut a giant's undershirt out of. I'd spent a couple weeks understanding and implementing DDE (yes, DDE) in the application we were building (and frankly, it was not the most pleasant experience) when I ran across an article in a magazine showing how this fancy thing called "Smalltalk" could do DDE in a couple of lines of code. *Wow!* I thought. *That's the way I want to program!* I've been working with and on UI frameworks pretty much ever since, which is how I ended up working on Windows Forms at Microsoft.

For V1 of Windows Forms, our goal was to produce a comprehensive UI framework that combined the ease of use of VB with the extensibility and flexibility of MFC. Along the way, we picked up additional goals, including rich design-time extensibility, GDI+ support, and support for partial trust for No Touch Deployment (NTD). I think we did a reasonable job of meeting these goals. Despite the focus on the "web stuff" when we first released, there are an enormous number of people using Windows Forms today to build all types of applications, from photo management software to applications supporting core business processes. I find seeing the interesting applications people build with Windows Forms one of the more rewarding parts of my job. However, to be honest, there are areas where we could have done better—for example, NTD had no Visual Studio support and could be complex to debug when things went wrong—so overall, I have to give V1 of Windows Forms a "shows promise" rating.

V2 of Windows Forms is about delivering on that promise. This is a major upgrade to Windows Forms. Almost every area of Windows Forms—design-time and run-time—has been improved. As Chris and Michael call out in Appendix A: What's New in Windows Forms 2.0, we have incorporated completely new features and a large number of improvements to our existing features (apparently we have 329 new types, 139 updated types, and 14,323 new members). Rather

than repeat Appendix A, I'm going to call out three new features that I think illustrate how we achieved our goals for this version of Windows Forms: solve deployment, enable great-looking apps, and enhance productivity.

Deployment

I think the single most significant feature in V2 of the .NET Framework (not just Windows Forms, but the whole .NET Framework) is ClickOnce. ClickOnce delivers on the promise of No Touch Deployment to bring easy, reliable, and manageable web-based deployment to client applications. Deploying your application via the web is now simply a matter of stepping through a wizard in Visual Studio 2005.

Great-Looking Apps

Ever since I joined Microsoft, customers have asked for the ability to build applications that look like Microsoft Office "out of the box," and you can do exactly that with V2 of Windows Forms using the new menu strip, tool strip, and status strip controls—ToolStrip, MenuStrip, and StatusStrip. Not only do the strip controls support the standard Windows and Office look and feel, but they can also be customized to look like pretty much anything you fancy.

Productivity

We've added a whole set of design-time and run-time improvements that we believe will help you to be more productive. One of my favorite new designer features is SnapLines, which allows you to quickly align controls with each other as you lay out your forms. Once you've used a designer with SnapLines, you never want to go back—it's the designer equivalent of IntelliSense.

The Future

After shipping V2, our thoughts are naturally turning to the future. Predicting the future is a dangerous business—most of the predictions from when I was a kid mean we should be supporting actors in either *The Jetsons* or *1984* by now—and so I'm a little nervous about making any long-term predictions. However, I can say a few things based on where we are and what I would like to see us do. First, the .NET Framework and managed code is here to stay: It is the programming model of the present and the future. Learning to use the .NET Framework and Windows

Forms is a solid investment for the future. Second, to paraphrase Samuel Clemens terribly, "Reports of the death of client apps are greatly exaggerated." Client applications are here to stay, can now be deployed as easily as web applications, provide significant business value, and will provide more value as time progresses. Third, as part of our continued investment in Windows Forms, we will ensure that Windows Forms works well with new technologies coming down the pipe such as those in WinFX. This allows you to build applications today with the knowledge that you will be able to enhance those applications in the future using both Windows Forms and these new technologies as they become available. Finally, from a Windows Forms perspective, I believe we need to broaden what we provide into a framework and design experience that addresses the end-to-end process of building a client application. We have a great designer to help you build your UI, but you still have to write way too much code to build your whole application. I would like to see us provide a great designer-based experience for your entire application, not just your forms.

So hopefully what I've said about Windows Forms has got you at least a little curious to find out more—which is where this book comes in. The first edition of this book was a great overview of and introduction to Windows Forms. The same is true of this second edition. Whether you are learning Windows Forms for the first time or if you just want to get a handle on the new stuff we've done in V2, this book will help you. It covers all of the significant feature areas, from the basics of creating Forms, through ToolStrips and data binding to deployment with ClickOnce.

The book is a great balancing act: It neither ignores Visual Studio 2005 nor reduces itself to a simplistic "Click here then click here" walkthrough of Visual Studio 2005 features. The book not only explains the concepts and shows you how to use those concepts in code, but it also shows you how the designer helps you to be more productive by automatically generating the code for you. This leaves you with a solid understanding of both how things work and how to use Visual Studio 2005 to get things done as productively as possible. The chapters on data binding (16 and 17) are a great example of this approach. The source code examples are another great balancing act: They are neither too short to be useful nor so long as to be overwhelming. To quote Alan Cooper, they are "Goldilocks code" examples because they are "just right."

I would like to particularly highlight the chapters on data binding (Chapters 16 and 17), not just because data binding is very close to my heart, but because the book does an excellent job of explaining how data binding works and how to use it effectively. I would also like to highlight the chapters on writing design-time behavior for your controls and components (Chapters 11 and 12) because this is a subject that is often neglected. These chapters alone make this a "must read" book.

So, in summary, this book will leave you not only in a position to effectively use what we provide as part of Windows Forms but also with the understanding you need to write your own run-time and design-time extensions to what we provide.

I'd like to close with some acknowledgments and thanks. First, thanks to the entire Windows Forms team, who have worked tirelessly to build and ship what I believe is a great product. I'm very proud of what we have achieved. Second, my thanks to Michael and Chris not only for producing a book that does a great job of explaining our product and will make it easier for our customers to use, but also for their contributions to the wider Windows Forms community. Thanks to Michael for his great articles on MSDN and feedback on Windows Forms V2—particularly his feedback on the ToolStrip controls. Thanks to Chris for his seemingly boundless enthusiasm for Windows Forms, his excellent writing on Windows Forms, his deep understanding of Windows Forms, MFC, and Windows, and his polite and measured but enthusiastic and copious feedback on every aspect of the product. Both Mike and Chris have helped enormously both in promoting understanding of the product and in helping make Windows Forms as good as it is today. And finally, my thanks to our customers: Every single feature in Windows Forms V2 is based on your feedback. So to all of you who took the time to give us feedback and suggestions: Thanks! Please keep it coming!

I hope you have fun using Windows Forms.

Mark Boulter
PM Technical Lead,
Client Development Tools, Microsoft