# Foreword

WOULD JAMES BOND be just as effective an agent without all the tools and gadgets that Q provides? Would he be able to get out of those tight spots without them or would it simply just take him longer? Would he still have his license to kill? Would anyone else be a superagent just by having access to the tools? What makes *Bond, James Bond* effective—is it the tools, or is it his training, knowledge, and skills that transcend tools and technologies?

I think that the same set of questions is applicable to building modern applications using Visual Studio: What makes a developer productive and effective—is it the tools, the wizards, and the .NET Framework classes, or is it the knowledge of how to best design and build applications? Can you really build maintainable, robust, reusable, extensible, secure, and consistent applications simply by doing drag-and-drop in Visual Studio? What does it take to train a developer to be productive and have that developer figure out the correct ways of using new tools and the associated techniques as time goes by? Will that developer be able to extend and improve on the basic offering of the tools and the Framework classes when needed, customize and specialize the generated code, and when appropriate, use those tools in a different scenario than what Microsoft had in mind?

Nowhere are these questions more apt than when it comes to data binding and data access. The Data Sources window in Visual Studio 2005 generates tons of specialized code, and the designers hook up that code to the visual controls, all in a seamless, automated manner. But is that all there is

to it? I am convinced that the key for achieving goals such as maintainability, quality, and extensibility is the understanding of what exactly the tools generate and why and understanding and appreciating the overall design approach and the implicit best practices involved. Only once you have that do you stand a fighting chance. The reason is simple—the machine-generated code as well as the Framework classes (such as the DataGridView) are designed for the broadest possible set of applications and use cases. The moment you deviate from the garden path (as you inevitably will), you are on your own, and only your skills and knowledge can carry you forward at that point. I believe that the purpose of wizards is not to allow anyone to develop applications. Rather, the aim is to off-shoulder from the skilled developers the time-consuming, detailed, mundane, and repetitive tasks, allowing them to be more productive and to focus on the application's logic and the required use cases. I think that to use these tools, you need a "license to wizard"—and only after understanding what and why the designers generate are you allowed to take advantage of it.

This book is all about the *what* and the *why* of binding to data sources in a Windows Forms application built using Visual Studio 2005. The book goes into great detail in explaining the rationale behind the designer-generated code, recommends best practices, gives tips and tricks, demystifies the machine-generated code and how to extend it, and shows how to compensate for its limitations. Not only that, but this book also prepares you to unleash the power and user-friendliness of the smart client today and tomorrow. This book is your license to wizard, designed to make you the James Bond of data binding, and I think that says it all.

*—Juval Löwy*
*August 2005*