Extending MPC Applications with the NET Framework
To anyone looking to augment their existing MFC code base and knowledge with the powerful .NET classes which provide such capabilities as disconnected data, in-memory database (IMDB), regular expressions, and data encryption, Tom Archer's book has it all.  
Erik Westermann, Lead Architect, Eidenai Innovations

Extending MFC Applications with the .NET Framework is the first book to show MFC developers how to boost productivity by incorporating .NET functionality into existing MFC applications. Tom Archer clearly illustrates how using the .NET Base Class Library (BCL) to complement or replace MFC classes when there is a clear advantage enables MFC developers to create elegant and robust Windows applications in the most efficient way possible.

Each chapter begins with an introduction that explains the technology, outlines its benefits, lists its pragmatic business uses, and summarizes the required syntax. As is an Archer trademark, this information is solidified with hands-on, practical demo applications.

This book answers the questions MFC developers have about .NET, including:

* What are the key issues when combining MFC and Managed Extensions? Page 15
* How can Regular Expressions be used to parse a document for multiple complex pattern types, such as email addresses? Page 84
* How can hash codes be used to validate users without storing passwords? Page 2
161

- How can MFC applications maintain data stored in XML format? Page 274
- Once an ADO.NET DataSet is filled, how can the data be searched, sorted, and filtered? Page 336
- What technique is used for reading image data from a database using ADO.NET classes? Page 363
- What are the options for handling disconnected data concurrency issues? Page 396
- How can Remoting be used to pass MFC objects between applications? Page 460
Extending MFC Applications with the .NET Framework

By Tom Archer, Nishant Sivakumar

Publisher: Addison Wesley
Pub Date: December 26, 2003
Pages: 656
Slots: 1.0

Copyright
Praise for Extending MFC Applications with the .NET Framework
Foreword
Preface
Assumptions and Goals
Technical Matters
Conventions Used in This Book
Demo Applications
Author Feedback
Acknowledgments
Technical Editors
About the Authors

Chapter 1. Combining MFC and Managed Extensions
Introduction
Writing a Managed Extensions Application
Writing an MFC Application That Accesses .NET
Mixed-Mode Programming Issues
Working with .NET Types
Summary

Chapter 2. Regular Expressions
Introduction
Using the Regex Class to Split Strings
Searching Strings with the Match and MatchCollection Classes
Processing Groups and Captures
Parsing and Replacing Strings
Replacing Matches Using Groups and Substitution Patterns
Putting It All Together: Writing an E-mail Parser
Summary

4
Chapter 3. File I/O and Registry
  Introduction
  Stream, Readers, and Writers in .NET
  File System Classes
  Accessing the Registry
  Summary

Chapter 4. Cryptography, Hash Codes, and Data Encryption
  Introduction
  Cryptography Basics
  Using Hash Codes
  Using Symmetric (Private) Key Encryption
  Using Asymmetric (Public) Key Encryption
  Combining Symmetric Encryption, Asymmetric Encryption, and Hash Codes
  Using Digital Signatures
  Summary

Chapter 5. XML and the DOM
  Introduction
  Writing and Reading XML Documents
  The DOM and the XmlDocument Class
  Summary

Chapter 6. Disconnected Data via ADO.NET and DataSets
  Introduction
  ADO.NET Basics
  Basic Database Operations with ADO.NET
  Working in Batch
  Working with Row State Information
  Searching, Sorting, and Filtering Data
  Working with Typed Datasets
  Summary

Chapter 7. Advanced ADO.NET
  Introduction
  Working with Binary ( BLOB ) Data
  Associating Related Tables with the DataRelation Class
  Understanding Concurrency
  Summary

Chapter 8. Combining ADO.NET and XML
  Introduction
  Serializing XML from a Single DataTable or Multiple Unrelated DataTables
  Serializing Datasets Containing Related Data Tables
  Mapping DataTable Columns to XML Node Types
  Saving Formatted XML
  Writing Schema Information
  Generating XML from a DataTable
  Filling DataSet Objects from XML
  Working with DiffGrams
  Summary

Chapter 9. Remoting
Copyright

About the Cover:

**Bridging Two Worlds**
By Tom Archer and Justin Dunlap

*Two bold entities*
*One litknown and understood*
*One darkmystery*

*Two worlds set apart*
*Both with so much to offer*
*How best to combine?*

*This booklike a bridge*
*Bringing both worlds together*
*Joining light and dark*

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Addison-Wesley was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers discounts on this book when ordered in quantity for bulk purchases and special sales. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
[corpsales@pearsontechgroup.com](mailto:corpsales@pearsontechgroup.com)

For sales outside of the U.S., please contact:

International Sales
(317) 581-3793
[international@perasontechgroup.com](mailto:international@perasontechgroup.com)

Visit Addison-Wesley on the Web: [www.awprofessional.com](http://www.awprofessional.com)
Praise for *Extending MFC Applications with the .NET Framework*

"Using Microsoft .NET in a new application is easy. Taking advantage of .NET in existing MFC applications is the real challenge! Tom Archer's book provides the information developers need to begin profiting from .NET today without a costly up-front rewrite of existing software."

Mark Burhop  
Systems Architect  
Integraph Corporation

"To anyone using MFC who hasn't decided whether or not to jump onto the .NET bandwagon do it! This book clearly illustrates the major productivity advantages and cost benefits when combining the .NET classes with existing MFC applications."

Brian Delahunty  
CodeProject.com

"To anyone looking to augment their existing MFC code base and knowledge with the powerful .NET classes providing such capabilities as disconnected data, in-memory database (IMDB), regular expressions, and data encryption Tom Archer's book has it all."

Erik Westermann  
Lead Architect  
Eidenai Innovations
"Yes!" That was how I felt when I started reading the proofs of this book. This is the first book I have seen that explains very clearly how Visual C++ developers can use the .NET Framework right now to increase their productivity in writing new MFC applications and enhancing existing ones. It addresses very successfully the mechanics of how you start using .NET APIs from your MFC applications. It explains where the feature our team called IJW (It Just Works), really just works and where you need to coax it across some pitfalls. But this book dispenses with that in short order and then goes straight to the real meat of selecting specific API sets from the vast breadth of the .NET Framework and explaining how to use them from your MFC code.

Just reading through the table of contents and the first few chapters it becomes clear immediately that Tom Archer is writing from the ideal vantage point of both having a deep understanding of the technology and the direct hands-on experience of using it in the real world, on a real project, with real deadlines, and on a team with real people facing all the pressures and deadlines of the typical modern development project.

What you won't find in here is the umpteenth overview of the .NET Framework, or even a book just focused on explaining the .NET Framework from the viewpoint of a C++ or MFC developer. What you will find is exactly which parts of the .NET Framework you can use now to increase your productivity which parts are clearly superior in the .NET Framework compared to equivalent (or in some case non-existent) Win32, COM, or MFC functionality. This book simply couldn't have been written a year ago because it is so deeply steeped in real-world usage. The author's experience with all the relevant parts of the technology C++, MFC, Win32, the CLR, and the .NET Framework clearly shows through in every single page. And so does his talent for clear, unambiguous explanations.

Because WinFX, the new API for the Longhorn OS, is a managed API set, using the techniques in this book ideally positions you to make your MFC applications great Longhorn applications. Starting now to use the .NET Framework for specific technologies like remoting, file access, cryptography, and XML will not only jump-start your experience, but also will later make it easier to integrate the new Longhorn features into your application.

Designing the new releases of Visual C++ to help make you, the C++ developer on the Windows platform, ideally positioned to take advantage of the new platform functionality, even with a major shift like .NET and Longhorn, is what our team is all about and what I personally spend almost all my waking moments on. This book is all about how that works in the real world, with the current product. It both validates where we got it right and shows you how to make it work in the places where we haven't yet reached the ideal solution. It makes the hard value judgments in the context of your MFC applications as to which parts of the .NET API are beneficial to use now...
and which parts don't offer compelling advantages over the traditional ways of implementing the functionality.

I have already been using the material in this book in guiding the future evolution of Visual C++. This is a book that is going on my shelf of books that I use daily. I think it will go on yours as well.

Ronald Laeremans
Group Program Manager, Visual C++
Microsoft Corporation
Preface

Assumptions and Goals

Technical Matters

Conventions Used in This Book

Demo Applications

Author Feedback
Assumptions and Goals

This book assumes that you are an experienced MFC developer who, while not ready to completely migrate your code to .NET, is interested in seeing how .NET can make you more productive. As an MFC developer, you're infinitely more comfortable with the document/view and dialog model of creating user interfaces and are simply much more productive with this development environment. In addition, you might have a large amount of source code that you're not willing or able to migrate to .NET right now if ever. However, you're still intrigued by some of the .NET classes and how they might complement or even replace various MFC counterpart classes. That is specifically who this book targets. As opposed to the many .NET books that seem to take the attitude that it will be all or nothing for developers, this book is more realistic in realizing that many developers (especially those using C++ and MFC) will want to use .NET only where there is a clear and obvious advantage.

In addition, while I do go into some detail on the various .NET classes that I illustrate throughout this book, the book is not a " .NET class library book." There are hundreds of .NET types and classes, and that simply isn't the focus here. My objective is to show you a few select classes that I feel would benefit most MFC developers; see, for example, the XML chapter.

XML has become the standard for exchanging data in many organizations. However, in order to use XML from MFC, you're probably going to be using the Microsoft IE (Internet Explorer) XMLDOM via COM (Component Object Model). While this is workable solution I've written several articles and book chapters on how this is done using the XMLDOM just doesn't compare to the very elegant and powerful set of .NET XML classes. Therefore, a complete chapter is devoted to the .NET XML namespace, covering everything from reading/writing XML files and traversing XML documents to querying XML data using XPath.

As a result of this approach, when finished with this book, you will know how to integrate the power of the .NET Base Class Library (BCL) into your existing and future MFC applications, which will ultimately make you a more productive and more marketable programmer.
Technical Matters

While it's sometimes helpful to think of Visual C++ .NET as containing two separate C++ compilers—one for managed (.NET) code and one for unmanaged code—technically, the Visual C++ .NET product contains only one compiler that has two distinct parts:

- **ISO Standard C++**, an ISO-compliant C++ compiler that can be used (with the inclusion of the appropriate libraries) to generate native (x86) executables and libraries for projects such as console applications, Windows applications, Windows services, and so on. When combined with the MFC class library, this is what pays the bills for most of the intended readers of this book (and its authors).

- **Managed Extensions**, which are the language constructs available for mixing managed C++ syntax with your standard C++ code. The Managed Extensions allow you to define and use managed types (garbage-collected reference type, value types, scoped enums, delegates, etc.) and use managed runtime facilities such as reflection and security.

Therefore, there's only one compiler, and you can think of Managed Extensions as somewhat like other Microsoft C/C++ extensions, such as __declspec, except obviously more extensive, as it has the responsibility of exposing all of the CLI features: garbage collection, reflection, and so on. You might even think of Managed Extensions as a superset language containing Standard C++ as a subset. So now we know that a single compiler can generate both native and managed code for Standard C++. What about mixed code applications? Let's look at two techniques for mixing native and managed C++ code: COM Interop and IJW (It Just Works).

There are actually two types of COM Interop. There is the COM Callable Wrapper (CCW), which allows you to create a "pure" native application (with no .NET build-time dependencies) that accesses .NET code via COM, and the Runtime Callable Wrapper (RCW), which allows you to access native code from managed applications. Therefore, one direction I could have taken in this book would have been to write MFC applications and then access the various selected .NET classes via CCW. However, CCW has many drawbacks that inevitably make it unappealing. First, CCW is not very efficient in terms of execution speed. Second, CCW cannot use the entire CLR type system—method parameters and properties are restricted to automation types. Finally, many .NET types are not exposed to COM, so accessing them would be very difficult or impossible.

IJW refers to the collection of mechanisms that allow managed code to call native functions, compile unmanaged types in MSIL, use unmanaged types in managed type method signatures, provide native entry points to MSIL methods, and so on. However, IJW is not only about managed code calling native functions. It's also about exposing...
native entry points to managed functions and basically doing all the plumbing necessary to get C++ code to work under .NET.

If you're a bit confused by the last paragraph, that's perfectly understandable. After all, it now sounds like I'm talking about writing .NET applications that use native code when the book's topic promises to be about the opposite using .NET to augment MFC applications! As it turns out, although at first blush this does seem to be backwards, and the designers of IJW probably intended it for .NET applications that need to occasionally call a native function, it turns out that it works just fine for a .NET application that is 90% native. As a result, since IJW has full access to the .NET class hierarchy and type system and is much more efficient than CCW, this is the technique used throughout the book.
Conventions Used in This Book

All programmers have their own pet terms and coding styles. Therefore, let me clarify a few terms you'll see throughout the book:

- **"Problem domain"** I first picked up this term many years ago while using the Coad/Yourdon Object-Oriented Analysis and Design methodology. "Problem domain" is a generic term that refers to the set of problems to be solved.

- **"Consumers" and "clients"** These terms are used interchangeably to represent any code that uses a class or type. You'll also see this term used from the perspective of the MFC application being a consumer, or client, of the .NET class library or a specific .NET class.

- **"Server"** This term is used to refer to a piece of code typically a class that is used by a client or consumer.

- **"Arguments" and "parameters"** Like most programmers, I use these terms interchangeably when referring to the values passed to a method.

- **"Function" vs. "method"** The object-oriented programming purists will tell you that you're supposed to use the term *method* instead of *member function*. However, having been an MFC developer for more than 10 years, I still find it difficult to refer to a C++ function as a method. Therefore, you'll see both terms used, depending on the context. If I'm referring to native C++ code, I use the term *function* and if I'm referring to .NET code, I use the term *method*.

- **Naming conventions** This particular book presented a unique challenge when it came to naming conventions. I've been using Visual C++ and MFC since version 1 and have always used Hungarian notation for the main reason that the MFC source code did, and I wanted to maintain consistency in my own code bases. However, the generally accepted naming convention with .NET programming is a mixture of camel and pascal casing. Mix the two development environments of MFC and .NET, and you can see the problem. In the end I tried to localize my .NET code to standalone functions, but this wasn't always possible. As a result, you will often see a combination of the two distinct naming conventions.

- **Breaking lines of code** Unfortunately, there are some times when the format of a book requires some odd line breaking of source code. This is regrettable and visually unappealing but unavoidable.
Demo Applications

System Requirements

I'm a big believer in "hands-on" training, which is why there are several step-by-step, tutorial-like demo applications in each chapter. In order for you to get the most out of this book, I highly recommend that you work your way through these demos. Most are intentionally very simple in scope in order to focus on the chapter's topic, but coding the demo will often help you to solidify what you've learned in the chapter's text.

Here's what you'll need to work through this book's demos:

- **Coding the demo applications** You'll need Visual Studio .NET (this book was written using Visual Studio 2003 and the .NET Framework 1.1).

- **Running the demo applications** You'll need to execute the applications on a machine equipped with the .NET runtime.

About the Demos

Here a few notes specifically related to this book's demos and chapter code snippets.

- I've included a Visual Studio .NET Custom AppWizard with this book that automates the creation of MFC/.NET projects. The wizard is located in the Tools folder of the CD-ROM and includes a `readme.txt` file that explains installation and usage. Note that the chapter's step-by-step instructions for creating the various demos do not refer to this wizard because some people will want to manually create and configure their projects. However, once you are comfortable with creating MFC/.NET mixed mode projects, I would suggest using the wizard, as it does save you a few steps and prevents needless recompiles when you manually create the project and forget to include a needed namespace or assembly reference in your code.

- When you run the demos provided on the book's CD-ROM, you will see that the dialogs and views will occasionally contain additional controls not seen in the chapter's figures or mentioned in the step-by-step instructions. The reason is that the chapter focuses on what you need to code in order to illustrate a given technical point or technique, while the additional controls on the finished demo are simply there to either make the final UI more appealing or to provide
additional instructions for running the demo.

- For the same reason noted above, some demos will have code that is not mentioned in chapter, so the chapter's code and the CD-ROM code won't always match up 100%. As an example, several demos use a list view control, and therefore each of these demos includes functions to allow for full-row selection and the auto-sizing of columns to match the control's text. However, to explain those functions or to even include them in the step-by-step instructions for something like a demo whose focus is regular expressions would be to take the focus away from the main point of the demo.

- Most of the demos have a button labeled Contact Me that allows you to easily access my Web site in order to view the most current version of the book's demos and a current errata list for the book, as well as to contact me.
Author Feedback

I'm always eager to hear from readers with constructive criticisms (and, I hope, ways of making the next edition even better) or questions about using what you've learned in these pages. To that extent, I can be reached through my company's Web site: www.ArcherConsultingGroup.com. Along with this book's updated demo applications, I will also maintain current errata for this book on my site.

If you do submit a question, critique, or anything that requires feedback from me, I would only ask for a bit of patience. Due to the success of my books and training seminars, my schedule gets very hectic and I travel quite a bit. Having said that, I'll respond as quickly as time permits, which typically translates to within a couple of business days.
Acknowledgments

Many people help take a book from the germ of an idea to completion, and it's important to recognize their individual contributions to the team. The book is much better as a result of the following contributors.

- **Sondra Scott** Without Sondra's truly caring and compassionate support, this book would never have been completed. Thanks so much for all your help, Sondra!

- **Nishant Sivakumar** Known affectionately on the CodeProject Web site simply as Nish, he contributed several chapters and demos to this book, and his contributions are greatly appreciated.

- **Erik Westermann** I'd like to give many heartfelt thanks to my dear friend and fellow author, Erik, not just for helping on the book, but also for being someone I can depend on in a crunch. At the very last minute, I made some rather significant changes to the text, and Erik jumped in literally on a second's notice and provided some incredible insight and technical help. The end result is what you hold in your hands now.

- **Mark Burhop** and **Dmitri Riz** were the technical editors assigned to this book from Addison-Wesley. It's widely accepted in the technical publishing industry that Addison-Wesley only hires the best, and you guys are shining examples of that. You were both incredibly constructive and helpful with your critiques and advice and helped to shape much of what became the final result.

- **Brian Delahunty** wrote and donated the "MFC.NET" Visual Studio Custom AppWizard. This wizard enables you to automate the process of making the required project settings and including the desired .NET namespaces and assembly references for a combined MFC and .NET project with a few clicks of the mouse. The wizard is located in the Tools folder of the CD-ROM and includes a readme.txt file that explains installation and usage. Great work, Brian!

- **Don J. Plaistow** is a Perl and Regular Expressions guru who was kind enough to help me with the Regular Expressions chapters specifically the expression used in parsing text files for e-mail addresses.

- **Mauricio Ritter** kindly donated his time and database expertise and helped solve one especially tricky problem—inserting relational data using the **DataSet**