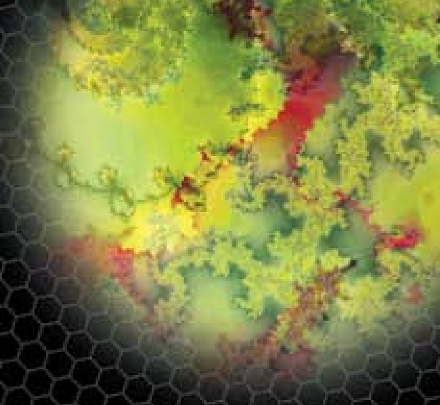
THE EXPERT'S VOICE® IN .NET



Expert C++/CLI

.NET for Visual C++ Programmers

Marcus Heege

Apress®

Expert C++/CLI: .NET for Visual C++ Programmers

■ ■ ■

Marcus Heege

Apress®

Expert C++/CLI: .NET for Visual C++ Programmers

Copyright © 2007 by Marcus Heege

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13: 978-1-59059-756-9

ISBN-10: 1-59059-756-7

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: James Huddleston
Technical Reviewer: Stanley Lippman
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Jeff Pepper, Paul Sarknas, Dominic Shakeshaft, Jim Sumser, Matt Wade
Project Manager: Elizabeth Seymour
Copy Edit Manager: Nicole Flores
Copy Editor: Damon Larson
Assistant Production Director: Kari Brooks-Copony
Production Editor: Lori Bring
Compositor: Gina Rexrode
Proofreader: Patrick Vincent
Indexer: Brenda Miller
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at `http://www.apress.com` in the Source Code/ Download section.

Contents at a Glance

About the Author xiii
About the Technical Reviewer. xv
Acknowledgments xvii

CHAPTER 1 Why C++/CLI? 1
CHAPTER 2 Managed Types, Instances, and Memory 11
CHAPTER 3 Writing Simple .NET Applications 31
CHAPTER 4 Assemblies, Metadata, and Runtime Services 49
CHAPTER 5 Defining Managed Types 73
CHAPTER 6 Special Member Functions and Resource Management 117
CHAPTER 7 Using C++/CLI to Extend Visual C++ Projects with Managed Code 143
CHAPTER 8 Mixing the Managed and the Native Type System 173
CHAPTER 9 Managed-Unmanaged Transitions 203
CHAPTER 10 Wrapping Native Libraries 233
CHAPTER 11 Reliable Resource Management 253
CHAPTER 12 Assembly Startup and Runtime Initialization 279
APPENDIX A Programmatically Updating the .NET Security Policy 303
APPENDIX B Measuring the Performance of Thunks 307

INDEX 319

Contents

About the Author xiii
About the Technical Reviewer. xv
Acknowledgments xvii

■CHAPTER 1 **Why C++/CLI?** 1

Extending C++ with .NET Features 1
What Is .NET?. 2
What Is C++/CLI? 3
Building C++/CLI Applications 3
Object File Compatibility. 4
Interaction Between Managed and Unmanaged Code 6
DLLs with Managed Entry Points 7
Compilation Models. 8
Wrapping Native Libraries 9
Summary 10

■CHAPTER 2 **Managed Types, Instances, and Memory** 11

System::Object. 12
Primitive Types. 13
Custom CTS Type Definitions 14
Managed Memory 15
    Managed Heap 15
    Tracking Handles 16
Values and Objects 18
Value Types and Reference Types. 20
    Boxing 20
    Unboxing 22
System::String 23
Managed Arrays. 24
    Managed Array Initialization. 25
    Iterating Through an Array 26
    Managed Arrays of Tracking Handles. 28
Summary 29

CHAPTER 3 **Writing Simple .NET Applications** 31
Referencing Assemblies 31
    Assembly References in Visual Studio 32
Assemblies and Type Identity 34
Avoiding Naming Conflicts 35
Command-Line Arguments 36
Stream-Based IO 37
    Text IO 38
    Reading and Writing Binary Data 39
Managed Exception Handling 40
    try...finally 42
Web Requests 42
Casting Managed Types 43
Managed Debug Helpers 45
Configuration Files 46
Summary 47

CHAPTER 4 **Assemblies, Metadata, and Runtime Services** 49
Assemblies and Metadata 49
    Assembly Manifests 51
Metadata APIs 52
Assembly Identity 53
Assembly Loading and Deployment 55
    The Global Assembly Cache (GAC) 56
    Version Redirections 57
    Manual Assembly Loading 59
Consuming Metadata for Types and Type Members at Runtime 59
    Dynamic Type Instantiation 61
    Runtime Information About Type Members 63
    Dynamic Member Access 64
    Access to Private Members 66
    Attributes 67
System::Runtime::Serialization 69
Summary 71

CHAPTER 5 **Defining Managed Types** 73

Type Visibility 74
    Friend Assemblies 74
Value Type Definitions. 76
Managed Enums 77
Type Member Visibility 79
    Visibility and Naming Conventions 82
Methods 82
    Default Arguments Are Not Supported 84
    const Methods Are Not Supported 85
Fields 85
    Bye-Bye const 86
Type Initialization. 86
Inheritance 88
    Inheritance and Versioning 90
    Virtual Functions 91
Overriding Interface Members. 94
Interfaces Are Immutable 96
Has-A Relationships 98
Components 102
    Handling Events of a Component 105
Defining Custom Component Classes 106
Defining Custom Events 108
    Event Handler Delegates 111
    Nontrivial Events 114
Summary 116

CHAPTER 6 **Special Member Functions and Resource Management** 117

Object Construction. 117
    Virtual Functions Called on an Object During Construction Time. 118
    Order of Calls to Dependent Constructors. 121
Object Destruction. 123
    Disposing Objects 126
Cleanup for Automatic Variables. 128
    Obtaining a Tracking Handle from an Implicitly Dereferenced Variable 129

Automatic Disposal of Fields 130
Access to Disposed Objects. 132
Requirements for Destructors of Managed Types 134
auto_handle 135
    auto_handle and cleanup. 137
Copy Constructors and Assignment Operators. 140
Summary 141

CHAPTER 7 Using C++/CLI to Extend Visual C++ Projects with Managed Code 143

Up-Front Considerations. 143
Which Compilation Model Should You Choose? 145
    Load-Time Dependencies to Other DLLs. 146
    Why Should You Use /clr:pure at All? 148
    Compilation Models and .NET Security 151
    Adapting the Security Policy for Assemblies
        Using C++/CLI Interoperability. 154
Compatibility with Other Compiler Switches 154
    Managed Compilation and the C/C++ Runtime Library 155
    Managed Compilation and Exception Handling (/EHa) 155
    Features Incompatible with C++/CLI 155
Reviewing Compilation Models. 156
Step by Step 157
    Step 1: Modifying Settings at the Project Level 158
    Step 2: Creating a Second Precompiled Header. 159
    Step 3: Building and Testing. 160
    Step 4: Adding Additional Source Files Compiled with /clr 161
    Step 5: Compiling Existing Files with /clr Only If Necessary. 162
Handling Exceptions Across Managed-Unmanaged Boundaries. 162
    Mapping SEH Exceptions to .NET Exceptions 164
    Catching C++ Exceptions. 166
    Catching Managed Exceptions in Native Code 167
General Hints for Mixed Compilation 168
    Avoid #pragma (un)managed 168
    Automatic Choice of Compilation Model: Avoid Warning 4793! . . . 169
    Predefined Macros for Compilation Models 169
    Compilation Models and Templates 170
Summary 171

CHAPTER 8 **Mixing the Managed and the Native Type System** 173

Using Native Types in Managed Code 174
Using C Structures in Managed Code 177
Using C++ Classes in Managed Code 180
String Literals 182
Passing Managed Memory to a Native Function 183
Converting Between Managed and Native Strings 186
Mixing the Type Systems When Defining Data Members 188
Referring to Managed Objects in C++ Classes 190
    Other Uses of gcroot and auto_gcroot 192
    General Hints Regarding gcroot and auto_gcroot 193
    Reducing the Overhead of gcroot and auto_gcroot 194
Handling Events in Native Classes 197
    Internals of the Delegate Map 199
Summary 201

CHAPTER 9 **Managed-Unmanaged Transitions** 203

Interoperability, Metadata, and Thunks 204
Calling Managed Functions from Unmanaged Code 205
    Interoperability Metadata for Unmanaged-to-Managed Transitions 205
    Default Calling Conventions 207
Implicit Optimizations of Native-to-Managed Transitions 208
    Native and Managed Callers 208
    Managed Callers Only 209
Calling Native Functions from Managed Code 210
    Calling Local Native Functions from Managed Code 210
    Calling Native Functions Imported from DLLs. 211
Calling C++ Classes Across Managed-Unmanaged Boundaries 214
Passing Native-Managed Boundaries with Function Pointers 217
Passing Native-Managed Boundaries with Virtual Function Calls 220
    Virtual Functions and Double Thunking 222
Performance of Thunks 223
Optimizing Thunks. 225
    GetLastError-Caching 226
    Be Aware of Implicit GetLastError-Caching Optimizations 229
Generic Thunks and P/Invoke Type Marshaling 231
Summary 232

CHAPTER 10 Wrapping Native Libraries 233

Up-Front Considerations. 233
    Should You Implement Wrapper Types in a Separate DLL or Integrate Them into the Native Library Project? 233
    Which Features of the Native Library Should Be Exposed? 234
Language Interoperability 235
Wrapping C++ Classes 237
    Mapping Native Types to CLS-Compliant Types 238
    Mapping C++ Exceptions to Managed Exceptions 242
    Mapping Arguments of Managed Array Types to Native Types . . . 243
    Mapping Other Non-Primitive Arguments 244
    Supporting Inheritance and Virtual Functions 248
General Recommendations 250
    Simplify Wrappers Right from the Start 250
    Be Aware of the .NET Mentality. 250
Summary 251

CHAPTER 11 Reliable Resource Management 253

Wrapping Native Resources. 255
Limits of IDisposable::Dispose 257
Garbage Collection and Last-Chance Cleanup 257
What Should a Finalizer Clean Up?. 259
Finalization Issue 1: Timing 260
    When Is a Reference on the Stack a Root Reference? 261
    Reproducing the Finalization Timing Problem. 262
    Preventing Objects from Being Finalized During P/Invoke Calls . . 265
Finalization Issue 2: Graph Promotion 266
Prioritizing Finalization 268
Finalization Issue 3: Asynchronous Exceptions 269
    ThreadAbortException. 270
    StackOverflowException. 271
    OutOfMemoryException 273
    ExecutionEngineException 274
SafeHandle 274
Summary 277

CHAPTER 12 Assembly Startup and Runtime Initialization 279

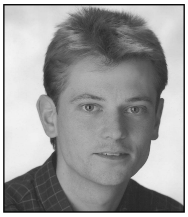
Application Startup 279
    CLR Startup. 280
Loading the Application Assembly 281
CRT Initialization in /clr[:pure] Assemblies 281
Linking the CRT in Mixed-Code Assemblies 283
The Module Constructor. 284
The Managed Entry Point. 285
DLL Startup. 288
    CRT Initialization in /clr DLLs. 291
    Custom Startup Logic and Load-Time Deadlocks 292
    Initialization of Global and Static Variables 296
    DLLs and Module Constructors. 298
    Initialization Timing Problems 298
    CRT Initialization in /clr:pure DLLs 302

APPENDIX A Programmatically Updating the .NET Security Policy. . . . 303

APPENDIX B Measuring the Performance of Thunks. 307

INDEX 319

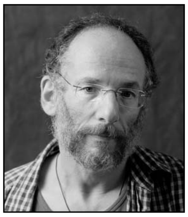
About the Author



MARCUS HEEGE has over 20 years of experience developing software for Microsoft languages and platforms. He is a course author and instructor for DevelopMentor, where he developed the Essential C++/CLI: Building and Migrating Applications and Components with C++/CLI seminar. He also serves as a troubleshooter and mentor for many professional software developers.

Marcus blogs about C++ and .NET topics at `www.heege.net/blog`, and he has written dozens of articles for a wide variety of magazines. Marcus is an MVP for Visual C++ and has been a Microsoft Certified Solution Developer and Microsoft Certified Trainer since 1997.

About the Technical Reviewer



STANLEY LIPPMAN served as architect with the Visual C++ team during the four-year development of C++/CLI. He is currently a senior software engineer with Emergent Game Technologies, a provider of middleware software for massive multiplayer online games. Stan also writes "Netting C++," a bimonthly column for *MSDN* magazine.

Acknowledgments

Writing this book was only possible because I got a lot of help from people that deserve to be named here. From the team at Apress I would like to thank Elizabeth Seymour, Lori Bring, Jim Huddleston, and Damon Larson.

I would also like to thank Stan Lippman, the technical reviewer of my book. His feedback constantly pushed me to make the book better. Without his feedback, the book would have been less correct and much less readable. I also got valuable reviews from Richard Dutton.

Several members of the Visual C++ team have been great sources for in-depth information about implementation details of C++/CLI and the CLR. These guys are Martyn Lovell, Brandon Bray, Arjun Bijanki, Jeff Peil, Herb Sutter, Ayman Shoukry, and Bill Dunlap. Many topics that I cover in this book are insights that I got from them.

Over the last few years, I have learned a lot about various areas of software development in discussions with my DevelopMentor colleagues, including Dominick Baier, Richard Blewett, Mark Vince Smit, and Mark Smith, as well as from other smart software developers, including Mirko Matytschak, Klaus Jaroslawsky, and Ian Griffiths.

Furthermore, I would like to thank the students of my C++/CLI classes. By asking me many interesting questions, they have allowed me to review my knowledge in many different practical contexts.

The biggest thank you goes to my wife, Marion, and my two kids, Lisa Maria and Jule. Since the time they started being the center of my world, they have continuously supported me with all the love and power they have. For the many months that I was writing this book, they had to accept that I had to spend most of my time researching and writing instead of being a good husband and a good daddy.

Marcus Heege
Kaisersesch, Germany
February 2007