

# Visual Basic® 6

## ERROR CODING & LAYERING

- Code smarter!
- Concrete strategies for excellence in Visual Basic development
- Architecting for easier development and maintenance
- More than 20 superior techniques for error coding
- Breakthrough solutions for managing Visual Basic 6 enterprise projects

Suranaree University of Technology



31051000622353

Tyson Gill



MICROSOFT® TECHNOLOGIES SERIES

# CONTENTS

.....

**ABOUT THE AUTHOR**      **XV**

**LIST OF FIGURES**      **XVII**

**PREFACE**      **XIX**

    Coding Smarter    xix

    This Book Is for You    xix

    Getting the Most from This Book    xx

    Acknowledgments    xx

## **ONE**    Your Software Development Mission    1

1.1    Drafting Your Mission    2

1.2    Retaining Corporate Knowledge    4

1.3    Standardizing the Creative Process    5

1.4    Error Coding    7

1.5    Coding Smarter    7

1.6    Identifying the Possible    9

1.7    Achieving the Possible    10

1.8    The Smart Coding Triangle    11

1.9    Barriers to Your Mission    12

## **TWO**    Understanding the Barriers to Your Mission    13

2.1    Visual Basic Error Coding    13

2.2    Why Good Error Coding Is Seldom Achieved    15

    2.2.1    *Sample Code Focuses on Functionality*    15

    2.2.2    *Error Coding Is Not Glamorous*    15

|        |                                                                             |    |
|--------|-----------------------------------------------------------------------------|----|
| 2.2.3  | <i>Error Coding Is Difficult to Learn</i>                                   | 15 |
| 2.2.4  | <i>Error Coding Is Difficult to Do</i>                                      | 15 |
| 2.2.5  | <i>Error Coding Is Seen as Ancillary</i>                                    | 16 |
| 2.2.6  | <i>Error Coding Is Taken for Granted</i>                                    | 16 |
| 2.2.7  | <i>Error Coding Takes a Large Amount of Code</i>                            | 16 |
| 2.2.8  | <i>Error Coding Cannot Be Properly Implemented Separately</i>               | 16 |
| 2.2.9  | <i>Error Coding Is Not Visible in Final Product</i>                         | 16 |
| 2.2.10 | <i>Error Coding Is the First Place to Skimp</i>                             | 16 |
| 2.2.11 | <i>If We Do Skimp on Error Coding, So What?</i>                             | 16 |
| 2.2.12 | <i>Even If Errors Appear Later, They Can Always Be Fixed as They Appear</i> | 17 |
| 2.2.13 | <i>Management Creates Disincentives</i>                                     | 17 |
| 2.2.14 | <i>Code Is Viewed as Disposable</i>                                         | 17 |
| 2.3    | <i>"We Will Adapt"</i>                                                      | 17 |
| 2.4    | <i>Achieving Good Error Coding</i>                                          | 19 |
| 2.5    | <i>Barriers to Error Coding</i>                                             | 19 |
| 2.6    | <i>Evaluating Error Coding</i>                                              | 20 |
| 2.7    | <i>Barriers to Code Standardization</i>                                     | 21 |
| 2.8    | <i>Barriers to Code Reuse</i>                                               | 22 |
| 2.9    | <i>Eliminating the Barriers</i>                                             | 23 |

### THREE Implementing Effective Error Coding 25

|       |                                                      |    |
|-------|------------------------------------------------------|----|
| 3.1   | <i>Raise Your Expectations</i>                       | 25 |
| 3.2   | <i>Manage Errors Early</i>                           | 26 |
| 3.3   | <i>Code Errors as You Go</i>                         | 27 |
| 3.4   | <i>Anticipate Errors</i>                             | 31 |
| 3.5   | <i>Prevent Errors</i>                                | 31 |
| 3.6   | <i>Handle Errors</i>                                 | 32 |
| 3.7   | <i>Trap Errors</i>                                   | 32 |
| 3.8   | <i>Report Errors</i>                                 | 33 |
| 3.9   | <i>Avoid Assumptions</i>                             | 35 |
| 3.9.1 | <i>I Won't Ever Need to Use This Code Again</i>      | 36 |
| 3.9.2 | <i>I'm the Only One Who Will Touch This Code</i>     | 36 |
| 3.9.3 | <i>I Only Designed It for a Particular Situation</i> | 36 |

|             |                                                               |           |
|-------------|---------------------------------------------------------------|-----------|
| 3.9.4       | <i>General Coding Assumptions</i>                             | 36        |
| 3.10        | <i>Design Functions for Reuse</i>                             | 37        |
| 3.11        | <i>Reuse Error Coding</i>                                     | 37        |
| 3.12        | <i>Systematic Error Coding</i>                                | 38        |
| <br>        |                                                               |           |
| <b>FOUR</b> | <b>Explicit Coding</b>                                        | <b>39</b> |
| 4.1         | <i>Explicit Variable Usage</i>                                | 40        |
| 4.1.1       | <i>Always Use Option Explicit</i>                             | 40        |
| 4.1.2       | <i>Explicitly Type Variables</i>                              | 41        |
| 4.1.3       | <i>Avoid DefType Statements</i>                               | 43        |
| 4.1.4       | <i>Use Specific Data Types</i>                                | 44        |
| 4.1.5       | <i>Initialize All Variables</i>                               | 44        |
| 4.1.6       | <i>Use One Variable per Line</i>                              | 47        |
| 4.1.7       | <i>Use TypeName, VarType, and TypeOf</i>                      | 48        |
| 4.1.8       | <i>Use Enumerations</i>                                       | 48        |
| 4.2         | <i>Arguments</i>                                              | 49        |
| 4.2.1       | <i>Always Use ByVal or ByRef</i>                              | 49        |
| 4.2.2       | <i>Explicitly Type Arguments</i>                              | 50        |
| 4.2.3       | <i>Set Explicit Default Values<br/>for Optional Arguments</i> | 51        |
| 4.2.4       | <i>Validate All Arguments</i>                                 | 52        |
| 4.2.5       | <i>Use Named Arguments</i>                                    | 54        |
| 4.3         | <i>Arrays</i>                                                 | 54        |
| 4.3.1       | <i>Never Assume Lower Array Bounds</i>                        | 54        |
| 4.3.2       | <i>Don't Hard Code Array Bounds</i>                           | 56        |
| 4.3.3       | <i>Avoid Using Option Base</i>                                | 58        |
| 4.4         | <i>Coding Recommendations</i>                                 | 58        |
| 4.4.1       | <i>Always Include an Else</i>                                 | 58        |
| 4.4.2       | <i>Avoid Using Default Properties</i>                         | 59        |
| 4.4.3       | <i>Avoid Mixing Data Types in Expressions</i>                 | 60        |
| 4.4.4       | <i>Use Constants</i>                                          | 61        |
| 4.4.5       | <i>Avoid Operator Precedence</i>                              | 61        |
| 4.4.6       | <i>Check String Lengths</i>                                   | 62        |
| 4.4.7       | <i>Close All Open Objects</i>                                 | 64        |

- 4.4.8 *Set Objects to Nothing* 65
- 4.4.9 *Always Explicitly Turn Off Error Trapping* 65
- 4.4.10 *Never Assume Anything About the External World* 67
- 4.4.11 *Don't Cut and Paste* 67
- 4.4.12 *Use + and & Properly* 68
- 4.4.13 *Pseudo-Code* 68
- 4.4.14 *Set Properties at Run Time* 68
- 4.5 *Explicit Coding Is Fundamental* 69

## **FIVE Error Coding Mechanics 71**

- 5.1 *Error Coding Is Not a Given* 71
- 5.2 *Visual Basic Error Handling* 73
- 5.3 *Without Error Handling* 73
- 5.4 *The Error Handler* 74
- 5.5 *On Error Resume Next* 74
- 5.6 *Error Suppression* 74
- 5.7 *On Error Goto* 76
- 5.8 *Resuming Program Execution* 77
  - 5.8.1 *Resume* 77
  - 5.8.2 *Resume Next* 78
  - 5.8.3 *Resume Line* 79
- 5.9 *Multiple Error Handlers* 80
- 5.10 *Checking for Errors* 81
- 5.11 *Checking Err.Number* 83
- 5.12 *Handling the Error* 85
- 5.13 *Clearing the Error Object* 85
- 5.14 *Disabling the Error Handler* 87
- 5.15 *Scope of Error Handling* 87
- 5.16 *Error Bubbles* 92
- 5.17 *Errors within Errors* 93
- 5.18 *Changing Error Handlers* 95
- 5.19 *The Error Trap* 96
- 5.20 *The Error Trap Handler* 97

|      |                                                |     |
|------|------------------------------------------------|-----|
| 5.21 | Handling Errors In-Line                        | 98  |
| 5.22 | Raising Errors                                 | 99  |
| 5.23 | Error Trap Block versus In-Line Error Handlers | 100 |
| 5.24 | When to Use the Error Trap Block               | 100 |
| 5.25 | When to Use In-Line Error Handling             | 101 |
| 5.26 | Avoiding Error Handling Completely             | 103 |

## SIX Error Prevention 105

|        |                                                   |     |
|--------|---------------------------------------------------|-----|
| 6.1    | Types of Errors                                   | 106 |
| 6.1.1  | <i>Programmatic Errors</i>                        | 106 |
| 6.1.2  | <i>Logical Errors</i>                             | 107 |
| 6.2    | Types of Prevention                               | 108 |
| 6.3    | Preventing Coding Errors                          | 108 |
| 6.3.1  | <i>Think Long Term</i>                            | 108 |
| 6.3.2  | <i>Write for Others</i>                           | 109 |
| 6.3.3  | <i>Code Defensively</i>                           | 109 |
| 6.3.4  | <i>Code Offensively</i>                           | 111 |
| 6.3.5  | <i>Avoid Error Suppression</i>                    | 112 |
| 6.3.6  | <i>Elegance as a Prevention Technique</i>         | 113 |
| 6.3.7  | <i>Fool Me Twice, Shame on Me</i>                 | 113 |
| 6.3.8  | <i>Never Fix the Same Error Twice</i>             | 113 |
| 6.3.9  | <i>Reuse</i>                                      | 114 |
| 6.3.10 | <i>Standardize</i>                                | 115 |
| 6.3.11 | <i>Wrap System Functions</i>                      | 115 |
| 6.3.12 | <i>Don't Use Error Trapping for Prevention</i>    | 116 |
| 6.4    | Preventing User Errors                            | 117 |
| 6.4.1  | <i>The Three Cardinal Rules of Program Design</i> | 118 |
| 6.4.2  | <i>Be Explicit</i>                                | 119 |
| 6.4.3  | <i>Refine the Design</i>                          | 120 |
| 6.4.4  | <i>Make Your User Interface Unambiguous</i>       | 120 |
| 6.4.5  | <i>Make Your Messages Clear</i>                   | 122 |
| 6.4.6  | <i>Filter User Input</i>                          | 122 |
| 6.4.7  | <i>Validate User Input</i>                        | 123 |

|      |                                                |     |
|------|------------------------------------------------|-----|
| 5.21 | Handling Errors In-Line                        | 98  |
| 5.22 | Raising Errors                                 | 99  |
| 5.23 | Error Trap Block versus In-Line Error Handlers | 100 |
| 5.24 | When to Use the Error Trap Block               | 100 |
| 5.25 | When to Use In-Line Error Handling             | 101 |
| 5.26 | Avoiding Error Handling Completely             | 103 |

## SIX Error Prevention 105

|        |                                                   |     |
|--------|---------------------------------------------------|-----|
| 6.1    | Types of Errors                                   | 106 |
| 6.1.1  | <i>Programmatic Errors</i>                        | 106 |
| 6.1.2  | <i>Logical Errors</i>                             | 107 |
| 6.2    | Types of Prevention                               | 108 |
| 6.3    | Preventing Coding Errors                          | 108 |
| 6.3.1  | <i>Think Long Term</i>                            | 108 |
| 6.3.2  | <i>Write for Others</i>                           | 109 |
| 6.3.3  | <i>Code Defensively</i>                           | 109 |
| 6.3.4  | <i>Code Offensively</i>                           | 111 |
| 6.3.5  | <i>Avoid Error Suppression</i>                    | 112 |
| 6.3.6  | <i>Elegance as a Prevention Technique</i>         | 113 |
| 6.3.7  | <i>Fool Me Twice, Shame on Me</i>                 | 113 |
| 6.3.8  | <i>Never Fix the Same Error Twice</i>             | 113 |
| 6.3.9  | <i>Reuse</i>                                      | 114 |
| 6.3.10 | <i>Standardize</i>                                | 115 |
| 6.3.11 | <i>Wrap System Functions</i>                      | 115 |
| 6.3.12 | <i>Don't Use Error Trapping for Prevention</i>    | 116 |
| 6.4    | Preventing User Errors                            | 117 |
| 6.4.1  | <i>The Three Cardinal Rules of Program Design</i> | 118 |
| 6.4.2  | <i>Be Explicit</i>                                | 119 |
| 6.4.3  | <i>Refine the Design</i>                          | 120 |
| 6.4.4  | <i>Make Your User Interface Unambiguous</i>       | 120 |
| 6.4.5  | <i>Make Your Messages Clear</i>                   | 122 |
| 6.4.6  | <i>Filter User Input</i>                          | 122 |
| 6.4.7  | <i>Validate User Input</i>                        | 123 |

- 6.4.8 *Use Control Arrays* 125
- 6.4.9 *Select the Right Control* 125
- 6.4.10 *Wrap Controls* 125
- 6.5 **Form Preventative Habits** 126

## **SEVEN** Safe Coding Framework 127

- 7.1 **Reuse-Quality Routines** 127
- 7.2 **Safe Procedures** 128
- 7.3 **Safe Functions** 131
  - 7.3.1 *Ignore the Error* 131
  - 7.3.2 *Report the Error* 132
  - 7.3.3 *Pass Back the Error* 132
  - 7.3.4 *Return a New Error* 133
  - 7.3.5 *Add to Audit Trail* 134
  - 7.3.6 *Handle the Error* 135
- 7.4 **Safe Error Messages** 135
- 7.5 **Defensive Functions** 136
- 7.6 **Defensive Subroutines** 137
- 7.7 **Safe Classes** 138
- 7.8 **Reuse of SPF Procedures** 140
- 7.9 **Self-Contained Procedures** 140
- 7.10 **Code Blocks** 140
- 7.11 **Naming Conventions** 141
- 7.12 **Arguments** 141
- 7.13 **Limited Scope** 143
- 7.14 **Counter Variables** 144
- 7.15 **Revision Numbering** 144
- 7.16 **Reuse-Quality Documentation** 144
  - 7.16.1 *Procedure Headers* 145
  - 7.16.2 *A Sample Header* 146
  - 7.16.3 *Revision History Comments* 147
- 7.17 **Cleanup** 147
- 7.18 **Using the SPF** 148
- 7.19 **Implementing the Standard** 148

## **EIGHT    SPF Sampler    149**

- 8.1    General Structure    149**
- 8.2    Safe Error Utilities    151**
  - 8.2.1    Create a Safe Error Message    152*
  - 8.2.2    Count Errors in a Safe Error Message    153*
  - 8.2.3    Parse a Safe Error Message    154*
  - 8.2.4    Reporting a Safe Error Message    156*
- 8.3    Array Handling    157**
  - 8.3.1    Getting the Lower Array Bound    157*
  - 8.3.2    Getting Both Array Bounds    158*
  - 8.3.3    Getting an Array Count    159*
- 8.4    Type Conversion and Data Validation    160**
  - 8.4.1    Converting a String    161*
  - 8.4.2    Converting a Date    162*
  - 8.4.3    Converting a Number    163*
  - 8.4.4    Validating a Number    163*
- 8.5    String Handling    165**
  - 8.5.1    Safe Len Wrapper    165*
  - 8.5.2    SSN Format    167*
- 8.6    Forms and Controls    169**
  - 8.6.1    Determining If a Form Is Loaded    170*
  - 8.6.2    Unloading All Forms    171*
  - 8.6.3    Setting Focus    172*
  - 8.6.4    Making Sure It Is Safe to Resize    173*
  - 8.6.5    Copying a List Control    174*
- 8.7    Database Routines    175**
  - 8.7.1    Formatting SQL Strings    175*
  - 8.7.2    Checking the Cursor Position    177*
  - 8.7.3    Editing a Field    178*
- 8.8    Putting Safe Procedures to Work    180**

## **NINE Corporate Strategies 183**

- 9.1 Smart Coding Teams 183
- 9.2 Cooperative Competition 184
- 9.3 Developing Your Standard 187
- 9.4 Creating Safe Procedures 187
- 9.5 Motivational Catalysis 188
- 9.6 Certifying for Reuse 189
- 9.7 Sharing Certified Procedures 190
- 9.8 Using Hyper-Libraries 190
- 9.9 Rewarding Lasting Contributions 191
- 9.10 Code Review through Certification 191
- 9.11 Adaptive Development 192
- 9.12 Eliminating Nontechnical Barriers 193
- 9.13 You're Not There Yet! 194

## **TEN Program Architecture 195**

- 10.1 Modes of Program Failure 195
  - 10.1.1 *Failure to Complete* 195
  - 10.1.2 *Failure to Perform* 196
  - 10.1.3 *Failure to Maintain* 196
- 10.2 Beadwork Programs 196
- 10.3 Maintainability 197
- 10.4 Maintenance Nightmares 199
- 10.5 Implicit Business Logic 200
- 10.6 Architectural Dimensions 201
- 10.7 Lobes and Layers 201
- 10.8 Universal Layered Architecture 204
  - 10.8.1 *The User Layer* 204
  - 10.8.2 *The Business Layer* 204
  - 10.8.3 *The Data Layer* 205
  - 10.8.4 *The User Connection Layer* 205
  - 10.8.5 *The Database Layer* 206
  - 10.8.6 *The Data Connection Layer* 206

- 10.9 Reusable Layers 206
- 10.10 Layered Flow 207
- 10.11 Layering versus Binding 209
- 10.12 Layering versus Classes 209
- 10.13 Layering versus Tiers 209
- 10.14 Layer Packaging 210
- 10.15 Deploying Layers 210
- 10.16 Benefits of Layering 211
- 10.17 Layering Case Study 211
- 10.18 Datasets 213
- 10.19 Insulation from Technology Changes 214
- 10.20 Implementing a Layered Application 215

## **ELEVEN** Designing a Layered Application 217

- 11.1 The Database 218
- 11.2 Planning Your Data Layer 219
- 11.3 Mapping Your Controls 221
- 11.4 Creating Your Layers 222
- 11.5 Pseudo-Coding the User Layer 222
  - 11.5.1 *Form\_Load* 222
  - 11.5.2 *Students\_Click* 224
  - 11.5.3 *Customers\_Click* 225
  - 11.5.4 *Editing Fields* 226
  - 11.5.5 *Saving Changes* 226
  - 11.5.6 *Adding and Deleting* 227
- 11.6 Pseudo-Coding the Business Layer 228
  - 11.6.1 *Showing Datasets* 228
  - 11.6.2 *Showing Calculated Fields* 230
  - 11.6.3 *Changing Data in Datasets* 232
- 11.7 Methods in the Data Layer 234
- 11.8 Methods in the User Connection Layer 234
- 11.9 Methods in the Data Connection Layer 235
- 11.10 Cool Features of Layered Applications 235

- 11.11 Creating a Safe Layered Library 236
- 11.12 Using Layer-Wrapped Controls 237
- 11.13 Take It from Here 238

## **TWELVE Accomplishing Your Mission 239**

- 12.1 Getting the Panoramic View 239
- 12.2 Assessing Your Success 242
- 12.3 Taking Your Next Steps 243
- 12.4 Keep the Ball Going! 244

## **APPENDIX A Naming Conventions 245**

## **APPENDIX B SPF Checklist 249**

## **APPENDIX C Certification Rating Sheet 253**

## **INDEX 255**