

Index

A

- abbreviations, naming conventions, 244–245**
- access, building data model, 66**
- accessibility, 99–103**
 - adding new controls, 226
 - adding tooltips and descriptions, 101–102
 - adding visual elements, 100–101
 - coloring techniques, 102–103
 - testing application colors with Vischeck, 571–574
 - using voice version of CAPTCHA, 134
- Accessibility for Everybody: Understanding the Section 508 Accessibility Requirements, 99***
- AccessibleDescription property, controls, 101**
- AccessibleName property, controls, 102**
- AccessibleRole property, controls, 102**
- acronyms, in naming conventions, 244–245**
- ACTF (Eclipse Accessibility Tools Framework), 574**
- activations, Visio UML, 73**
- activity diagrams, Visio UML, 72**
- Adaptive Software Development (ASD), 52**
- Add Counters dialog box, System Monitor, 110–112**
- Add New Test dialog box, 304**
- Add Project Output Group dialog box, 334–335**
- Add Special Folder menu, Setup Project template, 333**
- Add-In Manager, 168**
- add-ins**
 - defined, 165
 - for extended functionality, 168–169
 - folders for, 159
 - removing unused, 166
 - researching Visual Studio, 181
- Add/Remove Columns dialog box, 309–310**
- Administrative setting, XCopy, 419, 426**
- administrators**
 - documentation for, 269–270
 - issues surrounding ownership of data, 405–406
 - overcoming poor practices, 133
 - performance triangle characteristics and, 364–367
 - sending alerts, 148–149
 - verifying resource availability, 122–123
- AdventureWorks2008 database, 462**
- adware, 342**
- AES (Advanced Encryption Standard), 144**
- aggregate controls, 216**
- agile programming, 46–54**
 - Adaptive Software Development, 52
 - benefits of, 47–48
 - combining methods, 53–54
 - Crystal Clear, 49–50
 - deficiencies of, 48
 - defined, 17–18
 - Extreme Programming, 50–52
 - Feature Driven Development, 52–53
 - Scrum, 48–49
 - understanding, 46–47
- Agile Software Development with Scrum (Takeuchi and Nonaka), 48***
- alerts, sending administrator, 148–149**
- aliases, Visual Studio commands, 174**
- All Languages folder, Text Editor, 157**
- Altova XML Spy, 420**
- AlwaysCreate property, application deployment, 335**
- ampersand (&), 100–101**
- analysis, incremental/iterative model, 43**

anonymous methods, C# support, 8
AppDeploy Repackager, 324
Appearance tab, System Monitor, 114
application counter, 115–119
application lifecycle, 17–18
choosing model, 35–36
design strategy, 54
using agile programming. See agile programming
using evolutionary model, 41–42
using incremental/iterative model, 42–43
using RAD model, 43–45
using spiral model, 36–37
using throwaway model, 40–41
using waterfall model, 38–40
application lifecycle, stages, 18
building development team, 19–21
creating design, 24–26
debugging and testing, 29–31
defining specification, 21–22
deployment, 33–34
determining whether concept will work, 22–23
developing concept, 18–19
enhancing reliability and security, 32–33
implementing design, 27–28
retirement of application, 35
support and maintenance, 34–35
testing design, 26–27
testing user requirements, 31–32
writing application, 28–29
application pessimism, 392
Application setting, XCopy, 419, 425
Application.Exit() method, 258–259
applications
breaking, 396–398
configuring with Windows PowerShell, 180
scripting, 184–185
windowed, 83–84
archives
recording software release date in, 547
resource and tool, 545–546
arrays, LINQ queries, 451–453
ASD (Adaptive Software Development), 52
AssemblyInfo.CS file, 204

associations, Visio UML, 72
attributes
control, 220
custom, 262
as type of comment, 271
automated compiles, 170–174
automated installations, 343–344
automated testing, 408
Autos window, 194–195, 317
AutoTrader, 17

B

backups, for encryption, 413
batch files, 169
bathtub model of reliability, 128
Beck, Kent, 50
behavior modification, properties, 220
beta cycle, feedback during, 548
Big Brother syndrome, 405–406
binary files, saving settings in, 127
binary searches, of serialized XML data, 436
black box approach, Visio UML component diagrams, 73
breaking into applications, 407
checking for the impossible, 409–410
creating lessons-learned journal, 397–398
relying on internal testers, 407–409
during testing, 396–397
understanding of interactions, 410–411
using third-party, 409
Breaking indicator, running code analysis, 400
btnStartStop_Click() method, adding counters, 118
buffer overrun technique, 138
Build and Run subfolder, Projects and Solutions, 157
build process, using FxCop during, 581–582
build-and-fix (throwaway) model, 40–41
Business Intelligence folder, Options dialog box, 157
business modeling, RAD model, 44–45

C**C#**

C/C++ syntax and, 5–6
 coding support in, 7–8
 complexities of C#, 9
 creating scripts, 178–179
 debugging, 7
 as developer favorite, 8
 disadvantages of, 9–10
 IntelliSense and, 7
 languages used with, 10–11
 limitations of using code snippets with, 167
 reliability concerns, 9
 unsafe code capability of, 6–7
 using expressions, 179, 185
 using numbers and special characters with, 245

C++ language, with C#, 10**CA numbers, code analysis, 398–400****CAB deployment method, 340–341****calculated fields**

displaying serialized XML data, 439
 using XML data in reports, 421–422

calculated values, LINQ queries, 453–454**camel case, 242–243****candidate list, 57–59****Caphyon Advanced Installer, 324****CAPTCHA (Completely Automatic Public Turing Test to Tell Computers and Humans Apart), 133–134****CARFAX, 17****Cascading Style Sheets (CSS), HTML Designer folder, 158****catastrophic failures, 122****C/C++ syntax, 5–6****CDs, for application deployment, 342****Certified Information Systems Auditor (CISA), 409****Certified Information Systems Security Professional (CISSP), 409****characters**

controlling use of unwanted, 140–141
 disallowing certain, 135

chickens, Scrum, 49**CISA (Certified Information Systems Auditor), 409****CISSP (Certified Information Systems Security Professional), 409****class candidates, 57–59****class concepts, object models, 56–60****Class Coupling, code metric results, 401****Class View, Object Test Bench, 236****classes**

adding external, 61–62
 choosing between structures and, 59
 choosing from candidates, 58–60
 components vs., 228–229
 considering trade-offs, 62–63
 creating data, 67
 creating user, 70
 developing internal, 60–61
 improving design of, 63–64
 testing using Object Test Bench, 235–238
 using standard exception, 286–294

Clear All Panes, Microsoft PowerCommand, 563**Clear Recent File List, Microsoft PowerCommand, 563****Clear Recent Project List, Microsoft PowerCommand, 563****Clippy, 97****clock cycles, and processors, 488****Close() method, XML, 427, 429****Close All, Microsoft PowerCommand, 563**

CLR (Common Language Runtime)
 automatic length checking of, 138
 thread management, 488–489

clutter

creating archive to reduce, 544–546
 of resources and tools on hard drives, 543

cmdlets, using Windows PowerShell, 180**COBOL programmers, 10, 35****code**

adding counter to application, 117–118
 calculating metrics, 400
 checking security holes in, 149
 configuring IDE, 175–176
 generating from UML, 75–77
 incremental/iterative model, 43
 for multithreaded applications, 499–502

code (continued)

- running analysis of, 398–399
- snippets. See snippets
- starting database of reusable, 61
- third party vendor advantages, 568
- Code Coverage Results window, 312–314**
- Code Snippets Manager, 167**
- code-and-fix (throwaway) model, 40–41**
- CodeDOM (Code Document Object Model), 185**
- coding application, 241–263**
 - calculating code metrics, 400–401
 - command line functionality. See command line
 - for components and controls, 239
 - for deployment, 345–346
 - for documentation, 280
 - error handling, 300
 - exiting properly, 254–260
 - for maintenance, 356
 - naming conventions, 241–246
 - running code analysis, 398–400
 - for scripting, 191–192
 - style requirements, 26, 28
 - summary, 262
 - for testing, debugging and quality assurance, 321
 - using custom attributes and exceptions, 262
 - using custom features, 261
 - using multithreading, 490
 - for viewing data in IDE, 209
- collaborate phase, ASD programming, 52**
- collaboration diagrams, Visio UML, 72–73**
- Collapse Projects, Microsoft PowerCommand, 563**
- color**
 - changing Environment folder settings, 157
 - choosing from System tab, 96
 - nudging users with, 97
 - Show Code Coverage Coloring, 314
 - visual environment and, 87
- color, and accessibility, 99 requirements**
 - good techniques for, 102–103
 - testing in application using Vischeck, 571–574
- color coding**
 - of blocks in object model, 62
 - of debugging windows, 317
 - of XML data in XML visualizer, 199–200

colorblindness (deuteranomalía)

- good techniques for, 102–103
- other checkers for, 574
- testing application using Vischeck, 571–574

columns, Test View, 309–310

COM (Component Object Model), 337

COM+ application, 11

Command, Windows PowerShell, 186–187

command line, 169–175, 246–253

- accessing, 169–170
- adding switches, 247–252
- executing commands, 174–175
- experimenting with F# at, 481–485
- running application, 175
- testing, 252–253
- types of switches, 246–247
- using as interface, 90–91
- using switches, 170–174, 246
- using Windows PowerShell. See Windows PowerShell

/Command line switch, Visual Studio commands, 174–175

Command Window

- asking IDE questions using, 318
- defined, 194
- viewing data in IDE, 196–197

command-driven interface, setup programs using, 94

commands

- customizing and adding to Tools menu, 164–165
- Microsoft PowerCommands, 563–565

Commands tab

- Microsoft PowerCommands, 562
- working with menus, 162
- working with toolbars, 161–162

comments

- C# multiline, 7
- creating, 267
- kinds of, 270–273
- placing correctly, 273–275
- throwaway code and, 41

Common Language Runtime (CLR)

- automatic length checking of, 138
- thread management, 488–489

communications

- developing interaction strategy, 95–99
- as XP core value, 51

Community Technology Preview (CTP), 477**compatibility**

- checking for problems of, 133
- third party vendor issues, 568

competitive spirit, in testing, 396**complaint process, design implementation, 28****Completely Automatic Public Turing Test to Tell Computers and Humans Apart (CAPTCHA), 133–134****Component Designer, 213****Component-Oriented Programming (COP), 228****Component-Oriented Programming (Wang and Qian), 228****components, 211–212, 228–239**

- adding new code, 230–232
- coding application, 239
- considerations for building, 228–229
- controls vs., 213–215
- defining new project, 229–230
- testing new, 233–235
- testing using Object Test Bench, 235–238
- Visio UML deployment diagrams, 73

concept, application

- determining viability of, 22–23
- developing, 18–19

conceptual complexity, of multithreaded applications, 491**concurrency**

- F# applications vs. imperative languages, 477–478
- using multithreaded applications for, 487, 492

Condition property, deployment setup, 335–336**CONFIG file, saving settings in, 126****connections**

- dynamic vs. static, 66
- LINQ to DataSet, 460–461
- testing LINQ to SQL, 462–463
- in Visio UML deployment diagrams, 73

consistency

- error-handling issues, 285, 296–298
- improving using FxCop. See FxCop

CONSOLE_SCREEN_BUFFER_INFO structure, 536–537, 538–539**constructor, creating for visualizer window, 205–206****consultants, reliability and, 394****content**

- basic application deployment, 334–337
- of self-modifying reports, 515
- of user reports, 511–512
- of windowed applications, 83–84

context-sensitive help (/A/?) switch, 246**control, keeping user in, 96****controlled installations, 343****controls, 211–212**

- accessibility properties for, 101–102
- building derived, 216–223
- building new, 225–227
- building UserControls, 223–225
- coding application, 239
- components vs., 213–215
- creating user classes, 70
- minimizing user-typed entries, 134–135
- role-driven interface, 95
- sorting serialized XML data entries, 437
- testing for best user experience, 135
- testing using Object Test Bench, 235–238
- types of, 215
- using consistent design for interface, 98
- using LINQ with, 455–457
- visual element considerations, 100–101

COORD, 535–537**coordinate, screen**

- creating for P/Invoke, 535–537
- declaring function calls, 537–538

COP (Component-Oriented Programming), 228**Copy As Project Reference, Microsoft PowerCommand, 563****Copy Class, Microsoft PowerCommand, 563****Copy Path, Microsoft PowerCommand, 563****Copy References, Microsoft PowerCommand, 563**

costs

- considering when choosing tools, 550
- custom features and, 261
- products of third party vendors and, 568–569

Counter, Add Counters dialog box, 110

counters

- defined, 109
- defining application, 115–119
- measuring speed with, 109
- selecting another view, 114–115
- setting graph properties, 112–114
- testing release functionality, 325
- using, 110–112
- viewing standard, 109

courage, as XP core value, 52

cowboy programming, agile vs., 47–48

CRaG Systems tutorial, 72

Create New Test List dialog box, 312

Create Shortcut wizard, 170

CreateDT() method, 460

CreateQuickData() method, 460

Crystal Clear, 49–50

Crystal Clear, A Human-Powered Methodology for Small Teams (Cockburn), 49

Crystal Reports, 514–515

CSS (Cascading Style Sheets), HTML Designer folder, 158

CTP (Community Technology Preview), 477

Custom Actions view, 339

custom attributes, 262, 273

custom error handling, 294–296

custom exceptions, 262

custom features, 261

custom toolbars, 160

custom visualizers, 202–209

customers

- participation in evolutionary model, 41
- participation in incremental/iterative model, 42–43
- performance triangle characteristics affecting, 364–367

customizing, RibbonX interface, 129

Cyclomatic Complexity, 401

D

data

- checking length of, 138–140
- checking range of, 136–138
- defining elements for XML storage, 422–424
- encrypting and decrypting, 413–414
- hiding from view, 142–144
- issues surrounding ownership of, 405–406
- keeping out of hands of users, 404–405
- saving, 124–125
- viewing error log, 349–352

Data Encryption Standard (DES), cracking, 144

data models

- building, 64–67
- RAD, 44–45

data sources, LINQ

- for built-in LINQ, 454–455
- creating queries by joining multiple, 450–453
- creating simple queries, 449–450
- LINQ interaction with any, 454
- SQL vs. LINQ queries, 448
- testing Visual Studio connection for LINQ to SQL, 462–463

data sources, verifying, 142

Data Sources window, 181

data stores, 93

data structures, with P/Invoke

- creating, 535–537
- defining function calls, 537–538
- enumerating standard console handles, 535
- using within managed code, 538–540

Data tab, System Monitor, 113–114

Database Tools folder, Options dialog box, 157

databases

- philosophies of agile management, 48
- reliability-related, 395
- for reusable code, 61

DataRead.Deserialize() method, XML, 429

DataSet control, LINQ to DataSet, 457–461

Dataset Designer, 181

DataSet visualizer, 201–202

DEBUG constant, 250–251

debugging

- asking IDE a question, 317–318
- in C#, 7
- deficiencies of multithreaded applications, 492
- obtaining bugs from users, 319
- testing and, 29–31, 308
- using command line switches, 250–251, 253
- using Command Window, 197
- using Watch window, 196–197
- viewing memory content, 318
- windows, 316–317
- Windows PowerShell and, 191

Debugging folder, Options dialog box, 157–158

decryption, 413–414

default data input switches, 247

DefaultLocation property, 335

deferred evaluation, LINQ, 456

defining errors, 348

deleting, serialized XML data entries, 436–437

Department of Defense (DOD), and waterfall model, 38

Dependencies property, deployment setup, 337

deploying applications, 33–34, 323–324

- basic setup. See deploying applications, basic setup
- choosing media type, 342
- coding application, 345–346
- evaluating results, 344–345
- performing deployment, 343–344
- types of deployment. See deploying applications, choosing type
- using CAB method, 340–341
- using Setup Wizard method, 339–340
- using XCopy method, 330–331
- Visio UML diagrams, 73

deploying applications, basic setup, 331–340

- adding setup content, 334–335
- adding special folders, 333–334
- building installation application, 337–338
- creating project, 332–333
- modifying details, 338–339
- setting installation properties, 335–337
- using Setup Wizard, 339–340

deploying applications, choosing type, 324–329

- enterprise deployment, 326–327
- local deployment, 325–326
- major vs. minor releases, 327–328
- shrink-wrap deployment, 327
- testing release, 324–325

Depth of Inheritance, code metric results, 401

derived controls, 215

derived controls, building, 216–223

- adding code, 218–220
- adding to another application, 222–223
- checking functionality, 221–222
- considerations for developing, 216
- defining project, 216–218

DES (Data Encryption Standard), cracking, 144

descriptive labels, nudging users with, 97

deserialization, defined, 417

design, 55–56

- application lifecycle development, 54
- application speed, 119–120
- building data model, 64–67
- considering user requirements, 67–70
- creating, 24–26
- developing, 77
- high-level, 24–25
- implementing, 27–28
- incremental/iterative model, 43
- language environment, 4, 15
- low-level, 25–26
- object models. See object models, creating
- reliability, 130
- security, 150–151
- testing, 26–27
- UML, 70–77
- user interface, 103
- waterfall model, 39

Deuteranope option, Vischeck, 573

developers

- application reliability and, 365–366
- application support and maintenance, 34–35
- C# as favorite among, 8
- creating reports, 508–510, 511–514
- documentation for, 267–268

development

- issues surrounding ownership of data, 405–406
- multiple language experience of, 4
- not falling for fad development techniques, 46
- productivity of, 29
- reliable, 396

development

- custom features adding time to, 261
- performance triangle and. *See* performance triangle
- of reliability. *See* reliability, application
- of security. *See* security
- trade-offs in object model, 63

development teams

- building, 19–21
- considering skills and experience of, 13–14
- creating toolbox for, 543, 550–551
- implementing design, 27
- including technical writers in, 369
- inventory of language resources for, 13
- QA team vs., 319–320
- security design, 149–150
- transition to development phase. *See* performance triangle

DevEnv command, 169

Device Tools folder, Options dialog box, 158

dialog boxes

- creating, 80–82
- creating using `ExternMessageBox()`, 527–528
- using tabbed interface, 85
- wizard-driven interface relying on, 93

DialogDebuggerVisualizer class, 203–204, 207

DialogResult property, exiting application, 255

directives, P/Invoke, 522

disabled persons. *See* accessibility

Disk Operating System (DOS), 330

displayMsg, 482, 483–484

Dispose() method, 255

DoCallback() method, 502

documentation, 265

- administrative needs, 269–270
- agile programming deficiency, 48
- alternative uses for, 279–280
- application maintenance, 35
- candidate list, 57–58

- coding application, 280
- command line interface, 91
- comments, 270–275
- design implementation, 27
- developer needs, 267–268
- low-level design stage, 25–26
- of original design document, 267
- reliability testing, 398
- resources and tools archive, 545–546
- resources and tools usage requirements and policies, 551
- self-documenting code, myth of, 265
- user needs, 268–269
- as view to document, 266–267

DOCX report format, 515–517

DOD (Department of Defense), and waterfall model, 38

DoMonitoring() method, 497

DOS (Disk Operating System), 330

007 Spy Software, 146

DVDs

- deployment using, 342
- deployment using distributed, 33
- documentation for users using, 268

dynamic connections, 66

E

Eclipse Accessibility Tools Framework (ACTF), 574

Edit Project File, Microsoft PowerCommand, 563–564

editing

- existing serialized XML data entries, 433–434
- macros, 168
- XML files using XML Notepad 2007, 558–561

efficiency, multithreaded applications, 490

electronic media, reports, 516

e-mail

- developing rapport with report users, 514
- sending administrator alerts, 148–149

Email CodeSnippet, Microsoft PowerCommand, 564

employee monitoring, 145–146

- EnableRaisingEvents property, event log, 148**
- encryption, 142–144**
 - security and, 413–414
- enterprise deployment, 326–327**
 - evaluating results of, 344
 - performing, 343
 - shrink-wrap deployment vs., 327
- entity classes, LINQ to SQL, 464–467**
- entity-relationship (ER) diagram, 65**
- enumerations**
 - checking data range, 136–137
 - using with P/Invoke, 522–525, 535
- environment**
 - application speed and, 106
 - making changes carefully to, 431
 - using XML data to modify, 430–431
- Environment folder, Options dialog box, 157**
- environment variables, F#, 481**
- Environment.Exit() method, 255**
- Environment.FailFast() method, 256–258**
- equations, functional language, 477–478**
- ER (entity-relationship) diagram, 65**
- errant input, eliminating, 133–141**
 - checking all user input, 135
 - checking data length, 138–140
 - checking data range, 136–138
 - keeping unnecessary characters controlled, 140–141
 - using correct control or component, 134–135
- error handling, 283–284**
 - assuming nothing when, 284–285
 - consistency in, 296–298
 - customizing, 294–296
 - defining errors, 348
 - importance of specific, 285–286
 - low-level application, 526, 529–534
 - self-healing applications and, 298–300
 - using FxCop, 576–578
 - using standard exception classes, 286–294
- Error List window, 398–400**
- Error Logging class, 349–352**
- error logs, 347**
 - creating, 347–353
 - deploying patches, 356
 - prioritizing maintenance activities, 355
 - providing support with, 353–354
 - testing maintenance actions, 356
 - verifying maintenance requirements, 355–356
 - viewing output, 352–353
- Error Lookup utility, 531**
- ERRORLEVEL, 256**
- evaluation**
 - of deployment results, 344–345
 - LINQ, 456
- event logs, 147–148**
- events**
 - adding names to object model, 61
 - exceptional, 285
 - new control considerations, 226
- evolutionary model, 41–42**
- Exception keyword, nonspecific error handling, 285–286**
- exceptional events, 285**
- exceptions**
 - correct use of standard, 286–294
 - creating new, 294–295
 - customizing, 262
 - describing special conditions using, 424–425
 - logging data on, 319
 - throwing, 300
 - using multiple levels, 295–296
- exclusive command line switch (/A), 247**
- exiting applications, 254–260**
 - with Application.Exit() method, 258–259
 - with P/Invoke, 259–260
 - with System.Environment, 255–258
 - when closing form is not enough, 254–255
- Expand utility, CAB, 341**
- expert system techniques, 298–299**
- expert users, 56**
- Explain, Add Counters dialog box, 110**
- exported usage documentation, 267**
- Expression field, visualizers, 198, 201**
- expressions, C#, 179, 185**
- eXtensible Markup Language. See XML (eXtensible Markup Language)**

eXtensible Stylesheet Language Transformation.

See **XSLT (eXtensible Stylesheet Language Transformation)**

external classes, adding, 61–62

external functions. See P/Invoke, calling external functions

external threats, identifying, 394–395

external tools, adding, 164–165

ExternMessageBox() function, 527–528

Extract Constant, Microsoft PowerCommand, 564

Extreme Programming (XP), 50–52

F

F# language, 477–478

adding commands to path, 481

as beta product, 477

C# using, 11

creating application, 484–486

defining basic script, 482

functional language of, 4, 477–478

installing, 480–481

interacting with, 483–484

obtaining support for Visual Studio 2008, 479–480

running application, 482–483

factories, class, 63

failure points, 393–395

security issues creating, 405

/FD command line switch, 253

FDD (Feature Driven Development), 52–53

Feature Driven Development (FDD), 52–53

feature sets, 56–57

features

recommended by users, 321

working with custom, 261

feedback

developing rapport with vendors, 548

from QA team, 320–321

as XP core value, 52

File Extension subfolder, Text Editor, 157

file processing thread, multithreaded applications, 494–495, 501–502

File System view, deployment setup, 333–334, 338

File types view, deployment setup, 338

FileExists() method, 472

FillConsoleOutputCharacter() function, 537–540

filters

monitoring using, 145–146

report, 512

using where keyword in LINQ queries, 447–448

finally statement, error handling, 297–298

FindStr utility, P/Invoke, 523

fired employees, and security, 133

FishNet Security, 409

Fixed3D setting, FormBorderStyle, 81

FixedDialog setting, FormBorderStyle, 81

FixedSingle setting, FormBorderStyle, 81

FixedToolWindow setting, FormBorderStyle, 82

flexibility, of third party products, 568

Flush() method, XML, 427

folders, application deployment, 333–337

fonts

accessibility requirements, 99

changing settings in Environment folder, 157

keeping user in control, 96

FormatMessage() function, 531–534

formats, report, 512, 515–517

FormBorderStyle property options, dialog boxes, 81–82

Form.Close() method, 254

FormClosing event handler, XML, 427–428

forms

uncluttering appearance of, 70

visual element considerations, 100–101

forms, types of, 80

command line as interface, 90–91

dialog boxes, 80–82

MDI windowed application, 84–85

placing application icon in Notification Area, 89

RibbonX applications, 87–88

SDI windowed application, 82–83

specialty skinned and free-form applications, 86–87
tabbed interface, 85

FORTRAN programmers, 10

free-form applications, 86–87

Freeze Display, 112

from operator, LINQ queries, 447–448

FromUrl() method, XmlMappingSource, 444–445

.FS extension, 482

FSC utility, 482–483

FSI utility, 482–483

functional language, 11

as basis of XSLT, 478

F# as, 4, 477–478

FxCop, 574–582

analyzing results with GUI, 576–577

obtaining, 575–576

repairing errors, 577–580

rules, 580–581

using during build, 581–582

G

GAT (Guidance Automation Toolkit), 158

General folder, Microsoft PowerCommands, 562

General tab, System Monitor, 112–114

Generate dialog box, 76

generic tests, 305

GetConsoleScreenBufferInfo() function, 537–538, 539

GetStdHandle() function, 537–538, 539

goals, scripting application, 184

Google search

for resources and tools, 557–558

using with P/Invoke, 524

government, security regulations, 367

graph view, 112–114

groups, 157–159

GUI

FxCop, 576–577

using command-line vs., 90

Guidance Automation Toolkit (GAT), 158

H

hand checks, security, 149–150

hard drives, getting rid of clutter on, 544–545

hardware

expecting the unexpected, 128

minimizing/disabling unneeded, 143

multithreaded application benefits, 491

requirements for application speed, 119

verifying resource availability, 122

hashes, transferring data using, 144

Haskell language, 11

hearing problems. See accessibility

help (/?) command-line switch, 246

Help button, designing, 98

help files

adding command line switches for, 248–250

creating user, 69

DataSet visualizer, 201

Sandcastle, 277–279

Hidden property, deployment setup, 337

hiding. See information hiding

high speed applications, 107–109

high-level design, 24–25

Highlight button, System Monitor, 112

histogram view, 114–115

HTML Designer folder, 158

HTML visualizer, 200–201

human testing, 315–316

Hungarian notation, 242

I

IBM Rational AppScan, 150

IBM's aDesigner, 574

icons

consistency in design for interface, 98

creating Notification Area, 89

Toolbars tab options, 161

IDE (Integrated Development Environment)

asking question about code, 317–318

command line switches for, 171–173

IDE (Integrated Development Environment), customizing, 155–176

add-ins, 165, 168–169
coding application, 175–176

IDE (continued)

configuring using Visual Studio Options, 156–159
external tools, 164–165
macros, 165, 168
snippets, 165–167
toolbars and menus, 159–163
Visual Studio command line, 169–175

IDE (Integrated Development Environment), viewing data in, 193–194

coding application, 209
creating custom visualizer, 202–209
need for visualizers, 198
obtaining third-party visualizers, 202
using Autos window, 194–195
using Command Window, 196–197
using DataSet visualizer, 201
using HTML visualizer, 200–201
using Locals window, 195–196
using text visualizer, 198–199
using Watch window, 196–197
using windows, 194
using XML visualizer, 199–200

IF ERRORLEVEL, exiting applications, 256

image files, testing Vischeck on, 527–573

Immediate window, 197

imperative languages, 477

incremental/iterative model, 42–43

information hiding, 142–144

C# and, 9
using RibbonX, 87–88, 129
using selective and partial, 404

information-driven user interface, 92–93

infrastructure, verifying resource availability, 122–123

inheritance

calculating code metrics, 401
object relationship vs., 63

INI file, saving settings in, 126

InitializeComponent() method, XML, 429

input

creating team toolbox using developer, 551
eliminating errant. See errant input, eliminating error handling by checking, 284
functionality of new controls, 226
for manual tests, 305
nontechnical, 212
switches for data, 247

Insert GUID Attribute, Microsoft PowerCommand, 564

Insert menu, XML Notepad, 560

Instance, Add Counters dialog box, 110

integration, waterfall model, 40

IntelliSense, 7

interactions, computer security and, 410–411

interfaces

designing interactive, 97–98
F# interactive, 483–484
prefixing with *I*, 244

internal classes, 60–61

internal testers, 407–409

Internet, verifying resource availability on, 122

Internet Archive Wayback Machine site, 558

Intranet, verifying resource availability on, 122

inventories

team language, 13
tool, 12

IronPython, 10

Isoware, 128

iterations, C# support for, 7

J

Java syntax, 5–6

JAWS (Job Access With Speech), 101

JNBridge, 11

Job Access With Speech (JAWS), 101

Join() method, LINQ, 452–453

join operator, LINQ, 448, 450–453

journal, and reliability, 397

K**keyboard, Office Tools folder settings, 158****keyboard shortcuts, 161–162****KeyGhost, 146****keywords, LINQ**

- creating queries with, 446–447
- defining simple queries, 450
- working with from and select operators, 447
- working with orderBy operator, 448–449
- working with where operator, 447–448

KISS (Keep It Simple, Stupid) acronym, in multithreading, 487**Knowledge Base, Microsoft, 556****L****lambda expressions**

- defining simple LINQ queries, 450
- sorting serialized XML data entries, 438
- using LINQ to DataSet, 461
- using LINQ to Objects, 456

language environment, 3

- C# benefits, 5–8
- C# deficiencies, 8–10
- creating inventory, 12
- design strategy, developing, 15
- design strategy, elements of, 4
- developing with multiple languages, 10–11
- functional vs. imperative, 11
- gathering resources, 12–15
- multiple platform scenarios, 11
- understanding, 3

Language INtegrated Query. See LINQ (Language INtegrated Query)**Launch Conditions view, 339****LayoutKind parameter, 536****learn phase, ASD, 52****legalities**

- of employee monitoring, 145–146
- ownership of data, 405
- report, 513
- telling users about monitoring policies, 412

Legend, System Monitor, 113**let keyword**

- F# script, 482
- LINQ queries, 448, 453–454

libraries, as source of external classes, 61–62**licensing terms**

- finding software with favorable, 546
- FxCop installation, 575

lifecycle, applications. See application lifecycle**lifecycle models, 35–45**

- choosing, 35–36
- developing design strategy, 54
- using evolutionary model, 41–42
- using incremental/iterative model, 42–43
- using RAD model, 43–45
- using spiral model, 36–37
- using throwaway model, 40–41
- using waterfall model, 38–40

Lines of Code, 401**LINQ (Language INtegrated Query), 441–476**

- accessing data with, 64
- built-in providers, 454–455
- comparing SQL to, 442–443
- creating queries, 449–454
- defined, 441
- defining basic operators, 446–449
- improving multithreaded application performance, 487, 505
- LINQ to DataSet, 457–461
- LINQ to Objects, 455–457
- LINQ to SQL, 462–467
- LINQ to XML, 467–472
- non-query uses of, 446
- providers, 442
- querying documentation file, 280
- scripting using, 180
- serialized XML data entries, searching, 436
- serialized XML data entries, sorting, 437–439
- third-party providers, 472–475

LINQ to DataSet, 457–461

- creating test tables, 458–460
- exploring connection, 460–461
- outputting results, 460

LINQ to Objects, 455–457

- using, 454–455
- using LINQPad to create queries, 583–585

LINQ to SQL, 462–467

- adding code, 467
- creating project, 466–467
- generating Northwind entity classes and XML mapping files, 464–466
- testing Visual Studio connection, 462–463
- using LINQPad to create queries, 583–585

LINQ to XML, 467–472

- building XML document, 468–470
- creating project, 468
- loading XML document, 471–472
- saving XML document, 470–471
- using LINQPad to create queries, 583–585

LINQPad, 582–585

ListBox control, XML, 437

ListDLLs tool, 554

load tests, 305

loading

- data in XML format, 428–431
- XML document, LINQ to XML, 471–472

local data, hiding, 143

local deployment, 325–326

- evaluating results of, 344
- performing, 343

locality

- database access and, 66
- placing data in protected, 144
- for saving data in XML format, 425–426

Locals window

- debugging with, 317
- defined, 194
- viewing data in IDE, 195–196

LocalTestRun.TESTRUNCONFIG file, 313

logging

- exception data, 319
- implementing, 146–147
- testing release functionality, 325
- using Event Log, 147–148

low-level application elements, 519

- creating basic data structures, 535–537
- defining function calls, 537–538

- defining P/Invoke, 519–521
- enumerating standard console handles, 535
- understanding, 25–26
- using data structures within managed code, 538–540

low-level application elements, calling external functions, 521–522

- adding required directives, 522
- creating external function reference, 525–526
- error handling, 529–534
- function data calls, 526–527
- implementing design, 540
- within managed code, 527–529
- structures, enumerations and constants, 522–525

M

m_ prefix (local memory), 244

Machine, Add Counters dialog box, 110

Machine setting, XCopy, 419, 425

Macro Explorer window, 168

macros

- automating tasks using, 168
- defined, 155, 165
- executing from command line, 175
- removing when no longer used, 166

Main() method, 251–252

Mainsoft, 11

Maintainability Index, code metric results, 400–401

maintenance, 347–358

- as application lifecycle stage, 34–35
- coding application, 356
- creating error logs, 347–353
- creating support logs, 354
- custom features for, 261
- deploying patches, 356
- developing reliability plan, 395
- prioritizing activities using error logs, 355
- scripting, 185
- testing of, 356

- using error logs, 353–354
- verifying requirements for, 355–356
- viewing error log output, 352–353
- managed code**
 - P/Invoke as interface between native and, 526–527
 - using data structures within, 538–539
 - using external functions within, 527–529
- management**
 - code snippets and, 167
 - evolutionary model and, 42
 - hostile, 23
 - incremental/iterative model and, 42–43
 - involving in design, 24–26
 - involving in specification process, 21
 - of monitoring process, 412
 - reliability and, 365–366
 - reports and, 508
 - security and, 367
 - speed and, 364–365
 - testing design, 27
 - testing reports, 510
 - using command line interface, 90
- man-in-the-middle attacks, 142**
- manual tests, 305**
- MapNetworkDrive () method, Windows PowerShell, 190**
- mapping network drive, Windows PowerShell, 188–191**
- [MarshalAs] attribute, 526–527**
 - creating basic dialog box, 528
- Marshal.GetLastWin32Error(), 530**
- marshaling techniques, P/Invoke, 520**
- MDI (Multiple Document Interface) windows, 84–85, 254**
- Mean Time Between Failures (MTBF), 128**
- measuring application speed, 109–115**
 - defining application counter, 115–119
 - selecting another view, 114–115
 - setting graph properties, 112–114
 - using counters, 110–112
 - viewing standard counters, 109
- media types**
 - choosing for deployment, 33–34, 342
 - verifying before sending out, 342
- medial caps, 243**
- memory**
 - loading Windows PowerShell script into, 190
 - verifying resource availability, 122–123
 - viewing content, 318
- menu-driven user interface, 91–92**
- menus, customizing and deleting, 162–163**
- message routing, for new controls, 226**
- Method Call Result dialog box, Object Test Bench, 237**
- methods**
 - adding to object model, 61
 - anonymous, 8
 - creating LINQ queries, 450, 452–453
 - custom derived controls, 218–220
 - new controls, 226
 - XML storage class, 422
- Microsoft**
 - Windows Form Designer Toolkit, 158–159
 - Windows Forms Control Library template, 216–217
 - Windows Media Player, 86–87
 - Windows PowerShell. *See* Windows PowerShell
 - Windows Server 2008 Server Core, 90
- Microsoft Console (MSC) files, 112**
- Microsoft Developer Network (MSDN), 554, 557–558**
- Microsoft Download, 555–556**
- Microsoft Help and Support, 556**
- Microsoft Installer (MSI), installing F#, 480–481**
- Microsoft Knowledge Base, 556**
- Microsoft Office Compatibility Pack, 517**
- Microsoft PowerCommands, 561–565**
- Microsoft Research Web site, 556–557**
- Microsoft resources and tools. *See* resources and tools, Microsoft**
- Microsoft Vista, User Account Control, 392**
- Microsoft.VisualStudio.DebuggerVisualizers, 203**

ML language, 11

mockups, creating and testing report, 510

monitor thread class

- creating file system monitor, 499–500
- creating multithreaded application, 495–499
- starting and stopping monitor process, 500–501
- testing multithreaded application, 502–504

monitoring, 145–149

- and Big Brother syndrome, 405–406
- defining strategy, 145–146
- implementing logging, 146–147
- managing security, 412
- for security, 133
- sending administrator alerts, 148–149
- using event log, 147–148
- using Watch window, 196–197

Mono, 480

MSC (Microsoft Console) files, 112

MSDN (Microsoft Developer Network), 554, 557–558

MSI (Microsoft Installer), installing F#, 480–481

MTBF (Mean Time Between Failures), 128

multiline comments, C#, 7

multi-platform scenarios, 11

Multiple Document Interface (MDI) windows, 84–85, 254

multiple languages, 4

multiple-source errors, 297

multiprocessing

- effects of, 109
- multithreading vs., 489–490
- tools for working with, 493

multithreaded applications, 487–506

- benefits of, 490–491
- creating. See multithreaded applications, creating
- current state of, 492–493
- deficiencies of, 491–492
- defining, 488–490
- high speed applications using, 107–108

multithreaded applications, creating, 493

- adding file processing thread class, 494–495
- adding monitor thread class, 495–499
- defining project, 494
- developing application code, 499–502
- implementing design, 505–506
- testing, 502–505

multithreading, defined, 487–490

N

naming conventions, 241–246

- abbreviations, 244–245
- acronyms, 244–245
- applying m_ for local memory, 244
- choosing class names from candidates, 58
- command line, 91
- custom toolbars, 161
- documentation of, 267
- prefixing interface with I, 244
- special cases, 245–246
- uppercase, 244–245
- using camel case, 242–243
- using Hungarian notation, 242
- using numbers, 245
- using Pascal case, 243

NASA (National Aeronautics and Space Administration), and waterfall model, 38

National Aeronautics and Space Administration (NASA), and waterfall model, 38

native code, 526–527

Nav command, 318

NDoc, 277

nested command line switch, 247

.NET assemblies, SQL Server support for, 184

.NET Framework

- application deployment and, 330
- choosing version when building controls, 217
- support for forms, 80

.NET Framework Solutions: In Search of the Lost Win32 API (Sybex, 2002), 538–540

networks

- download installations, 342
- verifying resource availability, 122

New Project dialog box

- building derived control, 216–217
- building new control, 226–227
- checking derived control functionality, 221
- creating F# application, 484–485
- defining new component project, 229

New Toolbar dialog box, 160–161**nodes, Visio UML deployment diagrams, 73****-NoExit command line switch, 186****non-deferred evaluation, LINQ, 456****None setting, FormBorderStyle, 81****nonprocedural languages, 478****nontechnical input, 212****Northwind database, 462–467****Notepad**

- command line switch mistake in, 246
- editing XML files using XML, 558–561
- example of SDI, 83

Notification Area, user interface design, 89**numbers, as naming convention, 245****O****object model maps, 59–61****object models, creating, 56–64**

- adding external classes, 61–62
- considering class trade-offs, 62–63
- defining class concepts, 56–60
- for design strategy, 55
- developing internal classes, 60–61
- improving class design, 63–64
- presenting with UML, 70–77

Object Test Bench, 212, 235–238**objects, using LINQ to Objects, 455–457****OCaml language, 11****Office Tools folder, Options dialog box, 158****Office User Fluent Interface, 68–69, 87–88****online references**

- [StructLayout] attribute, 535–536
- active COBOL applications, 35
- Adaptive Software Development, 52
- add-ins, 165, 168, 169
- agile programming, 46, 48
- C# expressions, 185

- C# Software Transactional Memory, 556–557
- CAB files, 341
- camel case mixed with Pascal case, 243
- CAPTCHA, 133
- Clippy, 97
- code snippets, 167
- color blindness, 102–103
- color coding blocks, object model, 62
- Component-Oriented Programming, 228
- control attributes, 220
- Crystal Clear, 50
- Crystal Reports, 514–515
- custom attributes and exceptions, 262
- Daltonization, 573
- DataSet visualizer Help, 201
- DOCX format, 517
- employee monitoring legalities, 145
- entity classes, 464
- Error Lookup utility, 531
- Expand utility, for CAB, 341
- Extreme Programming, 50
- F#, 479, 480
- Feature Driven Development, 52
- FishNet Security, 409
- functional languages, 11, 479
- FxCop download, 575
- Hungarian notation, 242
- Immediate window, 197
- Internet Archive Wayback Machine, 558
- IronPython, 10
- Job Access With Speech, 101
- lifecycle of individual car, tracking, 17
- LINQ evaluations, deferred and non-deferred, 456
- LINQ performance issues, 473
- LINQPad, 582
- man-in-the-middle attacks, 142
- messages, 260
- Microsoft Download, 555
- Microsoft Office Compatibility Pack, 517
- Microsoft PowerCommands, 561, 565
- Microsoft Research, 556
- Microsoft resources and tools, 557–558
- MSDN downloads, 554
- multi-platform scenarios, 11

online references (continued)

- multiprocessing, 109, 493
- Northwind database download, 462
- object model map, 59–60
- OOXML file format, 517
- Open Document, 516
- Open Web Application Security Project, 408
- PLINQ, 505
- required application speed, 106
- RibbonX, 87
- RoboCopy, 331
- Sandcastle, 277
- Scrum, 48–49
- Section 508 checklist, 103
- security testing, 150
- software reliability, 128
- SpeedGuide Security Scan Information, 408
- SQL Metal, 464
- Technet, 554–555
- test types, 305–306
- Text Template Transformation Toolkit, 158
- third-party testers, 409
- third-party visualizers, 202
- Type A personality, 106
- Unified Modeling Language, 56, 72
- UnmanagedType enumeration, 527
- virtualization technology information, 555
- Vischeck, 571
- Visual Studio 2008 support for SQL Server 2008, 181
- waterfall model derivatives, 38
- Web services for any need, 135
- Windows PowerShell, 90
- Windows PowerShell debugging, 191
- Windows PowerShell download, 186
- Windows PowerShell scripts, 187–188, 190
- WorldWide Telescope, 557
- XCopy deployment, 330
- XML Notepad, 420, 559
- XML Spy, 420
- OnPaint() event, controls, 227**
- OOXML (Open Office XML), 517**
- Open Command Prompt, PowerCommand, 564**

- Open Containing Folder, PowerCommand, 564**
- Open Document report format, 516**
- Open Web Application Security Project (OWASP), 408**
- operators, basic LINQ, 446–449**
- optional command line switches and data input, 247**
- optional exclusive command line switch, 247**
- optional parameters, C# lack of support for, 9**
- Options dialog box, configuring IDE, 156–159**
- OrderBy() method, LINQ to Objects, 456**
- orderBy operator, LINQ queries, 448–449**
- ordered tests, 305**
- organizational policies, including in reports, 512–513**
- Organizational setting, XCopy, 418, 425**
- OSDFE.OSD file, CAB files, 341**
- Output window, debugging with, 317**
- outputs, alternative report, 515–517**
- OutputTimer_Tick() method, adding counters, 118**
- overhead, multithreading and, 492**
- OWASP (Open Web Application Security Project), 408**
- ownership of data, 405–406**

P

- Pack parameter, 536**
- packing data structures, P/Invoke, 520**
- Parallel LINQ (PLINQ), 505**
- partial data hiding, 404**
- partial keyword, C#, 8**
- Pascal case, 243**
- passwords**
 - encrypting, 413
 - using hashes, 144
- Paste Class, PowerCommand, 564**
- Paste References, PowerCommand, 564**
- patches, deploying, 356, 408**
- pattern matching, 135**
- PC Probe II interface, 87**

- PDF (Portable Document Format), reports, 516**
- perceived speed, 107**
- performance**
 - enhancing speed, reliability and security, 32–33
 - monitoring with Add Counters dialog box, 110
 - problems with LINQ, 473
 - security affects on, 404
 - speed vs., 106–107
- Performance console, 111**
- Performance Counter Builder dialog box, 116**
- Performance Object, Add Counters dialog box, 110**
- Performance Tools folder, Options dialog box, 157**
- performance triangle, 361–370**
 - after initial development phase, 362–363
 - creating, 368–369
 - defining in development process, 363–364
 - reliability component, 365–366, 369–370
 - security component, 366–370
 - speed component, 364–365
- Permanent property, deployment setup, 337**
- personal revelations, lessons-learned journal, 397–398**
- personality, and required application speed, 106**
- pessimistic applications, 392**
- pigs, Scrum, 49**
- P/Invoke, 519–540**
 - creating basic data structures, 535–537
 - defining, 519–521
 - enumerating standard console handles, 535
 - exiting applications, 259–260
 - function calls, 537–538
 - using data structures within managed code, 538–540
- P/Invoke, calling external functions, 521–522**
 - adding required directives, 522
 - creating external function reference, 525–526
 - error handling, 529–534
 - with function data call, 526–527
 - implementing design, 540
 - within managed code, 527–529
 - structures, enumerations and constants, 522–525
- pipeline operator (|>), F#, 483**
- PLINQ (Parallel LINQ), 505**
- podcasts, 269**
- pointers**
 - code reliability concerns, 9
 - unsafe code blocks and, 6–7
- policies**
 - database access, 66
 - organizational report, 512–513, 516
 - resource and tool, 551
- portability, and P/Invoke, 519–520**
- Portable Document Format (PDF), reports, 516**
- porting issues, multithreading, 492**
- PostMessage() function, 260**
- power users, 269**
- PowerShell. See Windows PowerShell**
- POWERSHELL.EXE file, 186–187**
- preferences, generating code from UML, 75–77**
- prioritizing, maintenance activities, 355**
- Process Explorer, 554–555**
- process modeling, RAD model, 45**
- ProcessData() method, 140**
- processes, thread execution, 488**
- Product Owner, Scrum, 49**
- production system, test system duplicating, 393**
- productivity, third party vendor issues, 568**
- profile, updating security, 411**
- progress indicators, 107**
- Projects and Solutions folder, Options dialog box, 157**
- properties**
 - adding to object model, 61
 - common control, 214
 - control accessibility, 101–102
 - custom derived control, 218–220
 - deployment setup, 335–337
 - derived control, 221–222
 - dialog box, 81–82
 - graph, 112–114
 - new control, 226
 - test, 309–310
 - XML storage class, 424–425
- protected locations, placing data in, 144**

prototypes

- evolutionary model, 41–42
- RAD model, 43–45
- throwaway model, 40–41

providers, LINQ

- built-in, 454–455
- third-party, 472–475
- understanding, 442
- using LINQ to DataSet, 457–461
- using LINQ to Objects, 455–457
- using LINQ to SQL, 462–467
- using LINQ to XML, 467–472

PSI file extension, 187

Q

QA (quality assurance) checks, 319–321

- getting user involved, 321
- interacting with Team B, 319–320
- useful feedback, 320–321

queries, LINQ, 180, 449

- joining multiple data sources to create, 450–453
- simple, 449–450
- SQL queries vs., 445–446
- standard syntax of, 442
- using keywords, 446–449
- using let operator, 453–454
- using LINQPad to create, 582–585

queries, SQL Server, 178

R

RAD (rapid application development) model, 43–45, 52

range checks, 136–138

rapid application development (RAD) model, 43–45, 52

rapport, developing with users over reports, 513–514

RDF (Resource Description Framework), 180

ReadOnly property, deployment setup, 337

Ready to Install dialog box, FxCop, 575

Rearrange Commands dialog box, menus, 162–163

recording

- documentation, 267
- macros, 168

redundancy, and application strategy, 123

Registry

- modifying application deployment setup in, 338
- saving application settings in, 126

release date, recording in archive, 547

reliability, applications, 121–130. *See also*

reliability implications; error handling

- building data model for, 66
- creating application, 369–370
- design implementation and, 28
- designing interface for, 98, 130
- enhancing, 32–33
- expecting the unexpected, 128
- FxCop improving. *See* FxCop
- as performance triangle component, 363–366
- RibbonX benefits, 129–130
- saving data and, 124–125
- saving settings and, 125–127
- saving state and, 127
- security issues, 405
- third party vendor advantages, 568
- verifying availability of resources, 122–124

reliability implications, 391–402

- applications that do nothing, 392–394
- avoiding pessimistic applications, 392
- calculating code metrics, 400–401
- defining external threats, 394–395
- developing plan, 395
- placing users in monkey mode, 395–398
- running code analysis, 398–400

Reload Projects, Microsoft PowerCommand, 564

Remove and Sort Usings, Microsoft PowerCommand, 565

Rename Toolbar dialog box, 161

reporters, security and media, 367

reports, 507–508

- built-in SQL Server sprocs for, 183
- built-in SQL Server views for, 183–184
- creating automated application, 280
- creating other forms of output, 515–517
- elements of good, 508–510

implementing your design, 518
 self-modifying, 514–515
 setting view, 114
 testing, 510
 user requirements for, 511–514
 using XML data in, 421–422

Resource Description Framework (RDF), 180

resources

defining external, 14–15
 programming team, 12–14
 verifying availability of, 122–124

resources and tools, 543–552

creating archive, 545–546
 creating team toolbox, 550–551
 defining resource, 543
 defining useful tool, 549–550
 designing and building custom, 544–545
 developing rapport with vendors, 547–548
 ending time-wasting clutter, 544–545
 Microsoft. *See* resources and tools, Microsoft
 obtaining, 551–552
 rules of useful resources, 548–549
 third-party. *See* resources and tools, third-party
 updating, 547

resources and tools, Microsoft, 553–566

discovering online, 557–558
 finding at MSDN, 554
 Microsoft Download, 555–556
 Microsoft PowerCommands, 561–565
 Microsoft Research, 556–557
 obtaining, 565–566
 TechNet, 554–555
 XML Notepad 2007, 558–561

resources and tools, third-party, 567–585

FxCop. *See* FxCop
 LINQPad, 582–585
 locating, 567–571
 Vischeck, 571–574

respect, as XP core value, 52

responsiveness, and multithreading, 490–491

retirement strategy

application lifecycle, 35
 waterfall model, 40

ReturnValues, 528–529

reusable code, database for, 61

reviews, locating third party vendors through, 569–570

RibbonX interface

reliability of, 129–130
 task-driven interface of, 92
 for user interface design, 87–88

RoboCopy (Robust File Copy Utility), 331

role-driven user interface, 94–95

roles, database access, 66

rules

FxCop, 580–581
 information-driven interfaces, 93
 safe code, 398–399
 XCopy deployment, 331

/Run (/R)command line switch, 175

Run Selection button, 310

/Runexit command line switch, 175

S

SaaS (Software as a Service), 327–328

Sample Automatically Every setting, System Monitor, 113

Sandcastle, 277–279

Save() method, LINQ to XML, 470–471

saving data, in XML format, 425–428

scalability, of multithreaded applications, 490–491

scripted installations, 343

scripting applications, 177, 184–185

considering options, 177–180
 creating basic F#, 482
 developing Windows PowerShell solutions, 185–191
 interacting with SQL Server, 181–184
 using C# expressions, 185

Scrum, 48–49

ScrumMaster, 49

SDI (Single Document Interface)

windows, 82–85

search engines, using information-driven interface, 93

search times, serialized XML data entries, 435–436

Section 508 checklist, 103

security, 131–132, 403. See also error handling

- assuming worst-case scenario, 132–133
- avoiding pessimistic applications, 392
- breaking into your application, 407–411
- building data model, 66
- creating for application, 369–370
- design implementation, 28
- design strategy, 150–151
- designing interface, 98–99
- eliminating errant input, 133–142
- encrypting and decrypting data, 413–414
- enhancing, 32–33
- hiding data from view, 142–144
- identifying external threats, 394–395
- monitoring application, 145–149, 412
- multithreaded application deficiencies, 492
- for new controls, 226
- as performance triangle component, 363–364, 366–368
- role-driven interfaces and, 95
- trade-offs for, 404–406
- updating profile, 411
- using development team to overview, 149–150
- Windows PowerShell, 186, 190

Security TechCenter, TechNet, 555

Select() method, LINQ to Objects, 456

select operator, LINQ queries, 447–448

Select Run list, running tests, 309

selective data hiding, 404

self-documenting code, 265

self-healing applications, 298–300

self-modifying reports, 514–515

sequence diagrams, Visio UML, 73

sequential software development model, 38–40

serialized XML, 415–418

- adding new data, 432–433
- creating design, 439–440
- creating XCopy application, 418–419

creating XML storage class, 422–425

deleting data entries, 436–437

deployment setup, 334

displaying data entries on screen, 439

editing existing entries, 433–436

loading data, 428–431

logging using, 146

saving application settings in, 126

saving data in XML format, 425–428

sorting data entries, 437–439

using XML data in reports, 421–422

viewing sample data, 419–421

for XCopy installation, 33

Server Core, 90

Server Explorer, 181–182

servers, verifying resource availability, 122

Service Level Agreements (SLAs), 122

SetConsoleCursorPosition() function, 537–538, 540

setLastError=true parameter, 526, 530

setup programs, wizard-driven vs. command-driven, 94

Setup project. See deploying application, basic setup

Setup Project template, 333–334

Setup Wizard, 339–340

Shapes window, Visio UML, 72

Shell command, 318

shortcuts, 161–162

basic deployment setup, 335–336

loading projects using, 170

Show All Files, Microsoft PowerCommand, 565

ShowDialog() method, exiting application, 254

shrink-wrap deployment, 327, 344

simple queries, LINQ, 449–450

simplicity, as XP core value, 51

Single Document Interface (SDI) windows, 82–85

single-source errors, 297

Sizeable setting, FormBorderStyle, 82

skills, team, 13–14, 106

skinned applications, 86–87

skins, Device Tools folder, 158

SLAs (Service Level Agreements), 122

SMALL_RECT structure, 536–537

smart testing design, 309

snippets

- creating and using code, 166–167
- defined, 155, 165
- removing when no longer used, 166

software

- archiving, 546
- creating archive, 546

Software as a Service (SaaS), 327–328

Solution Explorer, building CAB project in, 340

sorting, serialized XML data, 437–439

Source Control folder, Options dialog box, 157

special needs. See accessibility

specification

- defining, 21–22
- presenting application concept using, 23
- using agile programming when project lacks, 47
- waterfall model and, 38–39

speculate phase, ASD programming, 52

speed, application, 105

- agile programming technique and, 46
- creating application counters, 115–119
- defining performance triangle, 363–365
- design strategy, 28, 119–120
- enhancing, 32–33
- error handling reducing, 285
- high speed, 107–109
- measuring application, 109–115
- multiprocessing and, 489, 493
- multithreading and, 489
- perceived vs. real, 107
- performance vs., 106–107
- RAD model and, 43–45
- security checks reducing, 405
- throwaway model and, 40–41

speed keys, as visual element, 100

SpeedGuide Security Scan Information, 408

spiral model, 36–37

splitters, using SDIs, 83

spoofing, 142

sprocs. See stored procedures (sprocs)

SQL (Structured Query Language), LINQ vs., 442–446

SQL Server

- accessing with stored procedures, 462
- scripting with, 178
- using LINQ to SQL, 462–467
- using .NET assemblies, 184
- using sprocs, 183
- using views, 183–184
- working from Visual Studio, 181–182

SqlCommand object, SQL Server, 178

SQLMetal

- generating classes for databases, 464–466
- using LINQ to SQL, 444–445, 462

stakeholders

- defined, 21
- in design process, 23
- in design testing, 26–27
- in specification process, 21
- XP using interaction of, 50

standard // comments, 270–271

standard /*... /* comments, 271

standard exception classes, 286–294

standards, and custom features, 261

Stardock games, 327

statechart diagrams, Visio UML, 74

static classes, 8

static connections, 66

static extern, P/Invoke calls as, 526

static structure diagrams, Visio UML, 74

STM (Software Transactional Memory), C#, 556–557

storage

- for application settings, 125–127
- discouraging for generalized data, 143
- using XML for. See serialized XML

storage class, XML, 422

- creating, 422–425
- using XML data in reports, 421–422

stored procedures (sprocs)

- accessing SQL Server using, 462
- using built-in SQL Server, 183
- working with, 178

StreamReader, 429

strong typing, Visual Basic, 9

[StructLayout] attribute, 535–537

structures, classes vs., 59

style requirements, coding, 26, 28

support

C#, 7–8

third party vendors and, 568–569

support logs, 354–356

switches, command-line

adding, 247–252

CMD.EXE, 170–174

POWERSHELL.EXE, 186–187

RoboCopy, 331

testing, 252–253

when to use, 246–247

SysInternals home page, 555

System Monitor

defined, 109

setting graph properties, 112–114

using counters, 110–112

System property, deployment setup, 337

system resources, tools for, 549–550

System tab, 96

System.Attribute class, 262

System.Environment class, 255–258

System.Exception class, 262

SystemParametersInfo() function, 430–431

T

T4 (Text Template Transformation Toolkit),

Options dialog box, 158

tabbed interface dialog box, 85

TabIndex property, visual elements, 100

Table and Database Designers subfolder,

Database Tools, 157

Table field, DataSet visualizer, 201

Tag property, controls, 215

task list comments, 272

task load, required application speed, 106

task-driven user interface, 91–92

tasks

database access and, 66

using macros to automate, 168

viewing current, 194–195

TechNet, 554–555

technical writers, on development team, 369

templates, test, 305–306

TemporaryMacro, 168

Test List Editor, 311–312

Test Results window, 308–311

Test Runs window, 314–315

test tables, LINQ to DataSet queries, 458–460

Test Tools folder, Options dialog box, 158

Test View, 309–310

Test.FS file, 482

testing, 303

breaking into applications. See breaking into applications

command line functionality, 252–253

complexity of multithreaded applications, 492

controls, 135

creating lessons-learned journal, 397–398

creating new test, 304–308

debugging and, 29–31

deployment, 34

design, 26–28

human view of, 315–316

incremental/iterative models, 43

maintenance and, 356

multiprocessed applications, 490

multithreaded applications, 502–505

new components, 233–234

new counters, 119

new custom visualizers, 207–209

organizing using Test List Editor, 311–312

products of third party vendors, 570–571

RAD, 45

releases, 324–325, 344

reliability, 395–397

reports, 510

retesting when code changes, 304

running, 308–309

- security, 149–150
- setting properties using Test View, 309–311
- smart design for, 309
- speed, 106
- understanding Code Coverage Results window, 312–314
- user interface, 25
- user requirements, 31–32
- UserControl, 222–225
- using third-parties for, 409
- viewing test runs, 314–315
- visualizers, 207–209
- Text Editor folder, Options dialog box, 157**
- text fields**
 - checking all user input, 135
 - techniques for avoiding, 135
- text format, manual tests, 305**
- Text Template Transformation Toolkit (T4), Options dialog box, 158**
- Text Templating folder, Options dialog box, 158**
- text visualizer, 198–199**
- third-party products. See also resources and tools, third-party**
 - adding external classes from, 61
 - building derived controls from, 216
 - circumventing RibbonX interface with, 88
 - for deployment, 323–324
 - issuing patches, 408–409
 - LINQ, 472–475
 - obtaining add-ins as, 155, 165, 168–169
 - obtaining visualizers as, 193, 202
 - testing for common exploits using, 408
 - for using XML files, 277–279
- this object, 195–196**
- this.Handle property, dialog boxes, 528**
- threads. See multithreaded applications**
- throwaway model, 40–42**
- time frame, for database access, 66**
- timers, adding counter to application, 118**
- ToArray() method, LINQ to Objects, 456**
- tokens, task list, 272–273**
- Toolbar, System Monitor, 113**
- toolbars**
 - adding to IDE, 159–162
 - user interface, 91–92
- Toolbars tab options, 161**
- Toolbox**
 - adding derived control to, 222–223
 - checking derived control functionality, 221
 - creating team, 543, 550–551
- tools. See also resources and tools**
 - adding external, to Tools menu, 164–165
 - creating archive, 545–546
 - creating inventory of, 12
 - defining useful, 549–550
 - developing rapport with vendors, 547–548
 - ending time-wasting clutter, 544–545
 - updating, 547
- Tools menu, 164–165**
- Tools.Shell command, 197**
- tooltips**
 - adding accessibility descriptions, 101–102
 - not requiring user interaction, 82
 - nudging users with, 97
 - Toolbar tab options, 161
 - turning on or off, 97
- trade-offs of security, 404–406**
- Transform Templates, Microsoft PowerCommand, 565**
- transitions, Visio UML, 74**
- Transitive property, deployment setup, 335**
- Trap statement, Windows PowerShell, 190**
- triple comment symbol (///), C#, 7**
- Triple DES, cracking, 144**
- troubleshooting, using logging, 146**
- TRX extension, 309**
- try...catch blocks**
 - error handling with, 284
 - trapping multiple-source errors using, 297
 - Windows PowerShell not using, 190
 - working with single-source errors using, 297
- tutorials, documentation for user, 269**
- TXT file extension, 187**
- Type A personality, 106**

U

UAC (User Account Control), 186, 392

uiAction, 431

uint type enumeration, 524

uiParam, 431

UML (Unified Modeling Language), 70–77

defined, 55

generating code from UML, 75–77

getting started with Visio, 71

trial version download, 56

tutorials on, 72

Visio functionality, 72–75

uncontrolled installations, 343–344

underscore (_), private variables, 244

Undo Close, PowerCommand, 565

Unified Modeling Language. See UML (Unified Modeling Language)

Unit Test Wizard, 306–308

unit tests, 306

Universal Product Code (UPC) values, 135

Unload Projects, PowerCommand, 565

UnmanagedType enumeration, 527

unsafe code blocks, in C#, 6–7

UPC (Universal Product Code) values, 135

Update Data, Microsoft Console files, 112

UpdateCallback delegate, 502

updates

issuing security, 408

providing required, 132–133

of resources and tools, 547

security profile, 411

UpdateStatus() event handler, 502

uppercase, naming conventions, 244–245

usage requirements, resources and tools, 551

use case diagrams, Visio UML, 74

User Account Control (UAC), 186, 392

user interface design, 79–80

accessibility requirements, 99–103

creating, 24

creating dialog utility, 80–82

defining user requirements, 68–70

developing specialty skinned and free-form applications, 86–87

developing strategy for, 95–99, 103

interacting with MDI, 84–85

interacting with SDI, 82–83

testing, 27

testing user requirements, 31–32

types of, 91–95

using command line interface, 90–91

using Notification Area, 89

using tabbed interface, 85

working in visual environment, 87

working with RibbonX, 87–88

User Interface view, deployment setup, 338

User setting, XCopy, 419, 426

user threads, multithreaded applications, 489

User32.DLL file, 526

UserControl, 215–216, 223–225

users

application reliability and, 366, 395–397

application security and, 367–368, 404–405

application speed and, 364–365

creating design for, 24–26

creating reports for, 508, 510–512

creating user classes, 70

defining requirements of, 67–70

design implementation, 27

developing interaction strategy, 95–99

documentation for, 268–269

involving in QA process, 321

issues surrounding ownership of data, 405–406

obtaining bugs from, 319

performing deployment, 343

response forms for, 85

security monitoring of, 412

testing using, 31–32, 325

User's Program Menu shortcut, 335

using directives, P/Invoke, 522

V

validation, XML data, 425

Value Bar, System Monitor, 113

values

range checking, 136

working with return, 528–529

variable data switches, 247

VBA (Visual Basic for Applications), 178

vendors

- developing rapport with, 547–548
- recording in archives, 546
- sending updates for resources and tools, 547

verbalization, of application concept, 18

verification

- of data sources, 142
- of maintenance requirements, 355–356

VeriSign, 409

view

- built-in SQL Server, 183–184
- checking colors for colorblind, 571–574
- documentation as, 266–267
- hiding data from, 142–144
- of IDE. *See* IDE (Integrated Development Environment), viewing data in
- modifying application deployment setup, 338–339
- of standard counters, 109–115
- of test run, 314–315
- of XML data, 419–421
- visualizers. *See* visualizers

virtualization technology information, 555

viruses, checking deployment media for, 342

Vischeck, 571–574

Visio

- generating code from UML, 75–77
- getting started with, 71
- UML functionality, 72–74

vision problems. *See* accessibility

Visual Basic for Applications (VBA), 178

Visual Basic.NET

- C# language vs., 5
- information hiding in, 9
- using with C#, 10

visual environment, user interface design, 87

Visual MainWin, 11

Visual Studio

- interacting with SQL Server, 181–182
- obtaining F# support for, 479–481

visualizers, 193–194, 197–198

- creating, 202–209
- DataSet, 200–201

HTML, 200–201

need for, 198

testing, 207–209

text, 198–199

third-party, 198–199

XML, 199–200

VSMDI file, 307

W

Watch window, 194

- debugging with, 317
- viewing data in IDE, 196–197

Watcher object, 497–501, 503–504

waterfall model, 38–40, 46, 48

Web services, 61–62, 135

Web sites, showing with Vischeck, 571–574

Web tests, 306

Webcasts, 269

where operator, LINQ queries, 447–448

widowed software, 568

Win32 API. *See* P/Invoke

window handle, new controls, 226

windows, debugging, 316–317

windows, viewing data in IDE, 193–197. *See*

***also* visualizers**

Autos window, 194–197

Command Window, 196–197

defining for custom visualizers, 204–206

Locals window, 195–196

Watch window, 196–197

Windows Form Designer Toolkit, 158–159

Windows Forms Control Library template, 216–217

Windows Media Player, 86–87

Windows PowerShell, 185–191

command line interface, 90

creating scripts, 187–190

defined, 90

executing script, 190–191

obtaining debugger support, 191

PS utility, 186–187

scripting options, 179–180

Windows Server 2008 Server Core, 90

Windows volume control, Notification Area, 89

WinZIP, 341

wizards

- considering user requirements, 68
- creating new test, 306–308
- nudging users with, 97
- relying on dialog boxes, 93–94
- using tabbed interface, 85

WM_QUIT message, 260

Word format, manual tests, 305

work area, windowed applications, 83–84

Workflow Designer folder, 159

workflows

- creating in user interface, 68–69
- RibbonX for, 88, 129–130

WorldWide Telescope (WWT), 557

worst-case scenario, security assuming, 132–133

Wrap option, text visualizer, 198–199

writing application, 28–29

WWT (WorldWide Telescope), 557

X

XCOPY application

- application deployment using, 33, 330–331
- using XML as storage methodology, 418–419

XML (eXtensible Markup Language)

- editing files using XML Notepad 2007, 558–561
- IntelliSense support and, 7
- MSC files containing, 112
- serializing. See serialized XML
- serializing for XCOPY installation, 33

setting application to create, 275

using, 276–279

using LINQ to XML, 467–472

XML /// comments, 271–272, 274–275

XML editor, 550

XML mapping files

LINQ to SQL, 464–467

LINQ to XML, 445

XML Notepad, 420, 558–561

XML Schema Definition (XSD), 420, 425

XML Serialization Assemblies, 334

XML Spy, 420

XML visualizer, 199–200

XMLDocument object, 417

XmlMappingSource, 444–445

XmlSerializer, 429

XP (Extreme Programming), 50–52

XSD (XML Schema Definition), 420, 425

XSLT (eXtensible Stylesheet Language Transformation)

displaying XML files using, 276

editing XML files using XML Notepad 2007, 560

functional language as basis of, 478

Y

yield keyword, C#, 7

Z

ZIP files

CAB files vs., 340

using DOCX option, 517