

Index

SYMBOLS AND NUMERICS

& (ampersand), as address operator for pointers, 207

*** (asterisk)**

as indirection operator for pointers, 207
for multi-line comments (`/*` and `*/`), 31, 67–68

@ (at sign) indicating identifiers, 74

\ (backslash)

preceding escape sequences, 44, 238
for searching for escape meta-characters, 240

: (colon)

for disabling `Group` object with regular expressions
(`?:`), 243
as namespace alias qualifier (`::`), 60–61
preceding format specifiers, 230
in ternary operator (`?:`), 136–137

, (comma) in derived class or struct declarations, 113

{ } (curly braces)

with `if` statements, 47, 48
joining statements in blocks, 31

" (double quotes), string literals enclosed by, 46

= (equals sign)

as assignment operator, 34, 134–135
as comparison operator (`==`), 48, 134–135, 147

- (minus sign)

as decrement operator (`--`), 135–136
for removing method calls from multicast delegates,
183

() (parentheses) for groups, 241–242

+ (plus sign)

in for loop iterator, 52
as increment operator (`++`), 135–136
regular expression groups and, 242

(pound symbol) for preprocessor directives, 70–73

? (question mark)

for disabling `Group` object with regular expressions
(`?:`), 243
in null coalescing operator (`??`), 139–140
in ternary operator (`?:`), 136–137

; (semicolon)

ending C# statements, 31
preprocessor directives not ended by, 71

' (single quotes), char literals enclosed by, 44, 46

/ (slash)

for multi-line comments (`/*` and `*/`), 31, 67–68
for single-line comments (`//`), 31, 67

[] (square brackets) for arrays, 57

32-bit processors, pointers and, 207–208

A

`Abort()` **method**

`MessageQueueTransaction` object, 1121–1122
`Thread` class, 354

aborting threads, 354

abstract modifier, 117, 125

`AcceptSocket()` **method** (`QuoteServer` class),
1278

access control lists. See ACLs

access modifiers, 94, 97

`Account` **class, 297–299**

`Accumulate()` **method**

`Account` class, 298–299

`Action` generic delegate, 299–300

described, 644

implementing for user-defined aggregates, 644–645

`AccumulateIf()` **method, 300–301**

AccumulateSimple() method (Account class)

`AccumulateSimple()` **method** (**Account class**), 298

acknowledgement message queues, 1120

ACLs (access control lists)

adding and removing from a file, 1225–1226

namespace for, 1222

reading from a directory, 1224–1225

reading from a file, 1222–1223

Action generic delegate, 299–301

ActionCancelEventArgs class

BusEntity class with, 190–191

Cancel property, 188

code listing, 189

derivation of, 188

Message property, 188

OnAction method, 189–190

using, 190–192

ActionEventHandler delegate, 188

Activator class

CreateInstance() method, 1027, 1028, 1029

GetObject() method, 1027, 1028, 1029, 1030

overview, 1027

activator for .NET Remoting client, 1012

Active Directory

accessing native ADSI objects, 735–736

accessing properties directly by name, 731

Active Directory Domains and Trusts MMC snap-in, 721

Active Directory Sites and Services MMC snap-in, 721

Active Directory Users and Computers MMC snap-in,
721–722

ADSI (Active Directory Service Interfaces), 723–724

ADSI Edit tool, 721, 722–723

architecture, 714–720

authentication, 729

binding, 725–729

cache, 732–733

configuration, 716

creating new objects, 733

data characteristics, 719

defined, 713

domain, 716

domain controllers, 716

domain tree, 717

DSML with, 724, 745–748

features, 714–715

finding message queues, 1103–1104

forest, 717

GC (global catalog), 717–718

getting directory entries, 730–731

hierarchical data organization, 714, 719

multimaster replication, 714, 718

object collections, 731–732

objects, defined, 715

read-mostly access, 719, 736

replication latency, 718

replication notification, 718

replication topology, 714

scheduled replication, 718

schemas, 715, 719–720

searching in, 736–745

security, 714–715

sites, 717

strong typing for objects, 715, 719

System.DirectoryServices namespace, 723, 725

updating directory entries, 734

UserSearch application, 740–745

uses for, 713

Active Directory Service Interfaces. See ADSI

Active Server Pages (ASP)

ASP.NET and, 22, 903

defined, 21

limitations of, 21–22

ActiveX controls in Windows Forms

ActiveX Control Importer for, 1168

ActiveX controls, defined, 1168

creating a Windows Forms application, 1169–1171

ActiveX Data Objects (ADO) versus ADO.NET, 580

Add() method

ArrayList class, 251, 272

Hashtable class, 261

List<T> generic class, 279

SortedList class, 258

AddCulturesToTree() method, 520–522

AddDocument() method

DocumentManager class, 284, 294

PriorityDocumentManager class, 290

AddRange() method (ArrayList class), 252

AddResource() method (ResourceWriter class), 527

address operator for pointers, 207

AddToMessage() method, 323

AddToOutput() method, 319

ADO (ActiveX Data Objects) versus ADO.NET, 580

ADO.NET. See also data binding; DataSet class; event-booking application; viewing .NET data

ADO (ActiveX Data Objects) versus, 580

alias names to database factory objects, 585

calling stored procedures, 594–597

closing database connections, 583, 585–588

commands, defined, 589

CommandType enumeration, 590

- constructing commands, 590
- converting relational data to XML, 694–696
- converting single table data to XML, 689–694
- converting XML to ADO.NET data, 696–699
- copying a `DataSet`, 625–626
- data access provided for C# by, 25
- data reader, 597–600
- database authentication, 583, 584
- database-specific classes, 581–582
- defining a database connection string, 584
- deleting database records, 591, 595–596
- ensuring resources are released, 585–588
- `ExecuteNonQuery()` method, 591
- `ExecuteReader()` method, 591–592
- `ExecuteScalar()` method, 592–593
- `ExecuteXmlReader()` method, 593–594
- executing commands, 590–594
- getting a single result from a SQL statement, 592–593
- inserting database records, 591, 596–597
- isolation level for transactions, 588–589
- for key generation with SQL Server, 627–629
- managing data and relationships with `DataSet` class, 600–612
- managing database connection strings, 584–585
- namespaces for, 580
- naming conventions, 629–630
- .NET versions and, 582
- OLE DB provider with, 597–600
- opening database connections, 583
- overview, 580–582
- persisting `DataSet` changes, 620–625, 691–692
- populating a `DataSet`, 619–620
- reading database records, 591–592, 593–594, 597–600
- reading DiffGram XML documents, 699–701
- shared classes, 581
- for tiered development, 625–627
- transactions, 588–589
- `try...catch...finally` blocks for releasing resources, 586, 587–588
- updating database records, 591, 594–595
- using database connections, 582–589
- using statement for releasing resources, 586–587, 588
- Web Forms and, 926–939
- XML schemas, 612–618
- ADSI (Active Directory Service Interfaces)**
 - accessing native objects, 735–736
 - ADSI Providers, 724
 - cache, 732–733
 - defined, 724
 - namespace for, 723
- ADSI Edit tool (Active Directory), 721, 722–723**
- aggregates**
 - defined, 643
 - user-defined, 644–645
 - using `AVG` aggregate, 643
- algorithms**
 - bubble-sorting, 180
 - for dictionaries, 260, 262–263
 - sorting, globalization and, 524–525
- aliases**
 - to factory objects for databases, 585
 - for namespaces, 60–61
- ampersand (&), as address operator for pointers, 207**
- `AnalyzeType()` **method, 318–319**
- anonymous methods**
 - `BubbleSorter` example, 180–182
 - defined, 176
 - defining delegates with, 176
 - `delegateTest` example, 176–177
 - event handlers, 187–188
 - multicast delegates with, 183–185
 - rules for, 177
 - `SimpleDelegate` example, 177–180
 - threading and, 353
- `AppDomain` **class, 414–416**
- `Append()` **method (StringBuilder class), 227**
- `AppendFormat()` **method (StringBuilder class), 231**
- `Application` **class**
 - `EnableVisualStyles` method, 785–786
 - methods and properties, 753
 - `Run()` method overloads, 753
- application configuration files for assemblies, 450, 453–456**
- application domains. See also .NET Remoting**
 - `AppDomain` class for, 414–416
 - application domain-bound objects, 1031
 - assemblies and, 413–416
 - contexts, 1015
 - processes and, 14–16, 413
 - remote objects confined to, 1017
 - for SQL Server 2005 databases, 634
 - strong data typing in IL and, 14–16
 - usefulness of, 14
 - using, 414–416
- `ApplyResources()` **method (ResourceManager class), 537**
- arguments. See parameters**

arithmetic operators

arithmetic operators

- overloading, 148, 151–152, 153–155
- pointers and, 210–211
- table listing, 134

array lists. **See also** `ArrayList` class

- adding elements, 251, 252
- arrays versus, 250
- boxing and unboxing and, 272
- converting to arrays, 253
- copying range of items to new list, 253
- dynamic growth of, 250–251
- elements as object references, 251
- indexers, 251
- performance compared to dictionaries, 259
- removing elements, 251–253
- setting capacity of, 250, 251

`ArrayList` class. **See also** array lists

- `Add()` method, 251, 272
- `AddRange()` method, 252
- dynamic growth of, 250–251
- `GetRange()` method, 253
- instantiating, 250
- `RemoveAt()` method, 251
- `RemoveRange()` method, 252–253
- setting capacity of, 250, 251
- `StringBuilder` class compared to, 250

arrays

- accessing elements, 58
- advantages and disadvantages of, 245
- array lists versus, 250
- `ArraySegment<T>` generic struct and, 303–304
- as collections, 247–248
- converting array lists to, 253
- `DataGridView` control for displaying data, 804–805
- finding number of elements in, 58
- `foreach` statements for iterating through items, 54
- initializing with specific dimensions, 58
- overview, 57–58
- pointers to, not allowed, 208
- as reference types, 58
- square brackets used for, 57
- stack-based, using pointers, 219–222
- `StreamReader` class with, 1215–1216
- `System.Array` as parent class for, 245

`ArraySegment<T>` generic struct, 303–304

as operator, 138

ASP (Active Server Pages)

- ASP.NET and, 22, 903
- defined, 21
- limitations of, 21–22

ASP.NET. **See also** `PCSDemoSite` application; Web

Forms

- ADO.NET and data binding with, 926–939
- application configuration, 939–941
- ASP and, 22, 903
- client-side code, 905–906
- code model, 909
- code-behind feature, 22
- COM objects in, 1171
- control palette, 913–914
- creating applications using C#, 21–23
- creating files in Visual Studio, 906
- Crystal Reports Web server controls, 914
- custom controls, 944, 952–956
- data Web server controls, 917–918
- DHTML versus, 22
- features, 22
- file extensions for code, 909
- forms authentication, 964–969
- further information, 910
- HTML server controls, 910
- IIS and, 904
- localization using, 544–545
- login system implementation, 969–970
- login Web server controls, 970–971
- master pages, 956–960
- navigation Web server controls, 960–963
- .NET Remoting servers for, 1050–1051
- overview, 904–905
- pages as classes, 22
- `PCWebApp1` example, 906–908, 909
- performance improvements over ASP, 22
- restricting directory access, 971–973
- `runat="server"` attribute, 905, 910
- `<script>` elements, 905
- security, 963–973
- server control example, 920–925
- similar technologies, 903
- standard Web server controls, 914–917
- state management in, 905
- themes, 974–979
- user controls, 944–952
- as user interface medium, 25
- validation Web server controls, 918–920
- viewstate field, 905
- WCF and, 1146
- Web Forms, 23
- Web server controls, 23, 910–925
- Web services issues, 1126
- WebParts Web server controls, 920
- XML Web services, 23

assemblies. See also configuring assemblies

adding resources to, 528–530
 as answer to DLL Hell, 412–413
 application domains and, 413–416
 attributes, 423–425
 building, 423–425
 calculating permissions required for, 481, 483–484
 code access security, 464–478
 creating using Visual Studio, 423–425
 defined, 17, 411, 412
 executable versus library code and, 17
 features, 413
 global assembly cache, 19, 413, 438–441
 ildasm symbols, 420–421
 installing, 412
 manifests, 418
 marking as CLS-compliant, 437
 metadata in, 18
 modules versus, 421–422
 namespaces and, 419
 obtaining details of types in, 319–320
 overview, 411–416
 primary interop assemblies for COM, 1166
 private, 18, 412–413, 419
 private versus shared, 412–413
 probing to find, 451
 reflection and, 19
 satellite assemblies for languages, 538
 shared, 19, 413, 419, 441–449
 structure of, 417–418
 using CLS for language independence, 427–436
 versioning, 412–413, 451–460
 viewing assemblies, 420–421
 viewing code access permissions for, 473–476
 viewing code groups for, 469–471
 Visual Basic .NET not case-sensitive, 77

Assembly Cache Viewer, 439–440**Assembly class, 319–320, 452****AssemblyInfo.cs project, 423****asserting permissions, 486–487****assignment operator**

assigning values to variables, 34
 comparison versus assignment, 48, 134–135
 shortcut operators, 135–136

asterisk (*)

as indirection operator for pointers, 207
 for multi-line comments (/* and */), 31, 67–68

asynchronous .NET Remoting, 1053–1054**at sign (@) indicating identifiers, 74****ATL Project Wizard (Visual Studio), 1159–1160****atomicity of Enterprise Services transactions, 1078****attributes. See also reflection**

of assemblies, 423–425
 assembly, defined with Enterprise Services applications, 1071
 COM and, 17
 COM interop attributes, 1175–1178
 of contexts (.NET Remoting), 1015–1016
 custom, 306–313
 for custom controls (Windows Forms), 787–788
 of Enterprise Services transactions, 1079
 .NET support for, 17
 for user controls (ASP.NET), 947
 for user controls (Windows Forms), 798–799

AttributeUsage attribute, 307–309**authentication**

in Active Directory binding path, 729
 AuthenticateUser() method, 1002
 database authentication with ADO.NET, 583, 584
 Login() method for obtaining token, 1002–1003, 1005
 for shared assemblies, 442
 for Web Forms, 964–969

automatic transactions, 1068, 1078**AVG aggregate, 643****B****backslash (\)**

preceding escape sequences, 44, 238
 for searching for escape meta-characters, 240

base classes

areas covered by, 20
 casts between derived classes and, 163–164
 default, 113
 defined, 19
 for generics, 277–278
 overview, 19–20
 for serialization, 1187
 for streams, 1206

BasicWebClient example

using WebClient class, 1240–1241
 using WebRequest and WebResponse classes, 1242–1243

Beginning XML (Wiley Publishing, Inc.), 659

BeginReceive() method (MessageQueue class), 1110

BeginTransaction() methods, 588

binary code reuse by generics, 273–274

BinaryFileReader application, 1210–1213

BinaryReader class

BinaryReader **class**, 1207

BinaryWriter **class**, 1207

binding (Active Directory)

- authentication in binding path, 729
- defined, 725
- with DirectoryEntry class, 729
- distinguished name (DN) in binding path, 727–728, 729
- downlevel logon and, 729
- example path for, 725
- items specified with, 725–726
- port number in binding path, 726
- protocol in binding path, 726
- server name in binding path, 726
- user name in binding path, 728–729
- user principal name (UPN), 729

Binding **class**, 819–820

binding methods (COM), 1155

binding .NET data. *See* data binding

binding (WCF), 1141–1143

BindingContext **class**, 818–819

BitBlt (bitmap block transfer), 873–874

Blacklist() **method** (CustomerTable **class**), 837

blittable data types, 638

blocks of code

- curly braces for, 31
- fixed blocks, 213–214
- marking unsafe, 206
- showing and hiding in Visual Studio 2005, 388–390

BookOfTheDayForm.cs, 534–539

books.xml file

- code listing, 662
- MSXML example, 662–663
- retrieving attribute data, 668
- transforming XML into HTML, 684–686
- validating XSD schema with XmlReader, 669–670
- XSD schema generated from, 668–669

bool type

- casting not permitted for, 144
- as if statement return type, 48–49
- overview, 43

Boolean properties of Type **class**, 315

boxing and unboxing

- array lists and, 272
- avoided by List<T> generics class, 272–273
- casts, 164–165
- comparing value types for equality, 147
- overview, 145–146
- performance impacts, 272

break **statement**, 49, 55

breakpoints, setting in Visual Studio 2005, 404

Brush **class**, 866–867

btnTwo_Click **method**, 187

BubbleSorter **class**, 180–182

bubble-sorting algorithm, 180

building. *See also* compiling

- assemblies, 421–425
- debug and release builds, 399–401
- defined, 399
- dictionaries, requirements for, 260
- projects in Visual Studio 2005, 399–403

BusEntity **class**, 190–191

business object tier, 25

Button **control** (Windows Forms)

- in application example, 756
- for IE-like toolbar, 1249–1253
- images on buttons, 763
- overview, 762–763
- PerformClick **method**, 763

ButtonBase **class** (Windows Forms)

- Button control derived from, 762–764
- CheckBox control derived from, 764
- RadioButton control derived from, 764

Button_Click **method**, 186–187

byte **type**, 41, 42

C

C++. *See also* Visual C++ 2005; Visual C++ 6; Visual C++ .NET

- advantages over C#, 6–7
- class for CLS example, 427–432
- CLR memory type safety tests and, 7
- constants in C# versus, 38
- destructors, 200
- memory management, 13
- templates compared to C# generics, 272

CalculateDocumentSize() **method**, 886–887

CalculateLineWidths() **method**, 886

calculating permissions required for an assembly, 481, 483–484

calendar_DayRender() **event handler**, 930–931

calendar_SelectionChanged() **event handler**, 930

call contexts (.NET Remoting), 1062–1064

callback functions, 171. *See also* delegates

camel casing, 76–77

CapsEditor **example**

- appearance, 880–881
- CalculateDocumentSize() **method**, 886–887
- CalculateLineWidths() **method**, 886

coordinate transforms, 889–890
 CreateFonts() helper method, 883–884
 extending for print functionality, 893–899
 File menu, 880
 Form1 class fields, 882
 functionality, 880
 initialization, 883
 Invalidate() method, 884–886
 LineIndexToPageCoordinates() method, 890
 LineIndexToWorldCoordinates() method, 889
 LoadFile() method, 884
 main menu, 883–884
 OnPaint() method, 887–888
 PageCoordinatesToLineIndex() method, 890, 893
 responding to user input, 890–893
 TextLineInformation class and, 882–883
 using statements, 881–882
 WorldCoordinatesToLineIndex() method, 889
 WorldYCoordinateToLineIndex() method, 889

case of names

camel casing, 76–77
 IL case sensitivity, 12
 Pascal casing, 76, 77
 Visual Basic .NET not case-sensitive, 77

caspol.exe (Code Access Security Policy tool)

applying full trust, 492, 502
 changing code group permissions, 494–495
 creating and applying permissions sets, 495–497
 creating code group using strong name, 498
 creating custom code groups, 493–494
 deleting code groups, 494
 Execution Checking setting, 468
 getting code group labels, 491
 labels for code groups, 469
 listing code groups and descriptions, 466–467
 listing code groups compactly, 468
 listing options for, 466
 Policy change prompt option, 469
 resetting security policy, 493
 specifying trust level for code groups, 468
 turning security on and off, 493
 verifying the trust level, 503
 viewing code access permissions for assemblies, 474–476
 viewing code groups at enterprise level, 477–478
 viewing code groups at user level, 476–477
 viewing code groups for assemblies, 469–471, 503–504
 viewing permission sets, 497

casting. See also user-defined casts

array list elements, 251
 compilation and, 165–166
 converting array element to struct member variable, 144
 converting nullable types, 144
 converting primitive types, 143–144
 dangers of, 143
 data loss from, 143–144, 166–167
 limitations of, 144
 between numeric and string types, 144–145
 between pointer types, 208–209
 pointers to integer types, 208–209
 QueryInterface() method and, with COM, 1154–1155

catch blocks

defined, 330
 error trapping process and, 330
 exceptions not handled and, 338
 MortimerColdCall example, 342–343
 multiple blocks, 331, 332–336
 omitting, 331
 for releasing database resources, 586, 587–588
 in SimpleExceptions class, 335–336
 syntax, 331
 throwing an exception and, 331
 throwing exceptions from, 339

CCOMDemo **class**, **1161, 1162–1163, 1167**

CCW (COM callable wrapper), **1172**

certificates. See digital certificates

channels (.NET Remoting)

created on both server and client side, 1021
 defined, 1012, 1020
 delayed loading of client channels, 1046
 getting configuration information for, 1022–1023
 HTTP channel, 1021
 interfaces for, 1022
 IPC channel, 1021
 listening to multiple channel by server, 1021–1022
 marshal-by-value classes and, 1031
 pluggability of, 1024–1025
 predefined, 1041–1042
 server-side creation, 1021
 setting properties for, 1023–1024
 ShowChannelProperties() method, 1023
 TCP channel, 1021

ChannelServices **class**, **1012, 1025–1026**

char **type**, **42, 43–44**

CheckBox **control (Windows Forms)**, **764**

checked **operator**, **137, 162**

CheckedListBox control (Windows Forms)

CheckedListBox control (Windows Forms), 765

chktrust.exe utility, 501

class hierarchies

for data-binding objects, 817–818

DataGridView control, 813–816

for exceptions, 329–330

for streams, 1206

for WebRequest and WebResponse classes,
1257–1258

for Windows Forms, 757–758

class id (CLSID) for COM objects, 1155

class keyword, 32

Class View window (Visual Studio 2005), 396–397

classes. *See also* class hierarchies; derived classes;

specific classes and members

abstract, 117

accessing fields outside the object, 37–38

ADO.NET, 581–582

ASP.NET pages as, 22

constructors not required for, 95

context attribute classes, 1015–1016

context property classes, 1016

data members, 85

declaring instances, 84

default initialization of variables in, 34
defined, 84

for file system operations, 1188

for fonts and font families, 877

formatter classes (.NET Remoting), 1025

function members, 85–99

generic collection classes, 276–277

generic, custom, 293–296

generic .NET base classes, 277–278

marking members unsafe, 205

members, defined, 85

namespaces for, 21, 58–59

.NET base classes, 19–20

.NET Remoting and, 1031–1032

non-compliant with CLS, 426

not-remotable, 1032

partial, 104–106

Pascal casing for, 76

for performance monitoring, 1309

pointers to members, 212–214

pointers to, not allowed, 208, 212

readonly fields, 99–101

as reference types, 40, 84

registry classes, 1229–1232

for regular expressions, 223

for resources, 527–528, 533–534

sealed, 117–118

static, 106

stored in the heap, 84

structs versus, 84, 102

System.DirectoryServices namespace, 725

System.Xml.XPath namespace, 679

in Visual Studio 2005 Class View window, 396–397

in Visual Studio 2005 Object Browser window, 397–398

for Windows Services, 1275

XML-related, 660–661

ClearAllFields() method

FileProperties application, 1194

FilePropertiesAndMovement application, 1201

ClickOnce deployment technology, 568–569

advanced options, 570–576

application cache, 569

Custom Actions Editor, 573–575

dialog box types available, 572–573

File System Editor, 570

File Types Editor, 571

Launch Conditions Editor, 575–576

mage.exe command-line tool, 568

mageUI.exe gui tool, 568

overview, 553, 567

properties for actions, 575

publishing an application, 568

Registry Editor, 570–571

security settings, 569–570

settings, 568–569

User Interface Editor, 571–573

Windows Installer compared to, 567–568

XML-based manifest files, 567

client applications

ASP.NET client-side code, 905–906

COM client creation, 1179–1181

COM client with a sink object, 1182–1183

creating Web service clients, 991–992

for DatabaseResourceReader class, 549

event-booking client, 993–997

fat-client, 22, 24

hardware requirements for deployment, 553

.NET Remoting simple client, 1019–1020

using a COM component from a .NET client,
1159–1171

using a .NET component from a COM client,
1171–1183

using UDTs from client-side code, 642–643

WCF, 1144

ClientActivated_Client.config file (.NET
Remoting), 1044

`ClientActivated_Server.config` file (.NET

Remoting), 1043–1044

clipping region, 848–850

closing

database connections with ADO.NET, 583, 585–588
forms, 782
streams, 1209, 1215

CLR (Common Language Runtime)

application domains, 634
C++ and memory safety tests, 7
CLR Object Remoting, 1011
code access permissions provided by, 471–472
compilation steps for, 4
defined, 4
demanding permissions and, 478
enabling .NET code to run with SQL Server, 634
garbage collector, 13–14
heap maintained by, 13
Identity Permissions, 472–473
managed code advantages, 4–7
passing arguments to `Main()`, 63
stored procedures with SQL Server, 646–647
Virtual Execution System, 465

CLS (Common Language Specification)

C# class for example, 434–436
C++/CLI class for example, 427–432
CLSCompliant attribute, 437
CTS and, 425–426
defined, 12
language independence using, 427–436
language interoperability and, 425, 426
marking assemblies as CLS-compliant, 437
non-compliant classes and methods, 426
overview, 12–13
requirements, 436–438
rules for compliance, 438
Visual Basic class for example, 432–434
writing non-CLS-compliant code, 12

CLSCompliant attribute, 437

CLSID (class id) for COM objects, 1155

code access security. See also code groups; permissions for code access

caspol.exe (Code Access Security Policy tool), 466–471
code groups, 464, 465–471
as code-based security, 14
distributing code using certificates, 499–504
distributing code using strong names, 497–499
importance of, 464
overview, 464
permissions, 464, 471–476

policy levels, 476–478

Virtual Execution System and, 465

Code Access Security Policy tool. See caspol.exe

code bloat, generics and, 274

code groups. See also permissions for code access

applying full trust, 492, 502
caspol.exe (Code Access Security Policy tool), 466–471
changing permissions for, 494–495
creating custom, 493–494
deleting, 494
evidence, 464
getting labels for, 491
labels for, 469
levels, 470
listing compactly, 468
listing with descriptions, 466–467
managing, 492
membership conditions, 465–466
overview, 464
specifying trust level, 468
viewing code groups at enterprise level, 477–478
viewing code groups at user level, 476–477
viewing for assemblies, 469–471
Zone condition, 465, 504–506

code-behind feature (ASP.NET)

event-booking application code-behind file, 929–930
overview, 22
PCSWebApp1 example code-behind file, 909

ColdCallFileFormatException, 341, 346–347

ColdCallFileReader class

declaring variables, 343
described, 341
`Dispose()` method, 344, 345–346
`NPeopleToRing` property, 345
`Open()` method, 343–344
`ProcessNextPerson()` method, 344–345
using, 342

collections

Active Directory object collections, 731–732
arrays as, 247–248
`Current` property of elements, 247
defined, 246
enumerators, 246–247, 248
foreach loops for iterating through, 54, 246
generic, displaying with `DataGridView` control, 811–813
`ICollection` interface for, 247
`IEnumerable` interface for, 246–247
`Reset()` method for returning to start, 247
`Vector` struct example, 248–250

colon (:)

colon (:)

- for disabling Group object with regular expressions (?:), 243
- as namespace alias qualifier (::), 60–61
- preceding format specifiers, 230
- in ternary operator (?:), 136–137

Color struct, 863

colors in graphics applications

- GDI+ struct for representing, 863
- graphics display modes and, 865
- named colors, 864–865
- palettes, 865–866
- RGB values, 863–864, 865
- safety palette, 866

COM

- ActiveX controls in Windows Forms, 1168–1171
- apartment models, 1156–1157, 1166
- attributes and, 17
- casting and QueryInterface() method, 1154–1155
- class id (CLSID) for objects, 1155
- connection points, 1166–1168, 1181–1182
- creating a client with a sink object, 1182–1183
- creating a COM callable wrapper (CCW), 1172
- creating a component, 1159–1163
- creating a runtime callable wrapper (RCW), 1163–1166
- creating a simple client, 1179–1181
- custom interfaces, 1153
- data types, 1155
- debugging issues, 9
- delegate for, 1181
- dispatch interfaces, 1153–1154
- dual interfaces, 1154
- Enterprise Services history and, 1066
- error handling, 1158
- event handling, 1158, 1166–1168, 1181–1182
- freeing memory, 1153
- _ICompletedEvents client interface, 1166–1167
- IDispatch interface, 1163, 1164
- IMath interface, 1162, 1174, 1177–1178
- instantiating components, 1165
- interfaces, .NET interfaces versus, 8, 126
- interfaces overview, 1153–1155
- interop attributes, 1175–1178
- interoperability with .NET (overview), 7
- IUnknown interface, 1163, 1164
- IWelcome interface, 1160–1161, 1162, 1174, 1177–1178
- language interoperability and, 8–9, 425
- marshaling, 1159

- memory management, 13
- metadata, 1152
- method binding, 1155
- MSXML based on, 662
- multi-threaded apartment (MTA), 1157, 1166
- namespace for objects, 1151
- .NET and, 1152–1159
- performance losses with, 8
- primary interop assemblies, 1166
- prog id for objects, 1155
- registration, 1155, 1178–1179
- releasing components, 1165
- running Windows Forms controls in Internet Explorer, 1183
- similar concepts in .NET, 1152
- single-threaded apartment (STA), 1156–1157, 1166
- threading, 1156–1157, 1166
- tlbexp utility, 1173
- tlbimp utility, 1163
- type library for, 1173–1175
- using a COM component from a .NET client, 1159–1171
- using a .NET component from a COM client, 1171–1183
- using objects in ASP.NET, 1171

COM callable wrapper (CCW), 1172

COM+

- contexts, .NET contexts compared to, 1014
- Enterprise Services history and, 1066
- interoperability with .NET (overview), 7
- simple Enterprise Services application, 1070–1073

combined characters. See Unicode characters

ComboBox control (Windows Forms), 765, 767

comma (,) in derived class or struct declarations, 113

comments

- multi-line, 31, 67–68
- single-line, 31, 67
- within source files, 67–68
- XML documentation, 68–70

Common Language Runtime. See CLR

Common Language Specification. See CLS

Common Type System. See CTS

comparison operators

- assignment versus comparison, 48, 134–135
- comparing reference types for equality, 147
- comparing value types for equality, 147
- overloading, 148, 155–157
- table listing, 134

compiling. See also building

- casts combined during, 165–166
- class library example, 64–65

- constructors and, 95, 119–120
- csc.exe command-line compiler for, 31
- custom attributes and, 306–307
- destructors and, 199, 200
- hiding methods and errors in, 116
- inlining by compiler, 94–95
- JIT (Just-In-Time) compilation, 5
- modules, 421
- MSIL code to native code, 439
- multiple Main() methods and, 61–62
- operators and, 149–150
- optimization in Visual Studio 2005, 399–400
- /out option for output file specification, 64
- performance improvement with IL, 5
- pointer arithmetic and, 210
- preprocessor directives for, 70–73
- PrintingCapsEdit project, 898
- /reference or /r switch for reference types, 64
- rules for identifiers and, 73–75
- steps in .NET, 4
- /target or /t switch for file type, 63–64
- TypeView assembly, 319
- usage conventions and, 75
- from Visual Studio 2005, 372, 399–403
- XML comments, 68–70
- Component **class**, **755**
- Component Services explorer, 1075–1077**
- Computer Management MMC snap-in, 1101–1102**
- concatenation, operator overloads for, 224**
- conditional statements, 47. See also if statements;**
switch...case **statements**
- configuration files (.NET Remoting)**
 - client code using, 1045
 - client configuration file for event handling, 1061
 - client configuration for client-activated objects, 1044
 - client configuration for well-known objects, 1043
 - debugging, 1046–1047
 - delayed loading of client channels, 1046
 - example online, 1039–1040
 - formatter providers, 1047–1048
 - lifetime services, 1047
 - main elements and attributes, 1040–1041
 - .NET Framework Configuration tool for, 1048–1050
 - .NET platform files versus, 1039
 - overview, 1039–1041
 - predefined channels, 1041–1042
 - RemotingConfiguration.Configure() method and, 1046
 - server code using, 1044–1045
 - server configuration file for event handling, 1059
 - server configuration for client-activated objects, 1043–1044
 - server configuration for well-known objects, 1042–1043
- Configure() **method (RemotingConfiguration class), 1044, 1045, 1046**
- configuring assemblies**
 - application configuration files, 450, 453–456
 - categories, 450
 - <codeBase> configuration, 460–461
 - configuration files used instead of registry, 449
 - directories, 460–462
 - machine configuration files, 450
 - overview, 450–451
 - <probing> configuration, 461–462
 - publisher policy files, 450, 456–458
 - remoting settings, 450
 - runtime settings, 450
 - runtime version, 459–460
 - security settings, 450
 - startup settings, 450
 - versioning, 451–460
- Connect() **method (RemotingServices class), 1028, 1029, 1030**
- connection points (COM), 1166–1168, 1181–1182**
- Connection Properties dialog box (Visual Studio), 822–824**
- consistency of Enterprise Services transactions, 1078**
- console applications**
 - creating in Visual Studio 2005, 377–383
 - defined, 21
 - for Enterprise Services client, 1077
 - first C# program, 30–33
 - .NET Remoting client, 1019–1020
 - .NET Remoting server, 1018–1019
 - role-based security example, 508–509
 - TestQuoteServer, 1279
 - threading with ThreadPool class, 365–368
 - uses for, 21
- console I/O**
 - displaying formatted output, 65–67
 - format strings, 66–67
 - method for reading, 65
 - methods for writing, 65
 - placeholder characters for output, 67
- Console.ReadLine() **method, 65**
- Console.Write() **method, 65**
- Console.WriteLine() **method**
 - displaying formatted output, 65–67
 - formatting expressions, 229–230, 231

const keyword

const **keyword**, 38, 85

constants

- advantages of using, 38–39
- associating with classes, 85
- in C++ versus C#, 38
- characteristics, 38
- as data members of classes, 85
- declaring, 38, 85
- defined, 38
- Pascal casing for, 76
- readonly fields versus, 99

constraints, database

- naming conventions, 630
- setting update and delete constraints, 611–612

constructors

- calling from other constructors, 98–99
- compiler generation of, 95, 119–120
- defined, 86
- of derived classes, 118–124
- FileStream class, 1208–1209
- initializers, 99
- instance and static in same class, 97
- no-parameter, adding in a hierarchy, 120–122
- not required for classes, 95
- overloading, 95
- with parameters, adding in a hierarchy, 122–124
- public or protected, 96
- QuoteServer class, 1277
- static, 96–98
- StreamReader class, 1214
- StreamWriter class, 1216
- for structs, 104
- syntax, 95
- Thread class, 352
- of Windows Form classes, 755–756

consuming Web services, 990–993

ContextDataRow **class**, 834

ContextMenuAttribute, 834, 835, 837

ContextMenuStrip **class (Windows Forms)**, 779

contexts (.NET Remoting)

- activation, 1015
- attributes, 1015–1016
- call contexts, 1062–1064
- COM+ contexts compared to, 1014
- communication between, 1016
- context-bound objects, 1015
- Enterprise Services and, 1067–1068
- interceptors, 1015
- overview, 1014–1015
- properties, 1015, 1016

continue **statement**, 55

Control class

- appearance properties, 760
- as base class for controls, 759
- Form class inheritance from, 780
- miscellaneous functionality, 762
- overview, 759
- size and location properties, 759–760
- user interaction and, 760–761
- Windows functionality, 761–762

control palette (ASP.NET), 913–914

ControlBindingsCollection **class**, 817, 819

ConvertAll() **method (List<T> generic class)**, 282–283

converting. See also generics

- ADO.NET relational data to XML, 694–696
- ADO.NET single table data to XML, 689–694
- array lists to arrays, 253
- enumerations from strings, 57
- explicit type conversions, 142–145
- implicit type conversions, 141–142
- need for type conversions, 141
- nullable types and, 142
- between numeric and string types, 144–145
- types with List<T> generic class, 282–283
- UDTs to and from strings, 640–641
- between value and reference types, 145–146
- to Visual Studio 2005 projects from previous versions, 374–376
- XML to ADO.NET data, 696–699

CookieContainer **class**, 1004–1005

Coordinate struct

- changing User-Defined Type template for, 637
- constructors to initialize variables, 639
- converting from a string, 641
- converting to a string, 640–641
- deploying, 642
- get accessor, 640
- INullable interface, 637
- IsNull property, 640
- Null property, 640
- Orientation enumeration for, 639
- serialization formats, 637–638
- [SqlUserDefinedType] attribute, 637, 638
- using from client-side code, 642–643

coordinates

- converting relative to client area, 861–862
- device coordinates, 863
- page coordinates, 863
- Point and PointF structs for, 851–852

- Rectangle and RectangleF structs for, 853–854
 - Region struct for, 854–855
 - Size and SizeF structs for, 852–853
 - System.Drawing structs for, 850
 - transforms in CapsEditor example, 889–890
 - world coordinates, 863
 - Copy() **method (File class)**, **1197**
 - Copy Web tool, deployment using**, **552, 555**
 - copying files**, **1197–1201**
 - CopyTo() **method (FileInfo class)**, **1197**
 - Course **class**, **1111–1112**
 - course order application (Message Queuing)**
 - assemblies, 1111
 - class library, 1111–1113
 - Course class, 1111–1112
 - CourseOrder class, 1113
 - CourseOrderSender application, 1114
 - Customer class, 1112–1113
 - Invoke() method, 1117
 - LableIdMapping class, 1117–1118
 - message receiver, 1116–1119
 - message sender, 1114
 - OnOrderSelectionChanged() method, 1118–1119
 - OnSubmitCourseOrder() method, 1114
 - PeekMessages() method, 1116–1117
 - ReceivedById() method, 1119
 - sending priority and recoverable messages, 1115
 - CourseOrder **class**, **1113**
 - Create() **method**
 - MessageQueue class, 1103, 1121
 - XmlReader class, 665
 - CreateFonts() **helper method**, **883–884**
 - CreateGraphics() **method (Form class)**, **845**
 - CreateInstance() **method (Activator class)**, **1027, 1028, 1029**
 - CreateSubKey() **method (RegistryKey class)**, **1230**
 - cross-language support**. **See CLS (Common Language Specification); language interoperability with .NET**
 - Crystal Reports Web server controls (ASP.NET)**, **914**
 - .cs file extension**, **30**
 - csc.exe **command-line compiler**
 - compiling programs using, 31
 - creating modules using, 421
 - CTS (Common Type System)**
 - CLS and, 425–426
 - hierarchy of types, 11–12
 - language interoperability and, 425
 - overview, 10–12
 - culturalization**. **See globalization; localization**
 - CultureAndRegionInfoBuilder **class**, **549–550**
 - CultureInfo **class**, **515, 518**
 - curly braces ({})**
 - with if statements, 47, 48
 - joining statements in blocks, 31
 - Currency **struct**, **159–162**
 - CurrencyManager **class**, **819, 820–822**
 - Current **property of collection elements**, **247**
 - Custom Actions Editor (ClickOnce)**, **573–575**
 - custom attributes**
 - AttributeUsage attribute and, 307–309
 - compilation and, 306–307
 - emitted as metadata, 306
 - FieldName attribute example, 307–310
 - finding dependencies for, 320
 - required specifications, 307
 - for serializing objects in XML, 702
 - specifying optional parameters, 309–310
 - specifying required parameters, 309
 - WhatsNewAttributes example, 310–313
 - custom controls (ASP.NET)**
 - complexity of, 952
 - deriving from WebControl class, 952
 - .NET Framework and, 944
 - RainbowLabel example, 953–956
 - <%@ Register %> directive for, 952–953
 - user controls versus, 952
 - using in ASP.NET pages, 952–953
 - using statements, 953
 - custom controls (Windows Forms)**
 - attributes useful to add to, 787–788
 - deriving from existing controls, 787
 - requirements for, 788
 - starting from scratch, 787
 - TreeView-based example, 788–794
 - user controls, 794–799
 - custom cultures**, **549–550**
 - custom generic classes**
 - constraints, 294–296
 - default values, 294
 - defining, 293
 - DocumentManager class example, 293–294
 - custom interfaces (COM)**, **1153**
 - Customer **class**, **1112–1113**
 - CustomerTable **class**, **835–837**
 - Cycle() **method**, **954**
- D**
- data access with .NET**. **See ADO.NET; viewing .NET data; Visual Studio .NET and data access**

data adapters

- deleting rows with, 623
- flexibility of, 621
- inserting new rows with, 621–623
- populating a `DataSet` with stored procedure in, 619–620
- `UpdateRowSource` property and, 622–623
- updating existing rows with, 623

data binding. See also `DataGridView` control; event-booking application

- `Binding` class, 819–820
- `BindingContext` class, 818–819
- class hierarchy for objects, 817–818
- `CurrencyManager` class, 819, 820–822
- data display with templates, 935–936
- data-binding objects, 817–822
- `DataGridView` control capabilities, 801–802
- defined, 816
- Dialog Data Exchange (DDX) versus, 816–817
- `PropertyManager` class, 819, 820–822
- `ScrollingDataBinding` example, 821–822
- simple binding, 816–817
- Web Forms and, 926–939

data members of classes, 85–86. See also constants; events; fields (member variables)

data types. See also reference types; UDTs (user-defined types); value types; specific types

- blittable, 638
- casting, 143–145
- COM, 1155
- CTS and, 10–12, 40–41
- enumerations, 55–57
- explicit conversions, 142–145
- hierarchy of, 11–12
- implicit conversions, 141–142
- modifiers and, 124–125
- namespaces for, 21, 59
- naming generic types, 274
- nesting, 125
- nullable types, 139, 142, 144, 301–302
- pointers and, 208
- predefined, 41–46
- primitive types as .NET structs, 40–41
- strong typing in IL, 9–16
- strongly typed resources, 531–533
- type library for COM, 1173–1175
- type safety, 7, 134, 140–146
- user-defined casts, 157–169
- value types versus reference types, 9, 39–40

- variable declarations and, 34
- for Web services, 990

data Web server controls (ASP.NET)

- data display controls, 918
- data source controls, 917–918
- templates with, 935–939

database access. See ADO.NET; viewing .NET data; Visual Studio .NET and data access

database connections

- closing with ADO.NET, 583, 585–588
- creating with Visual Studio .NET, 822–825
- ensuring resources are released, 583–588
- managing connection strings with ADO.NET, 584–585
- opening with ADO.NET, 583

database server. See SQL Server 2005

`DatabaseResourceManager` class, 548–549

`DatabaseResourceReader` class

- client application for, 549
- creating, 546–547
- database columns and values for example, 545–546

`DatabaseResourceSet` class, 547–548

`DataColumn` class

- generated for XSD schema, 831–832
- naming conventions for columns, 629
- overview, 602–603

`DataGrid` control

- `DataGridView` control versus, 801
- getting selected row with, 839–840

`DataGridView` control

- for array data display, 804–805
- binding capabilities, 801–802
- class hierarchy, 813–816
- constructing columns for, 814–816
- `DataGrid` control versus, 801
- `DataGridViewColumn` objects utilized by, 814
- `DataMember` property, 804
- for `DataSet` display, 809–811
- `DataSource` property capabilities, 803
- for `DataTable` display, 805
- for `DataView` display, 806–809
- `DisplayTabularData` application, 802–803
- for generic collection display, 811–813
- `IListSource` and `IList` interfaces with, 811

`DataList` control, 937–938

`DataRow` class

- altering data, 605–606
- generated for XSD schema, 830–831
- overview, 603–604
- for pop-up menu for a database row, 832

- RowState property, 605
- versioning by, 604–605
- DataSet class. See also DataColumn class;**
 - DataRow **class**; DataTable **class**
 - converting relational data to XML, 694–696
 - converting single table data to XML, 689–694
 - converting XML to ADO.NET data, 696–699
 - copying a DataSet, 625–626
 - data columns, 602–603
 - data constraints, 609–612
 - data relationships, 608–609
 - data rows, 603–606
 - data tables, 601–602
 - DataGridView control for displaying data, 805, 809–810, 814–816
 - DataViewManager for displaying data, 810–811
 - overview, 600–601
 - persisting DataSet changes, 620–625, 691–692
 - populating a DataSet, 619–620
 - ReadXml () method, 620, 696–697
 - ReadXmlSchema () method, 697
 - Recordset object versus, 609
 - schema generation, 606–608, 691
 - tiered development and, 625–626
 - Web services with DataSet objects, 997
 - WriteXml () method, 624–625, 690–691
 - writing XML output, 623–625, 690–691
- DataSource property (DataGridView control)**
 - arrays as data source for, 804–805
 - capabilities, 803
 - DataSet class as data source for, 809–811
 - DataTable class as data source for, 805
 - DataView object as data source for, 806–809
 - generic collections as data source for, 811–813
- DataTable class**
 - binding column to TextBox, 816–817
 - creating DataView from existing DataTable, 806
 - data-access methods, 835–837
 - DataGridView control for displaying data, 805
 - generated for XSD schema, 829–830
 - naming conventions for tables, 629
 - overview, 601–602
 - for pop-up menu for a database row, 832
 - properties of objects, 602
 - setting a foreign key, 610–611
 - setting a primary key, 610
 - setting update and delete constraints, 611–612
- DataView object, 806–809**
 - DataViewManager, **displaying DataSet data in, 810–811**
 - date formatting**
 - for DateTimePicker control, 767
 - for globalization, 519–520
 - DateTime **structure, 519**
 - DateTimePicker **control (Windows Forms)**
 - hosting on a ToolStrip, 777–779
 - overview, 765
 - DCOM protocol, 1126**
 - DCs (device contexts) of GDI+, 843–844**
 - DDX (Dialog Data Exchange), 816–817**
 - deadlocks, avoiding, 361–363, 364**
 - debugging**
 - COM issues for, 9
 - configuration files (.NET Remoting), 1046–1047
 - debug builds for, 399–401
 - exceptions and, 406–407
 - graphics and, 855–856
 - removing debugging commands (Visual Studio), 401
 - setting breakpoints, 404
 - symbols in Visual Studio, 400–401
 - Visual Studio 2005 debugger for, 9, 373, 404–407
 - Watch window for (Visual Studio), 405–406
 - Windows Services, 1301–1302
 - decimal **type**
 - overview, 42–43
 - placeholder characters for output, 67
 - declarative security**
 - for code access, 488
 - role-based, 509–510
 - declaring. See also instantiating**
 - abstract classes and functions, 117
 - constants, 38, 85
 - custom generic classes, 293
 - delegates, 173–174
 - enumerations, 55, 56
 - events, 188
 - methods, 86–87
 - permissions (declarative security), 488
 - pointers, 206–207
 - properties, 93
 - role-based security, 509–510
 - sealed classes and methods, 117–118
 - user-defined casts, 158–159
 - variables, 33–34
 - virtual methods, 114
 - virtual properties, 114

#define preprocessor directive

`#define` **preprocessor directive, 70, 71**

defining. See declaring

delegates. See also events

with anonymous methods, 176–185

with asynchronous .NET Remoting, 1053–1054

BubbleSorter example, 180–182

for COM, 1181

declaring, 173–174

delegateTest example, 176–177

EventHandler example, 186–188

EventHandler<TEventArgs> generic delegate,
302–303

for events, 185

generic, 299–301

instantiating, 173–174, 176–177

interfaces and, 181

multicast, 183–185

in .NET Framework, 188

passing without parameters, 179

proxy class for, 838

SimpleDelegate example, 177–180

syntax, 173

for threading, 352, 353, 366

type safety and, 172, 174

uses for, 171–172

using, 174–176

delegateTest **delegate, 176–177**

Delete() **method, 1197**

deleting or removing

array list elements, 251–253

code groups, 494

database records with ADO.NET, 591, 595–596

database rows with data adapters, 623

debugging commands with Visual Studio, 401

dictionary items, 261

files, 1197–1201

folders, 1197

method calls from multicast delegates, 183

setting delete constraints for databases, 611–612

demanding permissions, 478, 479–480

denying permissions, 485–486

deployment

automatic, for Enterprise Services components, 1073

ClickOnce technology for, 553, 567–576

Copy Web tool for, 552, 555

designing for, 551–552

including deployment project in applications solution,
563–564

installing client from Web server, 566–567

manual, for Enterprise Services components,
1073–1074

No Touch Deployment (NTD), 566–567

options available to .NET developers, 552–553

publishing Web sites for, 552, 555–556

requirements, 553

simple deployment, 554–556

simple Web application installer, 564–565

of triggers for SQL Server, 650

of UDTs, 642

of user-defined aggregates, 645

Windows Installer for, 552, 556–567

xcopy for, 552, 554–555

Dequeue() **method (Queue class), 256**

derived classes

calling base versions of methods, 116–117

casts between, 162–163

casts between base and derived classes, 163–164

commas in declarations, 113

constructors, 118–124

instantiating, 120

from interfaces, 126–127

Object class as parent of all classes, 106

syntax, 113

derived interfaces, 131–132

derived structs, 113

Design View window (Visual Studio 2005), 391–394

destructors. See finalizers

device contexts (DCs) of GDI+, 843–844

device coordinates, 863

DHTML versus ASP.NET, 22

Dialog Data Exchange (DDX), 816–817

dictionaries

adding objects to, 261, 262

algorithm for, 260, 262–263

defined, 259

getting details from, 259–260, 261

Hashtable class for, 261–262

how they work, 262–264

key class for, 262–263

keys, 260, 262

MortimerPhonesEmployees example, 264–269

in .NET, 261–262

overriding Equals method for, 264, 266

overriding GetHashCode() method for, 263–264,
266–267

performance compared to array lists, 259

real-life dictionaries compared to, 260

removing items from, 261

- requirements for building, 260
- uses for, 259, 260
- DiffGram **documents**
 - defined, 699
 - reading, 700–701
 - uses for, 699
 - writing, 699–700
- digital certificates**
 - applying FullTrust permission to assemblies, 502
 - buying versus creating, 499
 - checking code groups for your assembly, 503–504
 - configuring the test certificate, 501
 - creating a test certificate, 499
 - distributing code using, 499–504
 - getting a hexadecimal representation, 502
 - signing assemblies with, 499–501
 - verifying the trust level, 501, 503
- directories. See folders or directories**
- Directory **class**
 - Delete() method, 1197
 - described, 1188
 - DirectoryInfo class versus, 1189
 - Move() method, 1197
- Directory Services Markup Language. See DSML**
- DirectoryEntries **class, 733**
- DirectoryEntry **class**
 - binding Active Directory with, 729
 - CommitChanges() method, 733, 734
 - RefreshCache() method, 733
- DirectoryInfo **class**
 - Delete() method, 1197
 - described, 1188
 - Directory class versus, 1189
 - Exists property, 1189–1190
 - instantiating, 1189
 - methods, 1191–1192
 - properties, 1190, 1191
 - streams versus, 1206
- DirectorySearcher **class**
 - filters, 736–737
 - LDAP provider required for, 736
 - PropertiesToLoad, 737
 - read-mostly access and, 736
 - SearchRoot, 736
 - SearchScope, 737
 - setting search limits, 737–740
- DisableMoveFeatures **utility function, 1201**
- dispatch interfaces (COM), 1153–1154**
- DisplayAtStartup **example, 844–847**
- DisplayFileInfo() **method, 1194–1195, 1196**
- DisplayFolderList() **method**
 - FileProperties application, 1195, 1196, 1197
 - FilePropertiesAndMovement application, 1200–1201
- DisplayImage **project, 871–873**
- displaying images**
 - example, 870–873
 - issues when manipulating images, 873–874
- DisplayNames() **method, 525**
- DisplayTabularData **application, 802–803**
- DisplayText **example, 874–876**
- DisplayTypeInfo() **method, 322–323**
- Dispose() **method**
 - ColdCallFileReader class, 344, 345–346
 - Component class, 755
 - Form1 class, 872
 - IDisposable interface, 201–202
- distinguished name (DN) in Active Directory, 727–728, 729**
- distributed identity of remote objects, 1017**
- distributed programming's future. See WCF; Web services specifications**
- distributed transactions (Enterprise Services), 1068**
- distributing code**
 - certificates for, 499–504
 - strong names for, 497–499
- DLLs**
 - assemblies as answer to DLL Hell, 412–413
 - problems with (DLL Hell), 411–412
 - side-by-side feature for, 412
- DN (distinguished name) in Active Directory, 727–728, 729**
- Dns **class, 1260–1261**
- DNS servers**
 - Dns class, 1260–1261
 - DnsLookup application, 1261–1262
 - IP addresses and, 1259
- DnsLookup **application, 1261–1262**
- Document **class**
 - LinkedList<T> generic class example, 287, 289
 - Queue<T> generic class example, 284
- document management applications**
 - using custom generic class, 293–296
 - using LinkedList<T> generic class, 287–292
 - using Queue<T> generic class, 283–286
- DocumentManager **class**
 - custom generic class example, 293–294
 - Queue<T> generic class example, 284–285

DocumentTitleChanged event

`DocumentTitleChanged` event, **1248–1249**

DOM (document object model)

using MSXML 4.0 in .NET, 661–663

`XmlDocument` class as implementation of, 659, 673

`DotnetComponent` class, **1172–1173, 1175, 1181–1182**

double quotes ("), string literals enclosed by, 46

`double` type, **42**

`do...while` statements, **53–54**

`DownloadFile()` method (`WebClient` class), **1240**

`DrawEllipse()` method, **overloading, 845**

`DrawImage()` method (`Graphics` class), **874**

`DrawImageUnscaled()` method (`Graphics` class), **873**

drawing. See also graphics

lines, 868–870

scrollable windows, 856–862

shapes, 844–847, 868–870

text, 874–880

`DrawRectangle()` method, **overloading, 845**

`DrawString()` method (`Graphics` class), **874, 875–876**

`DriveInfo` class, **1188, 1220–1222**

drives, reading information from, 1220–1222

DSML (Directory Services Markup Language)

configuration scenario for Active Directory, 745–746

namespace for, 724, 745

searching Active Directory objects with, 746–748

standardized Web interfaces for, 724

`System.DirectoryServices.Protocols` classes, 746

dual interfaces (COM), 1154

durability of Enterprise Services transactions, 1078

E

early binding (COM), 1155

`#elif` preprocessor directive, **71–72**

`#else` preprocessor directive, **71–72**

`EnableVisualStyles` method (`Application` class), **785–786**

`#endif` preprocessor directive, **71–72**

`EndOfStreamException` class, **329**

`EndReceive()` method (`MessageQueue` class), **1111**

`#endregion` preprocessor directive, **72–73**

Enterprise Services

automatic transactions, 1068, 1078

business service layer for, 1067

client application, 1077–1078

Component Services explorer for, 1075–1077

contexts and, 1067–1068

current issues, 1126

data service layer for, 1067

defined, 1065

deployment, 1073–1075

distributed transactions, 1068

history of, 1066

installer packages for, 1074–1075

loosely coupled events, 1069

Microsoft Application Center Server and, 1067

object pooling, 1068

overview, 1065–1069

presentation service layer for, 1067

queued components, 1069

role-based security, 1068

sample application, 1080–1090

`ServiceConfig` class, 1090–1091

`ServiceDomain` class, 1090, 1091

services without components, 1069, 1090–1093

simple COM+ application, 1070–1073

transactions, 1068, 1078–1080

WCF and, 1146–1147

enumerations (enums)

benefits of using, 56

collection enumerators, 246–247

converting from a string, 57

defining, 55, 56

described, 55

`EnumFontFamilies` example, 878–880

`FileStream` class, 1208

instantiated as structs, 57

for messages in message queues, 1109–1110

retrieving string representations, 57

retrieving values, 57

`TimeOfDay` example, 56

enumerators for collections, 248

`EnumFontFamilies` example, **878–880**

`Equals()` method

comparing reference types for equality, 146–147

comparing value types for equality, 147, 148

described, 107

overriding for comparisons, 146–147, 148

overriding for dictionaries, 264, 266, 267

static version, 147

virtual version, 146–147

equals sign (=)

as assignment operator, 34, 134–135

as comparison operator (==), 48, 134–135, 147

error handling. See also exceptions

- catch blocks, defined, 330
- catching exceptions from other code, 336–337
- catching exceptions from predefined classes, 330–340
- catching user-defined exceptions, 341–343
- choosing exception type to throw, 334–335
- for code access permissions, 472
- COM, 1158
- defining your own exception classes, 346–348
- exceptions for, 16–17
- exceptions not handled and, 338
- finally blocks, defined, 330
- handling different exceptions in different places, 340
- modifying the type of exception, 340
- multiple catch blocks for, 331, 332–336
- nested throw statements for, 332
- nested try blocks for, 338–340
- for overflow and array out-of-bounds, 331–332
- process for, 330
- for releasing database resources, 586, 587–588
- SimpleExceptions class, 333–336
- syntax, 330–331
- SystemException properties for, 337
- throwing an exception, defined, 331
- throwing user-defined exceptions, 343–346
- for thrown SecurityException, 480, 482, 492
- try blocks, defined, 330
- for Windows Forms controls, 767–769

#error preprocessor directive, 72**ErrorProvider component (Windows Forms)**

- clearing, 768
- overview, 767–768
- using one error provider per form, 768–769

escape sequences

- for regular expressions, 236, 238–240
- searching for escape meta-characters, 240
- table summarizing, 44, 239

EvaluateException, 807**event logging for services**

- adding for other application types, 1305–1306
- adding for service classes derived from
 - ServiceBase, 1302, 1305
- adding to QuoteServer, 1306–1307
- architecture, 1304
- AutoLog property of ServiceBase class for, 1302, 1305
- classes, 1305
- creating an event log listener, 1307–1308
- custom event logging, 1303

example log entry, 1302–1303

trace messages and, 1307

event sink library (.NET Remoting), 1059–1060

EventArgs class, 188–189, 832

event-booking application

- AddEvent() method, 995
- adding events to the database, 931–933
- ADO.NET and data binding for, 926–933
- Application_Start() event handler, 994
- attendeeList control, 928
- Attendees table, 926–927
- binding to controls on the Web page, 999
- binding to the database, 927–928
- calendar_DayRender() event handler, 930–931
- calendar_SelectionChanged() event handler, 930
- client for Web service extension, 997–1001
- code-behind file, 929–930
- creating the Web site, 920–921, 926
- customizing the calendar control, 928–931
- data binding for client application, 1000
- data display with templates, 935–939
- database for, 926–927
- DataList control, 937–938
- DataSet objects with Web services, 997
- declaring objects needed for database access, 995–996
- deleting data sources for client application, 998
- event list display, 933–935
- EventList control, 936–937
- EventList_SelectedChanged() method, 1001
- Events table, 927
- ExecuteNonQuery() method, 932–933
- extracting event data, 930
- form for, 920–923
- FormView control, 936, 937–938
- GetData() method, 994–995
- GetFreeDate() method, 929–930, 933
- GridView control, 933–934
- Label control, 920
- Lock() and Unlock() methods, 996
- MRBEventData control, 929
- MRBService service, 994–995
- Page_Load() event handler, 924
- queryResult with, 996–997
- resultLabel control, 924
- results of a submission, 924–925
- roomList control, 928
- Rooms table, 927
- SqlDataSource controls, 927–928
- storing the DataSet for client application, 998–999

event-booking application (continued)

event-booking application (continued)

- submitButton_Click() event handler, 924, 931–932, 934, 1000–1001
- using statements, 995
- ValidationSummary control, 921
- Web server controls for, 921–924
- Web service for, 993–997
- web.config file, 993
- EventHandler **delegate, 186–188**
- EventHandler<TEventArgs> **generic delegate, 302–303**
- EventList **control, 936–937**
- EventLog **component, 1307–1308**
- events**
 - Click event example, 186–188
 - COM event handling, 1158, 1166–1168, 1181–1182
 - as data members of classes, 85
 - defining, 188
 - delegates for, 185
 - described, 85
 - focus, gaining and losing for controls, 761
 - generating, 188–192
 - keyboard events, 760–761
 - messages wrapped in, 185
 - methods for handling, 186–188
 - mouse events, 760
 - names of event handlers, 187
 - .NET Remoting and, 1056–1062
 - receiver's view of, 185–188
 - senders for, 185–186, 188
 - threading workaround using, 350
 - user control event handler, 797–798
 - for user controls (ASP.NET), 948
 - for Windows Form user interaction, 760–761
 - Windows Services event logging, 1302–1308
- exceptions. See also error handling**
 - for aborting threads, 354
 - base class exception classes, 328–330
 - catching from predefined classes, 330–340
 - catching user-defined exceptions, 341–343
 - choosing type to throw, 334–335
 - class hierarchy, 329–330
 - debugging and, 406–407
 - handling different exceptions in different places, 340
 - modifying the type of, 340
 - multiple catch blocks for, 331, 332–336
 - nested throw statements for, 332
 - in .NET, 16–17
 - not handled, outcome of, 338

- SimpleExceptions class, 333–336
- throwing, defined, 331
- throwing from catch and finally blocks, 339
- throwing user-defined exceptions, 343–346
- user-defined exception classes, 340–348
- ExecuteNonQuery() **method, 591, 932–933**
- ExecuteReader() **method, 591–592**
- ExecuteScalar() **method, 592–593**
- ExecuteXmlReader() **method, 593–594**
- Exists() **method (MessageQueue class), 1104–1105**
- explicit type conversions, 142–145, 158. See also user-defined casts**
- exposing Web services, 987–990**
- extern **modifier, 125**
- extracting characters from strings, 224**

F

fat-client applications, 22, 24

FieldName **attribute example**

- AttributeUsage attribute, 307–309
- code listing, 307
- specifying optional parameters, 309–310
- specifying required parameters, 309

fields (member variables). See also properties

- accessing instance fields outside the class, 38
- accessing static fields outside the object, 37–38
- casing of, 76–77
- as data members of classes, 85
- defined, 85
- marking unsafe, 206
- pointers to, 212–214
- public, in structs, 102
- readonly fields, 99–101
- scope, 35
- scope clashes for local variables and, 37–38
- usage conventions, 81

FIFO (First In Last Out) structure

- Queue class, 256
- Queue<T> generic class, 283

file associations, setting with ClickOnce, 571

File class

- Copy() method, 1197
- Delete() method, 1197
- described, 1188
- FileInfo class versus, 1189
- Move() method, 1197

file management. See file system operations

File System Editor (ClickOnce), 570**file system operations. See also specific classes**

- buffered streams, 1207
- classes for, 1188
- file browser example, 1192–1197
- file security, 1187, 1222–1226
- FilePropertiesAndMovement application, 1197–1201
- moving and deleting folders, 1197
- moving, copying, and deleting files, 1197–1201
- namespace for, 1187
- reading and writing to binary files, 1208–1213
- reading and writing to text files, 1213–1219
- reading from files, 1201, 1202–1204
- streams, 1205–1207
- writing to files, 1201, 1204–1205

File Types Editor (ClickOnce), 571**FileInfo class**

- CopyTo() method, 1197
- Delete() method, 1197
- described, 1188
- Exists property, 1189–1190
- File class versus, 1189
- instantiating, 1189
- methods, 1191–1192
- MoveTo() method, 1197
- properties, 1190, 1191
- streams versus, 1206

FileProperties application

- ClearAllFields() method, 1194
- currentFolderPath member field, 1194
- DisplayFileInfo() method, 1194–1195, 1196
- DisplayFolderList() method, 1195, 1196, 1197
- event handlers, 1195–1197
- events, 1194
- overview, 1192–1193
- using statements, 1193–1194

FilePropertiesAndMovement application

- ClearAllFields() method, 1201
- controls, 1198
- DisableMoveFeatures utility function, 1201
- DisplayFolderList() method, 1200–1201
- enabling and disabling controls, 1200
- event handlers, 1199–1200
- overview, 1197–1198

FileStream class

- BinaryFileReader application, 1210–1213
- closing streams, 1209
- constructors, 1208–1209

- described, 1207
- enumerations, 1208
- information required for instantiating, 1208
- reading data, 1209
- StreamReader class with, 1215
- StreamWriter class with, 1216
- writing data, 1209–1210

FileSystemInfo class, 1188**Finalize() method**

- for compiled destructors, 199, 200
- described, 107, 108

finalizers (destructors)

- in C++ versus C#, 200
- defined, 86
- as Dispose() method backup, 202–203
- overview, 199–200
- performance and, 200
- syntax, 200

finally blocks

- defined, 330
- error trapping process and, 330
- MortimerColdCall example, 343
- omitting, 331
- for releasing database resources, 586, 587–588
- in SimpleExceptions class, 334
- syntax, 331
- throwing exceptions from, 339

Find() method (List<T> generic class), 280**FindAll() method (List<T> generic class), 280****finding message queues, 1103–1104****First In Last Out (FIFO) structure**

- Queue class, 256
- Queue<T> generic class, 283

fixed keyword for pointers, 213–214**float type**

- converting integers to, 159–162
- overview, 42

floating-point types, 42. See also specific types**flow control. See also specific statements**

- conditional statements, 47–51
- jump statements, 54–55
- loops, 51–54

FlowLayoutPanel control (Windows Forms), 774–775**focus, gaining and losing for controls, 761****folders or directories. See also file system operations**

- configuring for assemblies, 460–462
- defined, 1188
- securing with ASP.NET, 971–973

FolderTree custom control (Windows Forms)

FolderTree **custom control (Windows Forms)**

- adding an image for Toolbox display, 794
- adding properties for handling files and folders, 791–792
- deriving from `TreeView` control, 789
- `FileNode` class for loading files, 789–790
- `FolderNode` class for loading folders, 789, 790–791
- further enhancements possible for, 794
- initializing file system loading, 792–794
- testing the control, 794

Font **class**

- Height property, 879–880
- overview, 877

FontFamily **class, 877, 879**

fonts and font families

- classes for, 877
- defined, 876
- enumerating font families, 878–880
- measuring size of text, 876
- selecting fonts, 877
- terminology, 876–877

for **statements**

- overview, 51–53
- scope of local variables declared in, 36–37

ForEach() **method (List<T> generic class), 281**

foreach **statements**

- iterating through collections with, 54, 246
- overview, 54
- for reading `Stack` values, 254
- as syntactical shortcuts, 248

Form **class**

- application example using, 752
- closing forms, 782
- controlling appearance of forms, 784–786
- `FormBorderStyle` property, 785
- `Icon` property, 785
- inheritance from `Control` class, 780
- initialization, 781
- instantiating, 781
- `Invalidate()` method, 884–886
- methods for keyboard events, 891
- methods for mouse events, 890
- properties relating to form startup, 782
- setting colors for forms, 785
- `Show()` method, 782
- `ShowDialog()` method, 782–784
- `ShowInTaskbar` property, 782
- system menu and, 784–785
- user interaction with, 782–784
- visibility of forms and, 781, 782

Format() **method (String class), 231**

formatter provider (.NET Remoting), 1012, 1047–1048

formatters

- Message Queuing, 1107–1108
- .NET Remoting, 1012, 1025

formatting expressions

- format specifiers, 230
- `FormatVector` example, 233–235
- how strings are formatted, 231–233
- interfaces for, 223
- need for, 229

FormatVector **struct, 233–235**

forms. **See Web Forms; Windows Forms**

FormView **control, 936, 937–938**

FromFile() **method (Image class), 871**

FTP, as high-level protocol, 1262

function pointers, 171. **See also delegates**

functions. **See also constructors; finalizers**

(destructors); methods; properties

- abstract, 117
- association with class required for, 86
- calling base versions, 116–117
- defining for structs, 101–102
- function members of classes, 85–86
- methods versus, 86
- overview, 86
- standard function calls, 78–80

future of distributed programming. **See WCF; Web services specifications**

G

GAC. **See global assembly cache**

gacutil (Global Assembly Cache Utility), 441, 446, 449

garbage collection

- closing forms and, 782
- destructors and, 199
- efficiency of, 193–194
- `fixed` keyword for pointers and, 213–214
- forcing, 199
- managed heap and, 13, 198–199
- not deterministic, 14
- overview, 198–199
- strong data typing in IL and, 13–14
- `System.GC` class for, 199

GDI (graphical device interface), 842. **See also GDI+ GDI+.** **See also graphics**

- color representation in, 863
- described, 841
- device contexts (DCs), 843–844

- displaying images using, 870–873
- GDI versus, 842
- Graphics class and, 844
- huge number of features provided by, 842
- namespaces for, 843
- responding to user input, 890–893
- Generate() **method** (ResourceWriter **class**), **528**
- GenericCustomer **class**, **119–124**
- generics**
 - ArraySegment<T> struct, 303–304
 - binary code reuse by, 273–274
 - C++ templates compared to, 272
 - code bloat and, 274
 - collection classes and their functionality, 276–277
 - creating custom generic classes, 293–296
 - delegates, 299–301
 - displaying collections with DataGridView control, 811–813
 - EventHandler<TEventArgs> delegate, 302–303
 - interfaces and their functionality, 275–276
 - introduction of, 269, 271
 - LinkedList<T> class, 287–292
 - List<T> class, 272–273, 278–283
 - methods, 297–299
 - namespace for, 270
 - naming guidelines, 274
 - .NET base classes for, 277–278
 - Nullable<T> struct, 301–302
 - performance advantages of, 272–273
 - Queue<T> class, 283–286
 - strong typing provided by, 269
 - type safety of, 273
- get **accessor of properties**, **93, 94**
- GetAssemblyFullName() **method**, **452**
- GetAttribute() **method** (XmlReader **class**), **668**
- GetChannel() **method** (ChannelServices **class**), **1026**
- GetCustomAttributes() **method** (Assembly **class**), **320, 323**
- GetDocument() **method** (DocumentManager **class**), **284, 294**
- GetEnumerator() **method** (IEnumerable **interface**), **246–247**
- GetFreeDate() **method**, **929–930, 933**
- GetHashCode() **method**, **107, 263–264**
- GetKeyList() **method** (SortedList **class**), **258**
- GetMembers() **method**, **319**
- GetMethods() **method** (Type **class**), **323**
- GetObject() **method** (Activator **class**), **1027, 1028, 1029, 1030**
- GetRandomQuoteOfTheDay() **method** (QuoteServer **class library**), **1277**
- GetRange() **method** (ArrayList **class**), **253**
- GetRowType() **method** (CustomerTable **class**), **836**
- GetType() **method** (Type **class**), **107, 108, 314**
- GetTypes() **method** (Assembly **class**), **319–320, 322**
- GetValue() **method** (RegistryKey **class**), **1231**
- GetValueList() **method** (SortedList **class**), **258**
- global assembly cache**
 - adding publisher policy assembly to, 457
 - Assembly Cache Viewer for, 439–440
 - ClickOnce limitations and, 567
 - COM registration using, 1178
 - defined, 19, 413, 438
 - gacutil (Global Assembly Cache Utility), 441, 446, 449
 - Ngen.exe native image generator, 438–439
- Global Assembly Cache Utility (gacutil)**, **441, 446, 449**
- globalization. See also localization**
 - creating custom cultures, 549–550
 - CultureInfo class for, 515
 - CurrentCulture and CurrentUICulture properties for, 516–517, 518, 519
 - date formatting, 519–520
 - defined, 513
 - localization versus, 513
 - namespace for, 513, 514
 - number formatting, 517–519
 - RegionInfo class for, 515
 - sorting and, 524–526
 - specific, neutral, and invariant cultures and, 515–516
 - Unicode issues, 514–515
 - Windows Forms culture demo application, 520–524
- globally unique identifier (GUID) in Active Directory**, **728**
- goto **statement**, **50, 55**
- graphical device interface (GDI)**, **842. See also GDI+ graphics. See also CapsEditor example; GDI+**
 - clipping region and, 848–850
 - colors, 863–866
 - debugging and, 855–856
 - device coordinates for, 863
 - display modes, colors and, 865
 - displaying images, 870–873
 - drawing scrollable windows, 856–862
 - drawing shapes and lines, 868–870
 - drawing shapes, application for, 844–847
 - drawing text, 874–880
 - editing a text document, 880–893
 - enumerating font families, 878–880
 - fonts and font families, 876–880
 - hidden window parts and, 846–847

graphics (continued)

- issues when manipulating images, 873–874
- measuring coordinates and areas, 850–855
- page coordinates for, 863
- painting shapes, 847–848
- pens and brushes for, 866–868
- printing, 893–899
- simple text example, 874–876
- world coordinates for, 863

Graphics class

- `DrawImage()` method, 874
- `DrawImageUnscaled()` method, 873
- `DrawString()` method, 874, 875–876
- GDI+ and, 844
- `MeasureString()` method, 874, 893
- methods, 868–869

GregorianCalendar class, 522–523

grouping characters with regular expressions, 240–243

groups of objects. See also generics; specific functionalities

- array lists, 250–253
- collections, 246–250
- dictionaries, 259–269
- .NET functionality supporting, 245–246
- queues, 256–257
- `SortedList` collections, 257–259
- stacks, 253–256

GUID (globally unique identifier) in Active Directory, 728

H

hash tables. See dictionaries

Hashtable class, 261–262

heap. See managed heap

HelpProvider component (Windows Forms), 769

hiding methods, 115–116

hosting with WCF, 1143–1144

HTML

- in ASP.NET code model, 909
- displaying output as HTML page, 1245–1258
- naming tags for user controls (ASP.NET), 946
- `<%@ Page %>` tag, 908
- server controls (ASP.NET), 910

HTTP performance issues, 1133

I

IANA (Internet Assigned Number Authority), 1263

IBankAccount interface, 127–130

IChannel interface, 1022

ICollection interface, 247

_ICompletedEvents client interface (COM), 1166–1167

IDE for .NET. See Visual Studio 2005

identifiers, 74–75

Identity Permissions, 472–473

IDispatch interface (COM), 1163, 1164

IDisposable interface

- deriving classes from, 126–127
- destructors as backup for, 202–203
- `Dispose()` method, 201–202
- implementing, 202–204
- overview, 201–202

IEnumerable interface, 246–247, 1109–1110

#if preprocessor directive, 71–72

if statements

- overview, 47–49
- ternary operator, 136–137

IFormattable interface, 231–232

IIS (Internet Information Server), ASP.NET and, 904

IL or MSIL (Microsoft Intermediate Language)

- application domains and, 14–16, 413
- attributes, 17
- case sensitivity, 12
- CLS and language interoperability, 12–13
- compilation steps in .NET and, 4
- compiling to native code, 439
- CTS and data types, 10–12
- defined, 3
- error handling with exceptions, 16–17
- garbage collection and, 13–14
- important features, 8
- interfaces supported by, 8–9
- language interoperability with, 5–7
- as managed code, 4
- object orientation supported by, 8–9
- performance improvement with, 5
- platform independence with, 4–5
- security and strong typing, 14
- strong data typing, 9–16
- value types versus reference types, 9

ildasm utility

- checking `.mresource` attribute, 528
- symbols used in, 420–421
- viewing assemblies with, 420

IList interface, 811

IListSource interface, 811

Image class, 871

ImageList component (Windows Forms), 769

images. See graphics; resources

IMath interface (COM), 1162, 1174, 1177–1178

`IMessageSink` **interface**, 1031

implementation inheritance

- abstract classes and functions, 117
- calling base versions of functions, 116–117
- hiding methods, 115–116
- not supported by structs, 112
- overview, 111–112
- sealed classes and methods, 117–118
- syntax, 113
- virtual methods, 114–115

implicit permission, 484–485

implicit type conversions, 141–142

indexers

- for array lists, 251
- defined, 86

indexes to XML types, 656–657

indirection operator for pointers, 207

inheritance. *See also derived classes; specific kinds*

- abstract classes and functions, 117
- calling base versions of functions, 116–117
- constructors of derived classes, 118–124
- derived interfaces, 131–132
- hiding methods, 115–116
- implementation inheritance, 111–112, 113–124
- interface inheritance, 112, 126–132
- modifiers, 124–125
- multiple, 112
- from `Object` class, 106
- sealed classes and methods, 117–118
- structs and, 103, 112–113
- virtual methods, 114–115

`Init()` **method**, 644–645

initialization

- of arrays with specific dimensions, 58
- `CapsEditor` example, 883
- of constructors, 99
- of `FolderTree` custom control (Windows Forms), 792–794
- of `Form` class, 781
- of structs, 103
- of user controls (Windows Forms), 798
- of variables, 34–35

`InitializeComponent()` **method**

- `Localizable` property for, 536
- setting minimum document size with, 858–859
- shape-drawing application, 844–845

inserting records with ADO.NET, 591, 596–597

installallutil.exe utility, 1291–1292

`InstalledFontCollection` **class**, 877

installing. *See also Windows Installer projects*

- assemblies, 412
- message queues, 1122
- shared assemblies, 446, 451
- Windows Services, 1287–1292

instantiating

- `ArrayList` class, 250
- classes and structs, 84
- COM components, 1165
- delegates, 173–174, 176–177
- derived classes, 120
- `DirectoryInfo` class, 1189
- enumerations, 57
- `FileInfo` class, 1189
- `Form` class, 781
- reference objects, 35
- `SortedList` class, 258

int type, 41, 42

integer types. *See also specific types*

- casting pointers to, 208–209
- converting between `string` type and, 144–145
- converting to floats, 159–162
- enumerations, 55–57
- overview, 41–42

interfaces

- for channels (.NET Remoting), 1022
- COM, 1153–1155
- for database connections, 825
- defining and implementing, 127–130
- delegates and, 181
- derived, 131–132
- for formatters (.NET Remoting), 1025
- for formatting expressions, 223
- for generic base classes, 278
- for generics, 275–276
- IL support for, 8–9
- inheritance, 112–113
- for `List<T>` generic class, 278
- for message sinks, 1031
- .NET Remoting, 1052
- .NET versus COM, 8, 126
- overview, 126–127
- partial, 104
- references, 130

Intermediate Language. *See IL or MSIL (Microsoft Intermediate Language)*

internal modifier, 124, 125

internationalization. *See globalization*

Internet access

- displaying output as HTML page, 1245–1258
- displaying the code of a requested page, 1256–1257
- DNS servers and, 1259
- DnsLookup application, 1261–1262
- downloading files, 1240
- giving applications IE-type features, 1248–1253
- IP addresses and, 1259
- launching Internet Explorer instances, 1248
- lower-level classes, 1263–1268
- lower-level protocols, 1262–1263
- .NET classes for IP addresses, 1260–1261
- printing using `WebBrowser` control, 1255–1256
- showing documents using `WebBrowser` control, 1254–1255
- simple Web browsing from applications, 1246–1247
- Socket class, 1263, 1268
- starting an Internet Explorer process programmatically, 1245–1246
- `System.Net.Sockets` namespace classes, 1263
- TCP classes, 1263
- TCP versus UDP, 1266
- `TcpReceive` application, 1264–1266, 1268
- `TcpSend` application, 1263–1264
- `UdpClient` class, 1266–1267
- uploading files, 1242
- Uri class, 1258–1259
- `UriBuilder` class, 1258, 1259
- utility classes, 1258–1262
- `WebClient` class, 1240–1242
- `WebRequest` and `WebResponse` hierarchy, 1257–1258
- `WebRequest` class, 1242–1245
- `WebResponse` class, 1242–1245

Internet Assigned Number Authority (IANA), 1263

Internet Explorer

- giving applications IE-type features, 1248–1253
- launching instances, 1248
- running Windows Forms controls in, 1183
- starting a process programmatically, 1245–1246
- threading by, 349–350

Internet Information Server (IIS), ASP.NET and, 904

Internet resources

- COM interoperability sample code, 1151
- event-booking application example, 920
- Internet Assigned Number Authority (IANA), 1263
- Microsoft Application Center Server information, 1067
- Mono project, 5
- `PrintingCapsEdit` project, 895
- SharePoint Web sites information, 920

- Web services specifications, 1127
- WROX Web site, 895, 920, 1151

INullable interface, 637

Invalidate() method (Form class), 884–886

invariant cultures, 516

Invoke() method, 1117

invoking methods

- generic methods, 297
- overview, 87–89

IP addresses

- Dns class, 1260–1261
- DNS servers and, 1259
- DnsLookup application, 1261–1262
- IPAddress class, 1260
- IPHostEntry class, 1260

IPAddress class, 1260

IPHostEntry class, 1260

is operator, 138

isolation levels for ADO.NET transactions, 588–589

isolation of Enterprise Services transactions, 1078

IsStyleAvailable() method (FontFamily class), 879

IsTransparentProxy() method (RemotingServices class), 1029

ITransferBankAccount interface, 131–132

IUnknown interface (COM), 1163, 1164

IWelcome interface (COM), 1160–1161, 1162, 1174, 1177–1178

J

J++, reading in projects to Visual Studio 2005, 387

JIT (Just-In-Time) compilation, 5

Join() method (Thread class), 354

JScript .NET, interoperability with .NET, 7

jump statements. See also specific statements

- anonymous methods and, 177
- overview, 54–55

K

key generation with SQL Server, 627–629

keyboard events

- methods for, 891
- overview, 760–761

keys, database

- generation with SQL Server, 627–629
- setting a foreign key, 610–611
- setting a primary key, 610

keys, registry, 1228–1229

keywords. See also specific keywords

- names and, 74, 78–80
- reserved keywords, 74
- using as identifiers, 74

L**Label control**

- Web Forms, 920
- Windows Forms, 769–770

LabelIdMapping class, 1117–1118**LandLineSpyFoundException, 341, 346****language interoperability with .NET. See also CLS (Common Language Specification)**

- COM and COM+, 7, 8–9, 425
- CTS and, 425
- defined, 5, 9
- scripting languages, 7
- strong data typing's importance for, 10–13
- Visual Basic 2005, 5–6
- Visual C++ 2005, 6–7
- Visual J# 2005, 7
- XML namespace and, 661

Last In First Out (LIFO) structure of Stack class, 253–254**LastModifiedAttribute class, 311–312****late binding (COM), 1155****Launch Conditions Editor (ClickOnce), 575–576****lifetime management (.NET Remoting)**

- changing default lease configurations, 1038–1039
- classes for, 1037
- getting lease information, 1037–1038
- lease renewals, 1036–1037
- leasing configuration values, 1037
- overview, 1036
- services in configuration files, 1046–1047

LIFO (Last In First Out) structure of Stack class, 253–254**line breaks in string literals, 46****#line preprocessor directive, 73****LineIndexToPageCoordinates() method, 890****LineIndexToWorldCoordinates() method, 889****lines**

- ScrollMoreShapes example, 869–870
- System.Drawing.Graphics methods for, 868–869

LinkedList<T> generic class

- advantages and disadvantages of, 287
- doubly linked list diagram, 287
- example using, 287, 289–292

LinkLabel control (Windows Forms), 1248**ListBox control (Windows Forms), 765–767****ListControl class (Windows Forms)**

- CheckedListBox control derived from, 765
- ComboBox control derived from, 765, 767
- ListBox control derived from, 765–767

ListenerThread() thread function (QuoteServer class), 1278**List<T> generic class**

- Add() method, 279
- boxing and unboxing avoided by, 272–273
- example using Racer class as elements for, 278–279
- finding elements, 280
- interfaces implemented by, 278
- with LinkedList<T> class, 287, 289
- performing an action with every element, 281
- sorting elements, 281–282
- type conversion using, 282–283

ListView control (Windows Forms)

- adding columns for details view, 771–772
- adding items to ComboBox (cbView), 771
- Alignment property, 772
- CheckBoxes property, 772
- CountryList example, 770–771
- overview, 770

literals, 44, 46**Load() method (Assembly class), 319****LoadFile() method**

- CapsEditor example, 884
- PrintingCapsEdit project, 895

LoadFrom() method (Assembly class), 319**local variables**

- scope, 35–36
- scope clashes for fields and, 37–38
- scope clashes for variables with same name, 36–37
- unsafe, 206

Localizable property of Windows Forms, 536, 537**localization. See also globalization; resources**

- automatic fallback for resources, 542
- changing the culture programmatically, 539–541
- creating custom cultures, 549–550
- CurrentCulture and CurrentUICulture properties for, 539
- custom resource messages for, 541–542
- custom resource reader, 545–549
- defined, 513
- globalization versus, 513
- namespace for, 513, 534
- outsourcing translations, 543–544
- satellite assemblies for languages, 538
- using ASP.NET, 544–545

localization (continued)

using Visual Studio, 534–555
winres.exe (Windows Resource Localization Editor), 543

lock statement

deadlocks, avoiding, 361–363, 364
overusing, avoiding, 361
race conditions, avoiding, 363–364
using, 360–361

logging Windows Services events. See event logging for services

login

maintaining using SOAP headers, 1001–1007
system implementation, 969–970
Web server controls (ASP.NET), 970–971

Login() **method**, 1002–1003, 1005

long **type**, 41, 42

LookUpWhatsNew **assembly**, 310, 321–324

loops. See also specific statements

break statement in nested loops, 55
overview, 51
scope of local variables declared in, 36–37

M

machine configuration files for assemblies, 450

machine.config **file**, 1041, 1046

mage.exe **ClickOnce tool**, 568

mageUI.exe **ClickOnce tool**, 568

Main() **method**

for Finnish sorting example, 525
first thread started at, 350
MortimerColdCall example, 341–343
overview, 32–33, 61
passing arguments to, 63
PriorityDocumentManager class, 292
QuoteService example, 1284–1285
single entry point provided by, 349
TypeView assembly example, 318
using multiple Main() methods, 61–62
Windows Form application, 752–753
for Windows Services, 1273–1274

making. See building; compiling

managed code. See also IL or MSIL (Microsoft Intermediate Language)

language interoperability with, 5–7
performance improvement with, 5
platform independence with, 4–5

managed heap

garbage collector and, 13, 198–199
reference types stored in, 9, 39, 196–198

stack and, 196–198

structs stored in, 9

value types declared within reference types stored in, 9

manifests of assemblies, 418

manufactured tables and rows

dispatching methods for, 837–838
getting the selected row, 839–840
required methods for, 835–837
using an attribute, 837

maps. See dictionaries

marshal-by-reference classes, 1031–1032

MarshalByRefObject **class**, 1188

marshal-by-value classes, 1031

marshaling

COM, 1159

.NET Remoting, 1032–1034

MaskedTextBox **control (Windows Forms)**, 773, 774

master pages (ASP.NET)

Content controls and, 958

.master file extension for, 956

modifying .aspx pages to use, 957

in PCSDemoSite application, 958–960

requirements for .aspx pages using, 958

selecting when adding .aspx pages to Web site,
957–958

standard .aspx pages compared to, 956–957

uses for, 956

MathOperations **class**

multicast delegate with, 183–184

simple delegate with, 177–179

MathTest **class**, 87–89

MDI (Multiple Document Interface), 786–787

MeasureString() **method (Graphics class)**,
874, 893

member variables. See fields

MemberwiseClone() **method**, 107, 108

memory management. See also managed heap; stack

in C++, 13

in COM, 13, 1153

destructors for, 199–200, 202–203

freeing unmanaged resources, 199–204

garbage collection, 13–14, 193–194, 198–199

IDisposable interface for, 201–204

reference types and, 196–198

value types and, 194–195

virtual addressing and virtual memory, 194

Windows platform methods for, 13

memory (RAM)

freeing in COM, 1153

requirements for application deployment, 553

- MenuCommand **proxy class**, 838
- menus (Windows Forms)**
 - ContextMenuStrip class for, 779
 - MenuStrip control for, 779
 - ToolStrip control for, 776
 - ToolStripManager class for, 780
 - ToolStripMenuItem class for, 779–780
- MenuStrip **control (Windows Forms)**, 779
- Merge() **method**, 644–645
- Message Queuing**
 - acknowledgement messages, 1099
 - acknowledgement queues, 1120
 - administration queues, 1100
 - administrative tools, 1101–1102
 - architecture, 1099–1101
 - asynchronous operation of, 1095–1096
 - asynchronous read, 1110–1111
 - components available for, 1098
 - connected and disconnected programming, 1096
 - course order application, 1111–1119
 - creating message queues programmatically, 1103
 - creating message queues with Computer Management tool, 1101
 - dead-letter queues, 1100, 1105
 - enumerating messages, 1109–1110
 - express delivery mode, 1099
 - features, 1098
 - finding a queue, 1103–1104
 - installing message queues, 1122
 - journal queues, 1100, 1105
 - message formatter, 1107–1108
 - message queue properties, 1101–1102
 - message queues overview, 1100–1101
 - messages overview, 1099–1100
 - opening known queues by format name, 1105–1106
 - opening known queues by path name, 1104–1105
 - overview, 1095–1098
 - priorities for messages, 1099
 - private queues, 1100, 1105
 - public queues, 1100, 1105
 - receiving messages, 1109–1111
 - receiving results, 1120–1121
 - recoverable delivery mode, 1099–1100
 - report messages, 1099
 - report queues, 1100
 - response messages, 1099
 - response queues, 1100, 1120–1121
 - sending messages, 1106–1109
 - service for, 1098–1099
 - system queues, 1100
 - transactional messages, 1100
 - transactional queues, 1121–1122
 - uses for, 1096–1097
 - WCF and, 1147
- message sinks (.NET Remoting)**
 - defined, 1012, 1015
 - overview, 1030–1031
- MessageArrived **handler method (MessageQueue class)**, 1110–1111
- MessageEnumerator **class**, 1110
- MessageQueue **class**
 - BeginReceive() method, 1110
 - Create() method, 1103, 1121
 - EndReceive() method, 1111
 - Exists() method, 1104–1105
 - IEnumerable interface implemented by, 1109–1110
 - MessageArrived handler method, 1110–1111
 - Receive() method, 1109, 1110
 - Send() method, 1106, 1108
- messages. See also events; Message Queuing**
 - defined, 185
 - .NET Remoting, 1012, 1030
 - wrapped in events, 185
- metadata**
 - in assemblies, 18
 - COM, 1152
 - custom attributes emitted as, 306
- methods. See also delegates; properties; specific methods**
 - abstract, 117
 - anonymous, delegates with, 176–185
 - Application class, 753
 - binding (COM), 1155
 - calling base versions, 116–117
 - calling in Web services with SOAP, 984–985
 - for combined characters, 515
 - declaring, 86–87
 - defined, 86
 - DirectoryInfo class, 1191–1192
 - for drawing shapes and lines, 868–869
 - event handlers, 186–188
 - FileInfo class, 1191–1192
 - format of definitions, 33
 - functions versus, 86
 - generic, 297–299
 - of generic classes, 276–277
 - of generic interfaces, 275–276
 - hiding, 115–116
 - initialization required for variables local to, 35
 - invoking, 87–89

methods (continued)

- for keyboard events, 891
- making accessible through Web services, 988
- for manufactured tables and rows, 835–838
- marking unsafe, 205
- for mouse events, 890
- multicast delegates and, 183–185
- non-compliant with CLS, 426
- Object class, 106–109
- of object type, 45
- overloading, 92
- overriding virtual methods, 114–115
- passing parameters to, 89–91
- public versus static, 33
- Queue class, 257
- for reading from console, 65
- Registry class, 1231–1232
- return statement for exiting, 55
- return type required for, 87
- sealed, 117–118
- ServiceController class, 1300
- ServiceComponent class, 1072
- SortedList class, 258–259
- Stack class, 255–256
- String class, 224–225
- StringBuilder class, 228–229
- System.Type class, 315–316
- usage conventions, 80
- virtual, 114–115
- in Visual Studio 2005 Object Browser window, 397–398
- for writing to console, 65
- XPathNavigator class, 679–680

Microsoft Application Center Server, 1067

Microsoft Intermediate Language. See **IL or MSIL**

Microsoft Management Console. See **MMC**

Microsoft SQL Server Database Engine (MSDE),
connecting to local database, 823

`Microsoft.SqlServer.Server` namespace, 635

minus sign (-)

- as decrement operator (--), 135–136
- for removing method calls from multicast delegates, 183

MMC (Microsoft Management Console)

- Active Directory Domains and Trusts snap-in, 721
- Active Directory Sites and Services snap-in, 721
- Active Directory Users and Computers snap-in, 721–722
- Computer Management snap-in, 1101–1102
- Services snap-in, 1293

mobile code, 463

modifiers. See also *specific modifiers*

- interface members and, 126
- visibility modifiers, 124–125

modules

- assemblies versus, 421–422
- creating, 421
- defined, 421
- purpose of, 423

Mono project, 5

MortimerColdCall **example**

- catching user-defined exceptions, 341–343
- ColdCallFileFormatException, 341, 346–347
- ColdCallFileFormatException handler, 343
- ColdCallFileReader class, 341, 342, 343–346
- defining exception classes, 346–348
- FileNotFoundException exception, 341, 342–343
- LandLineSpyFoundException, 341, 346
- Main() method, 341–343
- overview, 340–341
- testing, 347–348
- throwing user-defined exceptions, 343–346
- UnexpectedException, 341, 347

MortimerPhonesEmployees **dictionary example,**
264–269

mouse events

- CapsEditor example, 890–893
- methods for, 890
- overview, 760

Move() **method (File and Directory classes), 1197**

MoveTo() **method (FileInfo and DirectoryInfo**
classes), 1197

moving

- files, 1197–1201
- folders, 1197

.mresource attribute, 528, 538

MSDE (Microsoft SQL Server Database Engine),
connecting to local database, 823

MSIL. See **IL or MSIL (Microsoft Intermediate**
Language)

MSXML parser

- System.Xml namespace advantages over, 664
- using in .NET, 661–663

MTA (multi-threaded apartment) in COM, 1157, 1166

multicast delegates, 183–185

Multiple Document Interface (MDI), 786–787

multiple inheritance, 112

multitasking. See **threading**

`MyFirstCSharpClass` class, 30–33

N**names**

- Active Directory distinguish names (DNs), 727–728
- Active Directory user names, 728–729
- ADO.NET naming conventions, 629–630
- aliases, 60–61, 585
- camel casing, 76–77
- code group labels, 469
- for colors, 864–865
- distributing code using strong names, 497–499
- of event handlers, 187
- generics naming guidelines, 274
- of HTML tags for user controls (ASP.NET), 946
- IL case sensitivity, 12
- of interfaces, 128
- of namespaces, 59, 77
- .NET guidelines and philosophy, 75–76
- Pascal casing, 76, 77
- of properties, 93
- reserved keywords, 74
- rules for identifiers, 73–75
- strong, for shared assemblies, 441–443, 444–445
- styles for, 77
- variable prefixes, 75
- Visual Basic .NET not case-sensitive, 77

namespace **keyword**, 32

namespaces

- for ACLs, 1222
- for Active Directory, 723
- for ADO.NET, 580
- aliases, 60–61
- assemblies and, 419
- for classes, 32, 58–59
- for COM objects, 1151
- defined, 21
- for DSML, 724, 745
- for event senders, 188
- for file system operations, 1187
- for GDI+, 843
- for generics, 270
- for globalization, 513, 514
- for localization, 513, 534
- names for, 59, 77
- nesting, 21, 59
- for .NET Remoting, 1009
- Pascal casing for, 76
- for performance monitoring, 1309
- for power events support, 1314
- for registry operations, 1187

- for serialization base classes and interfaces, 1187
- for serializing objects in XML, 701–702
- shared assemblies and, 442
- soap namespace, 984–985
- for SQL Server 2005, 635
- for Stack class, 254
- System.Xml.XPath, 678–684
- System.Xml.Xsl, 684–689
- for threading, 351, 366
- types defined in, 21, 59
- using statement for, 32, 59–61
- for Windows Forms, 751
- for Windows Services, 1275
- for XML, 659, 660–661

navigation Web server controls (ASP.NET), 960–963**nesting**

- break statement in nested loops, 55
- for loops, 52
- namespaces, 21, 59
- partials, 105–106
- throw statements in try blocks, 332
- try blocks, 338–340
- types, 125

.NET data access. See ADO.NET; viewing .NET data; Visual Studio .NET and data access

.NET Framework. See also ASP.NET; CLR (Common Language Runtime); IL or MSIL (Microsoft Intermediate Language)

- application domains, 14–16, 413–416
- assemblies, 17–19
- attributes, 17
- base classes, 19–20, 277–278
- benefits of C# and, 25
- compilation steps in, 4
- C#'s relationship to, 3, 4
- C#'s role in enterprise architecture, 24–26
- delegates defined in, 188
- error handling with exceptions, 16–17
- garbage collector, 13–14
- installation base lacking for, 24
- namespaces, 21
- open source implementation, 5
- operating systems supported, 553
- security advantages of, 14
- server platforms supported, 553
- similar concepts in COM, 1152
- usage conventions, 75–81
- using a COM component from a .NET client, 1159–1171

.NET Framework (continued)

- using a .NET component from a COM client, 1171–1183
- Windows Controls and, 24
- Windows Forms and, 24
- Windows Services and, 24
- XML standards supported, 660
- .NET Framework Configuration tool, 1048–1050**
- .NET IDE. See Visual Studio 2005**
- .NET Remoting**
 - activator for client, 1012
 - application types and, 1010
 - asynchronous, 1053–1054
 - call contexts, 1062–1064
 - channels, 1012, 1020–1025, 1041–1042, 1046
 - ChannelServices class, 1012, 1025–1026
 - client-side process, 1012–1014
 - CLR Object Remoting, 1011
 - configuration files, 1039–1050, 1059, 1061
 - contexts, 1014–1016
 - defined, 1010
 - delegates with, 1053–1054
 - directional attributes, 1035–1036
 - events and, 1056–1062
 - formatter provider, 1012, 1047–1048
 - formatters, 1012, 1025
 - hosting servers in ASP.NET, 1050–1051
 - interfaces, 1052
 - key architecture elements, 1012, 1013
 - lifetime management, 1036–1039, 1046–1047
 - major classes for example, 1016–1017
 - marshal-by-reference classes, 1031–1032
 - marshal-by-value classes, 1031
 - marshaling, 1032–1034
 - message sinks, 1012, 1015, 1030–1031
 - messages, 1012, 1030
 - namespaces for, 1009
 - not-remotable classes, 1032
 - object activation, 1027–1030
 - OneWay attribute, 1054
 - overview, 1011–1014
 - passing objects in remote methods, 1031–1036
 - protocols, 1010–1011
 - proxies, 1012
 - remote objects, 1012, 1017–1018, 1056–1058
 - RemotingConfiguration class, 1012, 1026
 - RemotingServices class, 1026
 - security, 1035, 1054–1055, 1126
 - serialized objects and security, 1035
 - server for client-activated objects, 1026–1027

- server for well-known objects, 1026
- server-side process, 1014
- simple client example, 1019–1020
- simple server example, 1018–1019
- Soapsuds utility, 1052–1053
- unbound classes, 1031
- uses for, 1011
- WCF and, 1145–1146
- XML Web services with, 1010

- .NET security. See security**
- net.exe utility, 1293–1294**
- neutral cultures, 515–516**

new keyword

- declaring class and struct instances, 84
- hiding methods, 116
- as modifier for function members, 125
- for object activation (.NET Remoting), 1027
- reference types and, 196
- structs versus classes and, 103

- `NewRowFromBuilder()` **method (CustomerTable class), 836–837**

- Ngen.exe native image generator, 438–439**

- No Touch Deployment (NTD), 566–567**

- Northwind database example (ADO.NET). See ADO.NET**
- Northwind database example (Enterprise Services)**

- C# component library for, 1081
- client application, 1089–1090
- entity classes, 1081–1084
- Order class, 1081–1083
- OrderControl component, 1080–1081, 1084–1085
- OrderData component, 1081, 1085–1088
- OrderLine class, 1083–1084
- OrderLineData component, 1081, 1088–1089
- overview, 1080–1081

- NT Services. See Windows Services**

- NTD (No Touch Deployment), 566–567**

null values

- comparing reference types for equality, 147
- null coalescing operator, 139–140
- for reference types, 40

nullable types

- converting, 144
- implicit type conversions and, 142
- `Nullable<T>` generic struct and, 301–302
- operators and, 139
- special syntax for, 302

- `Nullable<T>` **generic struct, 301–302**

- number formatting for globalization, 517–519**

- `NumberFormatInfo` **class, 518**

O**object activation (.NET Remoting)**

- Activator class for, 1027
- for client-activated objects, 1028–1029
- messages, 1030
- new operator for, 1027
- proxy objects and, 1027, 1029
- specifying the application URL, 1027
- for well-known objects, 1027–1028

Object Browser window (Visual Studio 2005), 397–398**Object class**

- all .NET classes derived from, 106
- as default base class, 113
- Equals() method, 107
- Finalize() method, 107, 108
- GetHashCode() method, 107
- GetType() method, 107, 108
- MemberwiseClone() method, 107, 108
- ToString() method, 106, 108–109

object type, 44–45**object-oriented programming (OOP), 30****OLE DB provider with ADO.NET, 597–600****OnAction method, 189–190****OnContinue() method (QuoteService assembly), 1286****OnCustomCommand() method (QuoteService assembly), 1286****OnDisplayButtonClick event handler, 1195–1196****OnDoubleClick() event handler, 892–893****OnListBoxFilesSelected event handler, 1196****OnListBoxFoldersSelected event handler, 1196****OnMouseDown() method, 892****OnOrderSelectionChanged() method, 1118–1119****OnPaint() method**

- CapsEditor example, 887–888
- clipping region and, 848–850
- DisplayImage project, 872
- DisplayText example, 874–875
- EnumFontFamilies example, 879, 880
- painting shapes using, 847–848
- scroll bars and, 859–860, 862
- ScrollMoreShapes example, 870

OnPause() method (QuoteService assembly), 1286**OnPowerEvent() method (ServiceBase class), 1314****OnSelectCulture() method, 522****OnShutdown() method (QuoteService assembly), 1286****OnStart() method (QuoteService assembly), 1285, 1286, 1287****OnSubmitCourseOrder() method, 1114****OnUpButtonClick event handler, 1197****OOP (object-oriented programming), 30****Open() method (ColdCallFileReader class), 343–344****opening**

- database connections with ADO.NET, 583
- message queues, 1104–1106
- projects from previous versions in Visual Studio 2005, 374–376

OpenRead() method (WebClient class), 1240–1241**OpenSubKey() method (RegistryKey class), 1230****OpenWrite() method (WebClient class), 1241****operating systems supported by .NET, 553****operators. See also specific operators**

- as, 138
- assignment versus comparison, 48, 134–135
- checked and unchecked, 137
- comparison, 48, 134–135, 147
- compiler and, 149–150
- defined, 86
- is, 138
- null coalescing, 139–140
- nullable types and, 139
- overloading, 148–157
- pointer member access operator, 212
- for pointers, 207
- precedence, 140
- for removing method calls from multicast delegates, 183
- shortcut assignment operators, 135–136
- sizeof, 138
- table summarizing, 134
- ternary, 136–137
- typeof, 139
- unsafe, 134

Order class, 1081–1083**OrderControl component, 1080–1081, 1084–1085, 1091–1092****OrderData component, 1081, 1085–1088, 1092–1093****OrderLine class, 1083–1084****OrderLineData component, 1081, 1088–1089****/out option for compiler output file specification, 64****out parameters**

- anonymous methods and, 177
- passing to methods, 91

outsourcing translations, 543–544

OverflowException class

OverflowException class, 329

overloading constructors, 95

overloading methods

DrawEllipse() method, 845

DrawRectangle() method, 845

overview, 92

Run() method, 753

Write() method, 1216–1217

overloading operators

arithmetic operators, 148, 151–152, 153–155

comparison operators, 148, 155–157

compiler and, 149–150

for concatenating strings, 224

operators supporting overloading, 157

public and static declaration required for, 152

uses for, 148, 149

Vector struct example, 150–157

override modifier, 125

overriding

Equals method for comparisons, 146–147, 148

Equals method for dictionaries, 264, 266, 267

GetHashCode() method for dictionaries, 263–264, 266–267

OnPaint() method, scrolling windows and, 859–860, 862

Render() method for custom control, 954–955

virtual methods and properties, 114–115

P

page coordinates, 863

PageCoordinatesToLineIndex() **method, 890, 893**

Page_Load() **event handler**

event-booking application, 924

login application using SOAP headers, 1004, 1006

PaintEventArgs **class, 848**

painting shapes. See also OnPaint() method

clipping region and, 848–850

overview, 847–848

Panel **control (Windows Forms), 774**

parameters

camel casing for, 76

constructors with, adding in a hierarchy, 122–124

constructors without, adding in a hierarchy, 120–122

for custom attributes, specifying, 309–310

out parameters, 91

overloading methods and, 92

passing by reference versus by value, 89

passing to Main(), 63

passing to methods, 89–91

ref parameters, 90–91

ParameterTest **class, 89–90**

parentheses [()] for groups, 241–242

Parse() **method, 641**

partial **keyword, 104–106**

Pascal casing, 76, 77

passing methods. See delegates

Path **class, 1188, 1191–1192**

PCSDemoSite **application (ASP.NET)**

downloading, 944

master pages in, 958–960

navigation in, 962–963

security, 972–973

themes in, 975–979

user controls in, 950–952

PCSWebApp1 example

adding server controls, 911–913

code-behind file, 909

creating in Visual Studio, 906–908

PeekMessages() **method, 1116–1117**

Pen **class, 866, 867–868**

perfmon.exe utility, 1312–1313

performance

array lists versus dictionaries, 259

ASP.NET versus ASP 22

boxing and unboxing impacts on, 272

COM issues for, 8

demanding permissions and, 478

destructors and, 200

generics advantages for, 272–273

IL and improvements in, 5

monitoring Windows Services, 1308–1313

optimization in Visual Studio 2005, 399–400

pointer use for improving, 204, 219–222

processes versus application domains and, 15

properties and, 94–95

reference types and, 197

stack-based arrays for, 219–222

structs and, 103

value types and, 40

Web services specifications and, 1133–1134

performance monitoring for services

classes for, 1309

overview, 1309

perfmon.exe utility for, 1312–1313

performance counters for, 1309–1312

permcacl.exe permission-calculation tool, 481, 483–484

permissions for code access

changing for code groups, 494–495

creating and applying permissions sets, 495–497

creating custom, 487–488

defined, 464

- error handling blocks for, 472
- Identity Permissions, 472–473
- managing, 492
- named sets of, 473
- permission classes provided by CLR, 471–472
- policy levels, 476–478
- SQL Server assemblies, 634–635
- viewing for assemblies, 473–476
- permissions in .NET Framework**
 - asserting permissions, 486–487
 - CLR and demands for permissions, 478
 - creating custom code access permissions, 487–488
 - demanding permissions, 478, 479–480
 - denying permissions, 485–486
 - implicit permission, 484–485
 - performance issues, 478
 - requesting permissions, 480–484
 - specifying declaratively, 488
- PictureBox control (Windows Forms), 772**
- pictures. See graphics; resources**
- platform independence, IL and, 4–5**
- pluggability (.NET Remoting)**
 - of channels, 1024–1025
 - of proxy objects, 1029
- plus sign (+)**
 - in for loop iterator, 52
 - as increment operator (++), 135–136
 - regular expression groups and, 242
- Point struct, 851–852**
- pointer member access operator, 212**
- pointers. See also delegates**
 - address operator, 207
 - arithmetic, 210–211
 - for backward compatibility, 204
 - bugs introduced due to complexity of, 205
 - casting between pointer types, 209
 - casting to integer types, 208–209
 - to class members, 212–214
 - data types and, 208
 - declaring, 206–207
 - defined, 204
 - fixed keyword for, 213–214
 - function pointers, 171
 - high level trust needed at runtime, 205
 - indirection operator, 207
 - not allowed for arrays and classes, 208
 - for performance improvements, 204, 219–222
 - pointer member access operator, 212
 - PointerPlayaround example, 214–219
 - reference types compared to, 40, 204
 - sizeof operator with, 211
 - stack allocation for, 207–208
 - stack pointer, 194–195
 - stack-based arrays using, 219–222
 - to structs, 211–212
 - 32-bit processors and, 207–208
 - unsafe keyword for, 205–206
 - void, 209
- PointF struct, 851–852**
- policy levels for security**
 - overview, 476
 - viewing code groups at enterprise level, 477–478
 - viewing code groups at user level, 476–477
- Pop() method (Stack class), 254**
- pop-up menu for a database row**
 - code listing, 833–834
 - context menu functionality and, 832
 - ContextDataRow class for, 834
 - ContextMenuAttribute for, 834, 835
 - DataRow and DataTable classes for, 832
 - filtering methods on current object, 834
 - PopupMenu method for, 834–835
 - ways of providing, 832
- PopupMenu method, 834–835**
- pound symbol (#) for preprocessor directives, 70–73**
- power events, Windows Services and, 1314**
- #pragma preprocessor directive, 73**
- precedence of operators, 140**
- pre-emptive multitasking, 350**
- prefixes for variable names, 75**
- preprocessor directives**
 - #define, 70, 71
 - defined, 70
 - #elif, 71–72
 - #else, 71–72
 - #endif, 71–72
 - #endregion, 72–73
 - #error, 72
 - #if, 71–72
 - #line, 73
 - not ended by semicolon, 71
 - pound symbol beginning, 70
 - #pragma, 73
 - #region, 72–73
 - #undef, 71
 - #warning, 72
- printing**
 - displaying to screen versus, 894
 - graphics, 893–899
 - WebBrowser control for, 1255–1256

PrintingCapsEdit **project**

- compiling and testing, 898–899
- downloading, 895
- event handlers, 896–897
- Form1 class pagesPrinted field, 896
- LoadFile() method, 895
- using statements, 898

priorities for threads, 358–359

PriorityDocumentManager **class**

- AddDocument() method, 290
- AddDocumentToPriorityNode() method, 290–291
- code listing, 289–290
- GetDocument() method, 291–292
- LinkedList<Document> collection in, 289
- List<LinkedListNode<Document>> collection in, 289
- Main() method, 292

private assemblies

- overview, 18, 412–413, 419
- versioning and, 412–413, 451
- zero impact installation, 18

private member fields, camel casing for, 76

private modifier, 124

ProcessDocuments **class**

- custom generic class example, 294–296
- Queue<T> generic class example, 285–286

processes, application domains and, 14–16, 413

ProcessNextPerson() **method** (ColdCallFileReader **class**), 344–345

processors

- JIT compilation for, 5
- pointers and 32-bit processors, 207–208
- pre-emptive multitasking and, 350
- requirements for application deployment, 553

Professional ASP.NET 1.1 (Wiley Publishing, Inc.), 910, 940

prog id for COM objects, 1155

programming guidelines for C#

- rules for identifiers, 73–75
- usage conventions, 75–81

ProgressBar **control (Windows Forms), 772**

ProjectInstaller **class, 1288–1289**

properties

- access modifiers, 94
- for accessing user controls (ASP.NET), 947–948
- adding to user controls, 796–797
- Application class, 753
- casing of, 76–77

- of contexts (.NET Remoting), 1015, 1016

defining, 93

described, 86, 92

DirectoryInfo class, 1190

FileInfo class, 1190

of generic classes, 276–277

of generic interfaces, 275–276

get accessor, 93

of message queues, 1101–1102

naming conventions, 93

performance and, 94–95

quote server application, 1282–1283

read-only, 94

Registry class, 1231

ServiceController class, 1297

set accessor, 93

setting for channels (.NET Remoting), 1023–1024

System.SystemException class, 337

System.Type class, 314–315

usage conventions, 80

virtual, 114

Visual Studio 2005 Properties window, 394–396

write-only, avoiding, 81, 94

Properties window (Visual Studio 2005), 394–396

PropertyManager **class, 819, 820–822**

PropertyValueCollection **class, 734**

protected internal **modifier, 124, 125**

protected **modifier, 96, 124**

protocol stack, 1262

proxy classes

for delegates, 838

for .NET Remoting, 1012

transparent versus real proxy, 1012

for Web services, 990–992

proxy objects, .NET Remoting object activation and, 1027, 1029

public keyword

for constructors, 96

described, 124

for methods, 33

publisher policy files for assemblies, 450, 456–458

publishing Web sites, deployment using, 552, 555–556

Push() **method (Stack class), 254**

Q

QueryInterface() **method, 1154–1155**

question mark (?)

- for disabling Group object with regular expressions (?:), 243

- in null coalescing operator (??), 139–140
 - in ternary operator (?:), 136–137
 - Queue **class**, 256–257
 - Queue<T> **generic class**, 283–286
 - QueueUserWorkItem() **method** (ThreadPool **class**), 366
 - QuickArray **example**, 221–222
 - quote server application. See also specific classes and assemblies**
 - adding event logging, 1306–1308
 - handler methods, 1286
 - installation program, 1287–1292
 - overview, 1275–1276
 - project wizard for, 1281–1283
 - properties, 1282–1283
 - QuoteClient assembly, 1275, 1276, 1279–1281
 - QuoteServer class, 1275, 1276–1279
 - QuoteService assembly, 1275, 1276, 1281–1287
 - service start, 1285
 - ServiceBase class, 1283–1284
 - starting manually, 1292
 - TestQuoteServer console application, 1279
 - QuoteServer **class**
 - AcceptSocket() method, 1278
 - adding event logging, 1306–1308
 - constructor, 1277
 - creating, 1276–1279
 - GetRandomQuoteOfTheDay() method, 1277
 - ListenerThread() thread function, 1278
 - overview, 1275, 1276
 - ReadQuotes() method, 1277
 - RefreshQuotes() method, 1279
 - Resume() method, 1278–1279
 - Start() method, 1277–1278
 - Stop() method, 1278
 - Suspend() method, 1278–1279
 - QuoteService **assembly**
 - handler methods, 1286
 - Main() method, 1284–1285
 - OnContinue() method, 1286
 - OnCustomCommand() method, 1286
 - OnPause() method, 1286
 - OnShutdown() method, 1286
 - OnStart() method, 1285, 1287
 - OnStop() method, 1286
 - project wizard for, 1281–1283
 - service start, 1285
 - ServiceBase class, 1283–1284
- R**
- /r or /reference **switch for compiler**, 64
 - race conditions, avoiding**, 363–364
 - Racer **class**, 278–279
 - RadioButton **control (Windows Forms)**, 764
 - RainbowLabel **custom control (ASP.NET)**
 - Color enumeration, 953
 - creating Web site for, 953
 - Default.aspx.cs code for, 955
 - enabling color cycling, 954
 - modifying Default.aspx for, 955
 - Render() method override, 954–955
 - using, 955–956
 - using statements, 953
 - RAM. See also memory management**
 - freeing memory in COM, 1153
 - requirements for application deployment, 553
 - RCW (runtime callable wrapper) in COM**, 1163–1166
 - RDN (relative distinguished name) in Active Directory**, 727
 - Read() **method** (XmlReader **class**), 665
 - ReadElementString() **method** (XmlReader **class**), 665–667
 - reading drive information**, 1220–1222
 - reading files**
 - binary files, 1208–1213
 - overview, 1201
 - text files, 1213–1219
 - Windows Forms for, 1202–1204
 - ReadLine() **method** (StreamReader **class**), 1215
 - readonly **fields**, 99–101
 - read-only properties**, 94
 - ReadQuotes() **method** (QuoteServer **class library**), 1277
 - ReadStartElement() **method** (XmlReader **class**), 665
 - ReadToEnd() **method** (StreamReader **class**), 1215
 - ReadValueAs **methods** (XmlReader **class**), 667
 - ReadWriteText **application**, 1217–1219
 - ReadXml() **method** (DataSet **class**), 620, 696–697
 - ReadXmlSchema() **method** (DataSet **class**), 697
 - Receive() **method** (MessageQueue **class**), 1109, 1110
 - ReceivedById() **method**, 1119
 - Rectangle **struct**, 853–854
 - RectangleF **struct**, 853–854
 - Red-Green-Blue (RGB) values for colors**, 863–864, 865

ref parameters

- anonymous methods and, 177
- passing to methods, 90–91

refactoring, 408–410

reference objects, instantiating, 35

/reference or /r switch for compiler, 64

reference types. *See also specific types*

- arrays as, 58
- classes as, 40
- comparing for equality, 146–147
- compiler switch for, 64
- complex data types as, 40
- converting between value types and, 145–146
- defined, 9
- memory management, 196–198
- null value for, 40
- objects not created by, 39–40
- passing parameters to methods as, 90–91
- performance overhead, 197
- predefined, 41, 44–46
- stored in the heap, 9, 39, 196–198
- `System.Type` class for holding, 314
- value types versus, 9, 39–40

`ReferenceEquals()` **method**, 146, 147

references to shared assemblies, 449

reflection. *See also attributes*

- assemblies and, 19
- `Assembly` class, 319–320
- custom attributes, 306–313
- defined, 19, 305
- example applications described, 305–306
- `System.Type` class, 314–319
- uses for, 305

`RefreshQuotes()` **method** (`QuoteServer` class), 1279

regasm utility, 1179

regedit utility, 1227–1228

regedit32 utility, 1227

#region preprocessor directive, 72–73

`Region` struct, 854–855

`RegionInfo` class, 515

<%@ Register %> directive (ASP.NET)

- for custom controls, 952–953
- for user controls, 946

`RegisterActivatedClientType()` **method** (`RemotingConfiguration` class), 1027, 1029

`RegisterChannel()` **method** (`ChannelServices` class), 1025

`RegisterWellKnownClientType()` **method** (`RemotingConfiguration` class), 1028

`RegisterWellKnownServiceType()` **method** (`RemotingConfiguration` class), 1026, 1028

`Registry` class, 1229, 1230

`Registry Editor (ClickOnce)`, 570–571

registry operations

- COM registration, 1155, 1178–1179
- hierarchical structure of registry, 1227–1229
- keys, 1228–1229
- namespace for, 1187
- prevalence of registry use, 1226–1227
- regedit utility, 1227–1228
- regedit32 utility, 1227
- `Registry` class, 1229, 1230
- registry hive, 1228
- registry overview, 1226–1229
- `RegistryKey` class, 1229–1232
- security and, 1187
- `SelfPlacingWindow` application, 1232–1238

`RegistryKey` class

- `CreateSubKey()` method, 1230
- described, 1229
- `GetValue()` method, 1231
- methods, 1231–1232
- `OpenSubKey()` method, 1230
- properties, 1231
- reading registry keys, 1230
- `Registry` class versus, 1230
- `SetValue()` method, 1231
- writing to registry keys, 1230

regular expressions

- capturing sections from URIs, 242–243
- defined, 235
- disabling `Group` object in returns, 243
- displaying results, 240–241
- efficiency of, 236
- escape sequences, 236, 238–240
- groups, 240–243
- .NET classes supporting, 223, 236
- prevalence of, 235
- `RegularExpressionsPlayaround` example, 237–241
- searching for escape meta-characters, 240
- uses for, 236
- `WriteMatches()` method for, 240

`RegularExpressionsPlayaround` example, 237–241

relative distinguished name (RDN) in Active Directory, 727

releasing COM components, 1165

reliability, Web services specifications and, 1129–1131

remote objects

- defined, 1012
- distributed identity, 1017
- event handling and, 1056–1058
- overview, 1017–1018

RemoteObject **class**, **1056–1058**

RemotingConfiguration **class**

- Configure() method, 1044, 1045, 1046
- described, 1012, 1026
- predefined channels in, 1041
- RegisterActivatedClientType() method, 1027, 1029
- RegisterWellKnownClientType() method, 1028
- RegisterWellKnownServiceType() method, 1026, 1028

RemotingServices **class**

- Connect() method, 1028, 1030
- described, 1026
- IsTransparentProxy() method, 1029
- registering a well-known object to, 1026

Remove() **method (Hashtable class)**, **261**

RemoveAt() **method (ArrayList class)**, **251**

RemoveRange() **method (ArrayList class)**, **252–253**

removing. See deleting or removing

Render() **method, overriding**, **954–955**

Replace() **method**

- String class, 226
- StringBuilder class, 228

requesting permissions

- adding permissions as assembly attributes, 482
- calculating permissions required for an assembly, 481, 483–484
- options for applications, 483
- permission sets, 484
- reasons for, 480–481
- SecurityAction enumeration values, 482–483
- ways of requesting, 480

Reset() **method**, **247**

Resgen.exe (Resource File Generator) utility, **527, 543–544**

resource readers

- custom, 545–549
- .NET Framework, 545
- ResourceReader class, 534
- ResXResourceReader class, 534
- ResourceDemoForm **class**, **530–533**
- ResourceManager **class**, **530–531, 533, 537**
- ResourceReader **class**, **534**

resources. See also localization

- accessing embedded resources, 530–531
- adding to assemblies, 528–530
- automatic fallback for, 542
- creating resource files, 526–528
- custom messages for, 541–542
- custom resource reader, 545–549
- outsourcing translations, 543–544
- Resource File Generator (Resgen.exe) utility, 527
- ResourceDemoForm class for, 531–533
- ResourceManager class for, 530–531, 533
- ResourceReader class for, 534
- ResourceSet class for, 533
- ResourceWriter class for, 527–528, 534
- .resX files for, 526, 527–528
- ResXResourceReader class for, 534
- ResXResourceSet class for, 534
- ResXResourceWriter class for, 527–528, 534
- strongly typed, 531–533
- System.Resources namespace classes for, 533–534
- text files for, 526
- using resource files, 528–533
- Windows Forms, disposing of, 755
- ResourceSet **class**, **533**
- ResourceWriter **class**, **527–528, 534**
- response message queues**, **1100, 1120–1121**
- Resume() **method**
 - QuoteServer class library, 1278–1279
 - Thread class, 354
- resuming threads**, **353, 354**
- .resX files for resources**, **526, 527–528**
- ResXResourceReader **class**, **534**
- ResXResourceSet **class**, **534**
- ResXResourceWriter **class**, **527–528, 534**
- return statement**, **55**
- RGB (Red-Green-Blue) values for colors**, **863–864, 865**
- RichTextBox **control (Windows Forms)**, **773–774**
- role-based security**
 - declarative, 509–510
 - defined, 14, 506
 - Enterprise Services and, 1068
 - principal for, 507
 - roles, 509
 - uses for, 506–507
 - WindowsPrincipal example, 508–509
- Run() **method (Application class)**, **overloading**, **753**
- runat="server" **attribute (ASP.NET)**, **905, 910**
- runtime callable wrapper (RCW) in COM**, **1163–1166**

S

safety checking. See **type safety**

safety palette for colors, 866

SAX versus XmlReader class, 664

sbyte type, 41, 42

sc.exe utility, 1294–1295

schemas (Active Directory)

domain tree and, 717

getting the naming context for, 741–742

overview, 715, 719–720

schemas (database)

building an XSD schema, 827–832

defined, 601

hand-coded, 606–608

runtime generation of, 606

XLM schemas, 612–618, 691

SCM (Service Control Manager), 1273, 1274

scope of variables

clashes for fields and local variables, 37–38

clashes for local variables, 36–37

defined, 35

fields (member variables), 35

general rules, 35–36

local variables, 35–36

managed heap and, 197–198

reference variables, 197–198

stack and, 194–195, 197–198

using statement and, 202

scripting languages, interoperability with .NET, 7

scrolling windows in graphics applications

adding scroll bars, 858–859

converting coordinates relative to client area, 861–862

determining document size, 858–859

Form1 class for, 856–857

OnPaint() method override and, 859–860, 862

screen redraw and, 860–861

setting minimum document size at startup, 858–859

standard controls and, 857–858

SDI (Single Document Interface), 781

sealed modifier, 117–118, 125

searching in Active Directory

with DSML, 746–748

filters for, 736–737

PropertiesToLoad, 737

read-mostly access and, 736

SearchRoot, 736

SearchScope, 737

setting search limits, 737–740

UserSearch application, 740–745

security. See **also authentication**

ACLs, 1222–1226

Active Directory, 714–715

ASP.NET, 963–973

asserting permissions, 486–487

ClickOnce settings, 569–570

code access security, 464–478

configuration file, 489–492

declarative, 488, 509–510

demanding permissions, 478, 479–480

denying permissions, 485–486

Enterprise Services, 1068

file security, 1222–1226

implicit permission, 484–485

mobile code issues, 463

.NET Framework support for, 478–488

.NET Remoting, 1035, 1054–1055, 1126

performance issues, 478

policy management, 489–506

public key cryptography, 442–443

requesting permissions, 480–484

role-based, 506–510, 1068

role-based versus code-based, 14

strong data typing in IL and, 14

turning on and off, 493

Web services specifications and, 1128–1129

security policies. See **also caspol.exe (Code Access**

Security Policy tool)

applying full trust, 492

changing permissions for code groups, 494–495

creating and applying permissions sets, 495–497

creating custom code groups, 493–494

deleting code groups, 494

distributing code using certificates, 499–504

distributing code using strong names, 497–499

files storing, 489

managing, 489–506

managing code groups and permissions, 492

managing zones, 504–506

resetting, 493

security configuration file, 489–492

viewing code groups at enterprise level, 477–478

viewing code groups at user level, 476–477

Security Wizard (ASP.NET), 964–969

SecurityException, 480, 482, 492

secutil.exe tool, 498–499, 502

SelfPlacingWindow application, 1232–1238

semicolon (;)

ending C# statements, 31

preprocessor directives not ended by, 71

- `Send()` **method** (`MessageQueue` **class**), **1106, 1108**
- serializing. See also serializing objects in XML**
- base classes and interfaces for, 1187
 - defined, 1187
 - UDTs, 637–638
- serializing objects in XML**
- custom attributes for, 702
 - namespace for, 701–702
 - .NET Remoting security and, 1035
 - serializing, defined, 701
 - simple application for, 702–708
 - without source code access, 708–711
 - xsd.exe tool for, 702
- server controls (ASP.NET)**
- adding to PCSWebApp1 example, 911–913
 - control palette, 913–914
 - Crystal Reports Web server controls, 914
 - data Web server controls, 917–918
 - event handlers for, 911–912
 - example using, 920–925
 - further information, 910
 - HTML server controls, defined, 910
 - icon in system tray, 912
 - login Web server controls, 970–971
 - navigation Web server controls, 960–963
 - runat="server" attribute, 910
 - standard Web server controls, 914–917
 - validation Web server controls, 918–920
 - Web server controls, defined, 910
 - WebParts Web server controls, 920
- Server Explorer (Visual Studio), 1295**
- Server Explorer window (Visual Studio 2005), 398**
- server platforms supported by .NET, 553**
- Service Control Manager (SCM), 1273, 1274**
- `ServiceBase` **class**, **1283–1284, 1302, 1305, 1314**
- `ServiceController` **class**
- application using, 1295–1296
 - controlling services using, 1299–1301
 - methods, 1300
 - monitoring services using, 1296–1299
 - properties, 1297
- `ServiceComponent` **class**, **1070**
- `ServiceInstaller` **class**, **1289, 1290–1291**
- `ServiceInstallerDialog` **class**, **1291**
- `ServiceProcessInstaller` **class**, **1289–1290**
- Services administration tool, 1272**
- set **accessor of properties**
- access modifiers, 94
 - overview, 93
 - for user controls (ASP.NET), 948
- `SetDateAndNumber()` **method**, **535–536**
- setreg.exe utility, 501**
- setup programs. See Windows Installer projects**
- `SetValue()` **method** (`RegistryKey` **class**), **1231**
- shapes. See also graphics**
- application for drawing, 844–847
 - drawing, 868–870
 - painting, 847–848
 - `ScrollMoreShapes` example, 869–870
 - `System.Drawing.Graphics` methods for, 868–869
- shared assemblies**
- Authenticode signatures for, 442
 - creating, 444–449
 - defined, 19
 - delayed signing of, 448
 - global assembly cache for, 19, 413, 438–441
 - installing, 446, 451
 - integrity using strong names, 443
 - namespaces for classes and, 442
 - overview, 413, 419
 - public key cryptography, 442–443
 - references, 449
 - risks, 19
 - strong names for, 441–443, 444–445
 - using, 446–448
 - versioning, 412, 413, 451–460
- SharePoint Web sites information, 920**
- short **type**, **41, 42**
- `Show()` **method** (`Form` **class**), **782, 845**
- `ShowActivatedServiceTypes()` **method**, **1044–1045**
- `ShowChannelProperties()` **method**, **1023**
- `ShowDialog()` **method** (`Form` **class**), **782–784**
- `ShowRegionInformation()` **method**, **524**
- `ShowSamples()` **method**, **523**
- `ShowWellKnownServiceTypes()` **method**, **1044–1045**
- signcode.exe utility, 499**
- signtool.exe wizard, 499–501**
- Simple Object Access Protocol. See SOAP**
- SimpleClientApp solution**
- adding the assembly, 560–561
 - application properties and values, 559–560
 - Configuration Properties setting, 557
 - creating a desktop shortcut, 561
 - creating folders for deployment, 561
 - deployment project as separate solution, 558–562
 - deployment project in same solution, 563–564
 - project properties, 561–562
- `SimpleComponent` **class**, **1072–1073**

SimpleDelegate **delegate**, **177–180**

SimpleExceptions **class**, **333–336**

SimpleServer **application (Enterprise Services)**

assembly attributes, 1071

creating C# library application for, 1070

creating the component, 1072–1073

ServiceComponent class, 1070, 1072

signing the assembly, 1070

SimpleComponent class, 1072–1073

SimpleWebApp solution, **564–565**

Single Document Interface (SDI), **781**

single quotes ('), char literals enclosed by, **44, 46**

single-threaded apartment (STA) in COM,
1156–1157, 1166

SiteMapPath **control**, **960–962**

Size **struct**, **852–853**

SizeF **struct**, **852–853**

sizeof **operator**, **138, 211**

slash (/)

for multi-line comments (/* and */), 31, 67–68

for single-line comments (//), 31, 67

Sleep() **method (Thread class)**, **355**

sn (**strong name tool**), **442**

soap **namespace**, **984–985**

SOAP (Simple Object Access Protocol)

calling a method in a Web service, 984–985

declaring custom header for Web service, 1002
defined, 983

exchanging data using SOAP headers, 1001–1007
overview, 984–985

performance issues, 1133

RM sequence in messages, 1130–1131

sending data over HTTP and, 984–985

SoapHeaderAttribute attribute, 1002, 1003

SOAP-over-UDP 1133–1134

Soapsuds utility, 1052–1053

WCF message contract for, 1137, 1139

WS-Addressing specification with, 1134

Soapsuds utility, **1052–1053**

Socket **class**, **1263, 1268**

Sort() **method**

Array class, 525

BubbleSorter class, 180–181

globalization and, 525

List<T> generic class, 281–282

SortedList **class**, **257–259**

sorting. *See also* Sort() **method**

bubble-sorting algorithm, 180

DataView rows, 807–809

globalization and, 524–526

specific cultures, **515–516**

SplitContainer **control (Windows Forms)**, **775**

SQL Server 2005

key generation with, 627–629

namespace for, 635

as .NET runtime host, 634–635

stored procedures, 645–647

triggers, 648–650

user-defined aggregates, 643–645

user-defined functions, 647–648

user-defined types (UDTs), 636–643

XML data type, 650–657

SqlContext **class**, **635**

SqlPipe **class**, **635**

SqlTriggerContext **class**, **635**

square brackets ([]) for arrays, **57**

STA (single-threaded apartment) in COM,
1156–1157, 1166

stack

classes stored on, 9

creating stack-based arrays, 219–222

defined, 194

managed heap and, 196–198

pointer allocation on, 207–208

scope of variables and, 194–195

stack pointer, 194–195

value types stored on, 9, 39, 194–195

Stack **class**

foreach loops for reading values, 254

LIFO (Last In First Out) structure of stacks, 253–254

methods, 255–256

namespace for, 254

Pop() method for pulling items permanently, 254–255

Push() method for adding items, 254

Queue class versus, 256

uses for, 253

stackalloc **command**, **219–220, 221**

StackOverflowException **class**, **329**

Start() **method**

QuoteServer class, 1277–1278

Thread class, 352

static classes, **106**

static constructors

example, 97–98

execution of, 96–97

instance constructor in same class, 97

static **keyword**, **33, 125**

Stop() **method (QuoteServer class)**, **1278**

stored procedures

calling in ADO.NET, 594–597

creating for SQL Server, 645–646

naming conventions, 630

- for populating a DataSet, 619–620
- triggers for SQL Server, 648–650
- using with SQL Server, 646–647
- StreamReader class**
 - byte code markers, 1213
 - closing streams, 1215
 - constructors, 1214
 - described, 1207
 - encoding and, 1213, 1214–1215
 - FileStream class versus, 1213
 - hooking up to a FileStream, 1215
 - methods for reading or writing one line at a time, 1213
 - options, 1214
 - reading in arrays, 1215–1216
 - ReadLine() method, 1215
 - ReadToEnd() method, 1215
 - ReadWriteText application, 1217–1219
- streams. See also specific classes**
 - base class for, 1206
 - BinaryFileReader application, 1210–1213
 - BinaryReader class, 1207
 - BinaryWriter class, 1207
 - buffered streams, 1207
 - class hierarchy, 1206
 - defined, 1205
 - directions of data transfer, 1205
 - FileStream class, 1207, 1208–1210
 - outside source for, 1205–1206
 - overview, 1205–1207
 - ReadWriteText application, 1217–1219
 - StreamReader class, 1207, 1213, 1214–1216
 - StreamWriter class, 1207, 1213–1214, 1216–1217
- StreamWriter class**
 - byte code markers, 1213
 - constructors, 1216
 - described, 1207
 - encoding and, 1213–1214
 - FileStream class versus, 1213
 - hooking up to a FileStream, 1216
 - methods for reading or writing one line at a time, 1213
 - ReadWriteText application, 1217–1219
 - Write() method overloads, 1216–1217
 - writing data into a new file, 1216
- String class**
 - Format() method, 231
 - inefficiency for repeated modifications, 223, 225–226
 - methods, 224–225
 - Replace() method, 226
 - StringBuilder class versus, 223, 226
- string keyword, 45**
- string tables. See resources**
- string type**
 - char type and, 44
 - converting between numeric types and, 144–145
 - new string objects created by changes, 45–46
 - other reference types versus, 45–46
 - overview, 45–46
 - stored in the heap, 45
- StringBuilder class**
 - Append() method, 227
 - AppendFormat() method, 231
 - ArrayList class compared to, 250
 - memory allocation for, 226–227
 - methods, 228–229
 - Replace() method, 228
 - String class versus, 223, 226
 - ToString() method, 229
- StringInfo class methods, 514–515**
- strings. See also text, graphical**
 - concatenating using operator overloads, 224
 - extracting characters, 224
 - formatting expressions, 223, 229–235
 - Type class properties for retrieving, 314–315
- strong data typing in IL**
 - application domains and, 14–16
 - CLS and, 12–13
 - CTS and, 10–12
 - defined, 9
 - garbage collection and, 13–14
 - generics and, 269
 - importance for interoperability, 10–13
 - security and, 14
 - VB6 typing versus, 9–10
- strong name tool (sn), 442**
- strong names**
 - distributing code using, 497–499
 - integrity using, 443
 - for shared assemblies, 441–443, 444–445
- strongly typed objects in Active Directory, 715, 719**
- strongly typed resources, 531–533**
- strongly typed XML, 656–657**
- structs**
 - accessing fields outside the object, 37–38
 - ArraySegment<T> generic struct, 303–304
 - classes versus, 84, 102
 - for color representation, 863
 - constructors, 104
 - for coordinate representation, 850
 - declaring instances, 84, 101
 - default initialization of variables in, 34
 - defined, 84

structs (continued)

structs (continued)

- defining functions for, 101–102
- derived-struct syntax, 113
- enumerations instantiated as, 57
- inheritance and, 103, 112–113
- initialization, 103
- Nullable<T> generic struct, 301–302
- partial, 104
- performance and, 103
- pointers to, 211–212
- primitive types as .NET structs, 40–41
- public fields in, 102
- stored on the stack, 84
- uses for, 101
- as value types, 84, 102–103

style of names, 77

StyleSheetTheme (ASP.NET), 974–975

submitButton_Click() event handler, 924, 931–932, 934

SupportsWhatsNewAttribute class, 311–312

Suspend() method

QuoteServer class, 1278–1279

Thread class, 353, 354

suspending threads, 353–354

Swap<T> generic method, 297

switch...case statements

overview, 49–51

for user control event handler, 797–798

synchronization in threading

deadlocks, avoiding, 361–363, 364

defined, 360

importance of, 359

overusing, avoiding, 361

race conditions, avoiding, 363–364

syntax of C#, 360–361

System namespace, .NET types in, 32

System.ApplicationException class, 328

System.Array class, 245

System.AttributeUsage attribute, 307–309

System.Data namespace, 580, 581

System.Data.Common namespace, 580, 581

System.Diagnostics namespace, 1309

System.DirectoryServices namespace

accessing native ADSI objects, 735–736

ADSI COM object wrappers, 723

classes, 725

System.DirectoryServices.Protocols namespace

classes, 746

described, 724, 745

System.Drawing namespace

Color struct, 863

described, 843

Graphics methods, 868–869

Image class, 871

structs defined in, 850

System.GC.Collect() method, 199

System.Globalization namespace

CultureInfo class, 515

overview, 514, 523

RegionInfo class, 515

System.Net.Sockets namespace, 1263

System.Resources namespace, 533–534

System.Security.Permissions namespace, 479

System.ServiceProcess namespace, 1275

System.SystemException class, 328, 337

System.Type class

for holding reference types, 314

methods, 315–316

obtaining a Type reference, 314

properties, 314–315

TypeView example, 316–319

System.Xml namespace. *See also specific classes*

advantages over MSXML, 664

classes for handling XML, 661

classes for reading and writing XML, 660–661

language interoperability and, 659

MSXML 3.0 model and, 664

overview, 659

XmlDocument class, 661, 673–678

XmlNode class, 661, 672

XmlReader class, 660, 664–670

XmlTextReader class, 661, 664

XmlTextWriter class, 661, 664

XmlWriter class, 660, 664, 670–672

System.Xml.Serialization namespace, 701–702

System.Xml.XPath namespace

described, 678

key classes in, 679

using classes from, 681–684

XPathDocument class, 678

XPathNavigator class, 678, 679–680

XPathNodeIterator class, 678, 680–681

System.Xml.Xsl namespace

invoking methods during XML transforms, 686–689

overview, 684

transforming XML into HTML, 684–686

XsltArgumentList, 686–689

T

/t or /target **switch for compiler file type, 63–64**

TabControl **control (Windows Forms), 776**

TableLayoutPanel **control (Windows Forms), 774, 775**

TabPage **control (Windows Forms), 776**

/target or /t **switch for compiler file type, 63–64**

taskbar, showing forms in, 782

TCP classes, 1263

TCP versus UDP, 1266

TcpClient **class**

overview, 1263

Windows Services example, 1279–1281

TcpListener **class, 1263**

TcpReceive **application, 1264–1266, 1268**

TcpSend **application, 1263–1264**

Terminate() **method, 644–645**

ternary operator, 136–137

TestQuoteServer **console application, 1279**

TestSchema.cs **example**

adding elements, 829

DataColumn for, 831–832

DataRow for, 830–831

DataTable for, 829

EventArgs for, 832

initial code, 828

text, graphical. See also CapsEditor example

drawing, 874–880

editing a text document, 880–893

enumerating font families, 878–880

fonts and font families, 876–880

simple example, 874–876

TextBox **control (Windows Forms)**

adding controls to user control, 795–796

in application example, 756

DataBindings property, 816–817

overview, 773

WebBrowser control with, 1246–1247

TextBoxBase **class (Windows Forms)**

MaskedTextBox control derived from, 773, 774

RichTextBox control derived from, 773–774

TextBox control derived from, 773

TextLineInformation **class, 882–883**

themes (ASP.NET)

applying to pages, 974–975

defining, 975

overview, 974

in PCSDemoSite application, 975–979

thick-client applications, 22, 24

32-bit processors, pointers and, 207–208

this **keyword, 38, 96**

Thread **class**

Abort() method, 354

constructor, 352

CurrentCulture and CurrentUICulture
properties, 516–517, 518, 519, 539

Join() method, 354

namespace for, 366

overview, 351

Resume() method, 354

Sleep() method, 355

Start() method, 352

Suspend() method, 353, 354

ThreadPool class versus, 365

ways of manipulating, 355

ThreadAbortException, **354**

threading

aborting threads, 354

anonymous methods for, 353

COM, 1156–1157, 1166

deadlocks, avoiding, 361–363, 364

first thread started at Main() method, 350

by Internet Explorer, 349–350

lock statement, 360–361

namespace for, 351, 366

passing information to methods, 352

pre-emptive multitasking for, 350

priorities for threads, 358–359

problems with workarounds to avoid, 350

putting threads to sleep, 355

race conditions, avoiding, 363–364

resuming threads, 353, 354

starting threads, 351–353

suspending threads, 353–354

synchronization of access to variables, 359–364

thread, defined, 349

ThreadPlayaround example, 355–358

ThreadPool class for, 364–368

ThreadStart delegate, 352, 353

time slice of threads, 351

waiting for a thread to terminate, 354

Windows Services and, 1287

worker threads, 352

ThreadPlayaround **example, 355–358**

ThreadPool **class**

described, 364

maximum threads per processor, 364

namespace for, 366

QueueUserWorkItem() method, 366

ThreadPool class (continued)

ThreadPool class (continued)

Thread class versus, 365

uses for, 364

using, 365–368

WaitCallback delegate, 366

ThreadStart **delegate**, **352, 353**

throwing exceptions

from catch and finally blocks, 339

choosing type to throw, 334–335

defined, 331

nested throw statements for, 332

user-defined exceptions, 343–346

tiered development with ADO.NET, 625–627

time slice of threads, 351

tlbexp utility, 1173

tlbimp utility, 1163

ToolStrip **control (Windows Forms)**

controls derived from, 776–777

described, 776

formatting text, 776

ToolStripControlHost control, 777–779

ToolStripDropDownItem control, 777

using as a toolbar, 776

ToolStripContainer **control (Windows Forms), 780**

ToolStripControlHost **control (Windows Forms), 777–779**

ToolStripDropDownItem **control (Windows Forms), 777**

ToolStripManager **class (Windows Forms), 780**

ToolStripMenuItem **class (Windows Forms), 779–780**

ToString() **method**

called by AppendFormat() method, 231

date formatting for globalization using, 519

described, 106

of IFormattable interface, 232

number formatting for globalization using, 517–518

outputting StringBuilder class contents as a String, 229

for UDTs, 640–641

using, 108–109

TrackBar **control, 820–822**

transactional message queues, 1121–1122

transactions

in ADO.NET, 588–589

Enterprise Services, 1068, 1078–1080

Web services specifications and, 1131–1133

TreeView **control (ASP.NET), 960, 961**

TreeView **control (Windows Forms)**

custom control based on, 788–794

described, 788, 789

triggers for SQL Server

creating, 649–650

defined, 648

overview, 648–649

using, 650

troubleshooting Windows Services

debugging, 1301–1302

event logging, 1302–1308

interactive services, 1302

performance monitoring, 1308–1313

try blocks

defined, 330

error trapping process and, 330

exceptions not handled and, 338

handling different exceptions in different places, 340

modifying the type of exception, 340

MortimerColdCall example, 342

nested throw statements in, 332

nested try blocks, 338–340

for releasing database resources, 586, 587–588

in SimpleExceptions class, 334

syntax, 330

throwing an exception and, 331

T-SQL, 645

Type class

for holding reference types, 314

methods, 315–316

obtaining a Type reference, 314

properties, 314–315

TreeView example, 316–319

type safety

boxing and unboxing, 145–146

CLR memory safety tests, 7

delegates and, 172, 174

generics advantages for, 273

strong typing in IL, 9–16

type conversions, 140–145

unsafe operators, 134

typeof **operator, 139**

types. See data types

TreeView assembly

AddToOutput() method, 319

AnalyzeType() method, 318–319

code listing, 318–319

compiling, 319

GetMembers() method, 319

Main() method, 318

overview, 316–318

using statements, 318

U**UDP protocol**

- SOAP-over-UDP, 1133–1134
- TCP versus, 1266
- UdpClient class, 1266–1267

UDTs (user-defined types). See also Coordinate**struct**

- converting from a string, 641
- converting to a string, 640–641
- creating, 636–641
- deploying, 642
- enumerations, 55–57
- limitations on, 636
- object type as parent, 44
- overview, 636
- serialization formats, 637–638
- [SqlUserDefinedType] attribute, 637, 638
- using from client-side code, 642–643
- Visual Studio template for, 636–637

unboxing. See boxing and unboxing**unchecked operator, 137****#undef preprocessor directive, 71****UnexpectedException, 341, 347****Unicode characters**

- combining, 514
- globalization issues, 514–515
- in identifiers, 74–75
- StringInfo methods for combined characters, 515

unsafe code. See also pointers; type safety

- anonymous methods and, 177
- unsafe operators, 134

unsafe keyword, 205–206**updating Active Directory entries, 733****updating data sources with Visual Studio .NET, 826****updating records with ADO.NET**

- calling stored procedures, 594–595
- ExecuteNonQuery() method for, 591
- setting update constraints, 611–612
- using data adapters, 621–623

UploadData() method (WebClient class), 1242**UploadFile() method (WebClient class), 1242****UPN (user principal name) in Active Directory, 729****Uri class, 1258–1259****UriBuilder class, 1258, 1259****URLs, capturing sections with regular expressions, 242–243****usage conventions**

- casing of names, 76–77
- compilation and, 75

- fields, 81
- name styles, 77
- names and keywords, 78–80
- namespace names, 77
- .NET guidelines and naming philosophy, 75–76
- properties and methods, 80
- variable prefixes, 75

user controls (ASP.NET)

- adding methods, 949–950
- attributes for, 947
- creating Web site for, 945
- custom controls versus, 952
- defined, 944
- defining default state for, 946
- event handlers, 948, 950
- file extensions for code, 945
- graphics for card suit display, 946
- naming HTML tag for, 946
- overview, 944–945
- in PCSDemoSite application, 950–952
- properties for accessing, 947–948
- <%@ Register %> directive for, 946
- simple example, 945–950
- using in Web pages, 950

user controls (Windows Forms)

- adding properties, 796–797
- adding TextBox controls, 795–796
- attributes for, 798–799
- creating the container control, 795
- further enhancements possible for, 799
- get methods for properties, 797
- initialization, 798
- set methods for properties, 797–798
- TextChanged event handler, 797–798
- uses for, 794

user interaction. See also events

- CapsEditor example, 890–893
- Control class and, 760–761
- with Form class, 782–784
- interactive Windows Services, 1302

User Interface Editor (ClickOnce), 571–573**user principal name (UPN) in Active Directory, 729****user-defined aggregates**

- creating, 644–645
- deploying, 645
- uses for, 643
- using, 645

user-defined casts

- between base and derived classes, 163–164
- boxing and unboxing, 164–165

user-defined casts (continued)

- compilation and, 165–166
- converting integers to floats, 159–162
- between derived classes, 162–163
- implementing, 159–165
- implicit versus explicit casts and, 157–158
- multiple casting, 165–169
- syntax, 158–159

user-defined exception classes

- catching user-defined exceptions, 341–343
- defining, 346–348
- MortimerColdCall example, 340–348
- throwing user-defined exceptions, 343–346

user-defined functions with SQL Server, 647–648

user-defined types. See UDTs

UserSearch Active Directory application

- described, 740
- getting the schema naming context, 741–742
- getting user class property names, 742–743
- searching for user objects, 743–745
- user interface, 740–741

using statements

- CapsEditor example, 881–882
- for custom controls (ASP.NET), 953
- Dispose() method and, 201–202
- for Enterprise Services client, 1077
- for event-booking application, 995
- FileProperties application, 1193–1194
- making OleDb classes available, 597
- for namespace aliases, 60–61
- for namespaces, 32, 59–60
- PrintingCapsEdit project, 898
- QuoteClient assembly, 1280
- for releasing database resources, 586–587, 588

V

validation Web server controls (ASP.NET), 918–920

ValidationSummary control, 921

value types. See also specific types

- basic data types as, 40
- for Boolean values, 43
- comparing for equality, 147–148
- converting between reference types and, 145–146
- decimal type, 42–43
- defined, 9
- floating-point types, 42
- integer types, 41–42
- memory management, 194–195
- performance and, 40

- predefined, 41–44
- reference types versus, 9, 39–40
- for single characters, 43–44
- stored on the stack, 9, 39, 194–195

variables. See also constants

- anonymous methods and, 177
- assigning values, 34
- declaring, 33–34
- initialization, 34–35
- prefixes for names, 75
- scope, 35–38, 194–195
- unsafe local variables, 206

Vector struct

- adding collection support, 248–250
- adding formatting expressions, 233–235
- overloading operators, 150–157

VectorClass assembly, 310, 312–313

VectorEnumerator struct, 248–250

versioning assemblies

- application configuration files, 453–456
- getting the version programmatically, 452–453
- importance for shared assemblies, 412, 413, 451
- private assemblies and, 412–413, 451
- publisher policy files, 456–458
- runtime version, 459–460
- version numbers, 451–452

viewing .NET data. See also ADO.NET; specific topics

- data binding, 816–822
- DataGridView control for, 801–816
- Visual Studio .NET and data access, 822–840

viewstate field (ASP.NET), 905

virtual addressing, 194

Virtual Execution System of CLR, 465

virtual keyword, 114–115, 125

virtual memory, 194. See also managed heap; stack

visibility modifiers, 124–125

visibility of forms, 781

Visual Basic 2005

- development of, 5–6
- interoperability with .NET, 5–6

Visual Basic 6

- class for CLS example, 432–434
- converting to Visual Basic 2005, 6
- data typing versus IL data typing, 9–10
- reading in projects to Visual Studio 2005, 387
- unsuitability for .NET, 5–6

Visual Basic .NET

- extensive changes from VB6, 6
- not case-sensitive, 77

Visual C++ 2005. See also C++

- advantages over C#, 6–7
- CLR memory type safety tests and, 7
- interoperability with .NET, 6–7
- mixing managed and unmanaged types in, 6

Visual C++ 6. See also C++

- Microsoft-specific extensions, 6
- reading in projects to Visual Studio 2005, 387

Visual C++ .NET, 6. See also C++**Visual J# 2005, 7****Visual Studio .NET and data access. See also Visual Studio 2005**

- building an XSD schema, 827–832
- creating a database connection, 822–825
- manufactured tables and rows, 835–840
- providing a pop-up menu for a row, 832–835
- selecting data from a database, 825–826
- updating the data source, 826

Visual Studio 2005. See also Visual Studio .NET and data access

- accessing other programs from, 373
- adding projects to solutions, 385–386
- adding TextBox controls to forms, 392–394
- for ASP.NET file creation, 906
- ATL Project Wizard, 1159–1160
- building projects, 399–403
- changes from previous versions, 373–374
- Class View window, 396–397
- compiling from, 372
- console project, 381–383
- converting VB6 to Visual Basic 2005 and, 6
- creating assemblies using, 423–425
- creating installers in, 557–567
- creating projects, 377–383
- creating UDTs, 636–641
- creating Web Forms, 906–908
- debug and release builds with, 399–401
- debugger, 9, 373, 404–407
- debugger symbols, 400–401
- Design View window, 372, 391–394
- designer-generated code separated by, 754–755
- design-time debugging, 390
- editing configurations, 402–403
- Exceptions dialog box, 407
- features, 371–373
- folding editor (default code editor), 388–390
- localization example using, 534–555
- MSDN help integrated with, 373
- Object Browser window, 397–398
- opening projects from previous versions in, 374–376

- pinning and unpinning windows, 398–399
 - project, defined, 383
 - project files created by, 382–383
 - project types available, 379–381
 - Properties window, 394–396
 - RCW creation, 1164
 - reading in Visual Studio 6 projects, 387
 - refactoring tools, 408–410
 - removing debugging commands, 401
 - selecting a project type, 377–381
 - selecting configuration for, 401–402
 - Server Explorer window, 398
 - setting breakpoints, 404
 - setting the startup project, 386
 - showing and hiding blocks of code, 388–390
 - simple client application, deployment project as
 - separate solution, 558–562
 - simple client application, deployment project in same
 - solution, 563–564
 - solution, defined, 383
 - solutions versus projects, 383–384
 - supporting windows, 372
 - text editor, 371
 - Toolbox, 391–392
 - upgrade wizard, 374–376
 - User-Defined Type template, 636–637
 - Visual J# 2005 and, 7
 - Watch window, 405–406
 - Web Forms, 23
 - Windows application code, 387
 - Windows Form applications and, 754–755
- void pointers, 209**

W**WaitCallback delegate, 366****#warning preprocessor directive, 72****WCF (Windows Communication Foundation)**

- architecture, 1135
- ASP.NET and, 1146
- binding, 1141–1143
- client applications, 1144
- contracts, defined, 1137
- data contract, 1137, 1138
- described, 1125
- Enterprise Services and, 1146–1147
- features, 1134–1135
- hosting, 1143–1144
- layers of applications, 1135, 1136
- message contract, 1137, 1139

WCF (Windows Communication Foundation) (continued)

WCF (Windows Communication Foundation) (continued)

- Message Queuing and, 1147
- .NET Remoting and, 1145–1146
- overview, 1134–1136
- preparing for, 1145–1147
- programming with, 1137–1144
- service contract, 1137–1138
- service implementation, 1139–1141
- ways of sending messages, 1135
- Web services contract, 1137

Web Forms. *See also* ASP.NET; PCSDemoSite application (ASP.NET)

- ADO.NET and data binding for, 926–939
- application configuration, 939–941
- authentication using Security Wizard, 964–969
- creating files with text editor, 905
- creating in Visual Studio 2005, 906–908
- creating with C#, 23
- defined, 23
- design considerations, 905–908
- IDEs for, 905
- Label control, 920
- login system implementation, 969–970
- <%@ Page %> tag, 908
- PCSWebApp1 example, 906–908, 909
- process after user requests, 909
- restricting directory access, 971–973
- security, 963–973
- server control example, 920–925
- state management in ASP.NET and, 905
- Web server controls for, 23

Web server, client deployment from, 566–567

Web server controls (ASP.NET). *See also* event-booking application

- adding to PCSWebApp1 example, 911–913
- binding to the database, 927–928
- Crystal Reports controls, 914
- data controls, 917–918
- defined, 23, 910
- login controls, 970–971
- navigation controls, 960–963
- runat="server" attribute, 910
- standard controls, 914–917
- validation controls, 918–920
- WebParts controls, 920

Web Service Description Language. *See* WSDL

Web services. *See also* Web services specifications

- adding a method accessible through, 988
- consuming, 990–993
- CookieContainer class with, 1004–1005

- creating client for, 991–992
- declaring custom SOAP header, 1002
- defined, 983
- event-booking application, 993–1001
- exchanging data using SOAP headers, 1001–1007
- exposing, 987–990
- with .NET Remoting, 1010
- proxy class for, 990–992
- SOAP for, 983, 984–985
- types available for, 990
- viewing WSDL for, 990
- WCF contract for, 1137
- WSDL for, 983, 985–987
- WSDL.exe tool, 991

Web services specifications

- current issues, 1126–1127
- performance and, 1133–1134
- for transactions, 1131–1133
- WS-Addressing, 1134
- WS-AtomicTransactions, 1131
- WS-BusinessActivity, 1131–1133
- WS-Coordination, 1131
- WS-Federation, 1127
- WS-ReliableMessaging, 1129–1131
- WS-SecureConversation, 1128–1129
- WS-Security, 1128

WebBrowser control (Windows Forms)

- displaying the code of a requested page, 1256–1257
- giving applications IE-type features, 1248–1253
- printing using, 1255–1256
- showing documents using, 1254–1255
- for simple Web browsing from applications, 1246–1247
- starting an Internet Explorer process programmatically, 1245–1246

WebClient class

- BasicWebClient example, 1240–1241
- DownloadFile() method, 1240
- OpenRead() method, 1240–1241
- OpenWrite() method, 1241
- overview, 1240
- UploadData() method, 1242
- UploadFile() method, 1242

web.config file

- ASP.NET application configuration, 940
- for event-booking application, 993
- <globalization> element, 544
- hosting .NET Remoting servers in ASP.NET and, 1050–1051

WebParts Web server controls (ASP.NET), 920

- WebRequest **class**
 - asynchronous page requests, 1244–1245
 - authentication, 1244
 - BasicWebClient example, 1242–1243
 - class hierarchy, 1257–1258
 - HTTP header information support, 1243–1244
 - WebClient class versus, 1242
- WebResponse **class**
 - BasicWebClient example, 1242–1243
 - class hierarchy, 1257–1258
 - HTTP header information support, 1243–1244
 - WebClient class versus, 1242
- WelcomeMessage() **method**, **535**, **536**, **541–542**
- Wellknown_Client.config **file (.NET Remoting)**, **1043**
- Wellknown_Server.config **file (.NET Remoting)**, **1042–1043**
- WhatsNewAttributes **example**
 - LookUpWhatsNew assembly, 310, 321–324
 - overview, 310–311
 - VectorClass assembly, 310, 312–313
 - WhatsNewAttributes library assembly, 310, 311–312
- WhatsNewChecker **class**, **321–322**
- while **statements**
 - overview, 53
 - scope of local variables declared in, 36
 - in SimpleExceptions class, 334
- Wiley Publishing, Inc.**
 - Beginning XML*, 659
 - Professional ASP.NET 1.1*, 910, 940
- Windows Controls**, **24**
- Windows Forms. See also events; graphics; specific classes, controls, and components**
 - ActiveX controls in, 1168–1171
 - appearance properties for controls, 760
 - application for ActiveX controls, 1169–1170
 - Button control, 762–763, 1249–1253
 - CheckBox control, 764
 - CheckedListBox control, 765
 - class hierarchy, 757–758
 - ComboBox control, 765, 767
 - ContextMenuStrip class, 779
 - Control class, 759–762
 - controlling appearance of forms, 784–786
 - creating an application, 752–757
 - custom controls, 787–799
 - DateTimePicker control, 765
 - disposing of forms, 755
 - ErrorProvider component, 767–769
 - FlowLayoutPanel control, 774–775
 - focus, gaining and losing for controls, 761
 - FolderTree custom control, 789–794
 - Form class, 752, 780–786
 - Form, defined, 780
 - globalization demo application, 520–524
 - grouping controls on tab pages, 776
 - Height property, 92–93
 - HelpProvider component, 769
 - ImageList component, 769
 - Label control, 769–770
 - LinkLabel control, 1248
 - ListBox control, 765–767
 - ListView control, 770–772
 - Localizable property, 536, 537
 - localization application, 520–524
 - MaskedTextBox control, 773, 774
 - MDI (Multiple Document Interface), 786–787
 - MenuStrip control, 779
 - namespaces for, 751
 - overview, 24
 - Panel control, 774
 - PictureBox control, 772
 - ProgressBar control, 772
 - RadioButton control, 764
 - reading drive information, 1220–1222
 - reading from files, 1202–1204
 - RichTextBox control, 773–774
 - running controls in Internet Explorer, 1183
 - SDI (Single Document Interface), 781
 - setting colors for forms, 785
 - shape-drawing application, 844–847
 - showing forms in taskbar, 782
 - size and location properties for controls, 759–760
 - SplitContainer control, 775
 - standard controls and components, 762–780
 - system menu for forms, 784–785
 - TabControl control, 776
 - TableLayoutPanel control, 774, 775
 - TabPage control, 776
 - TextBox control, 773, 1246–1247
 - ToolStrip control, 776–779
 - ToolStripContainer control, 780
 - ToolStripControlHost control, 777–779
 - ToolStripDropDownItem control, 777
 - ToolStripManager class, 780
 - ToolStripMenuItem class, 779–780
 - TreeView control, custom control based on, 788–794
 - user interaction, 760–761, 782–784
 - UserSearch Active Directory application, 740–745

Windows Forms (continued)

- visibility of forms, 781, 782
- visual styles for forms, 785–786
- WebBrowser control, 1245–1247, 1248–1257
- Windows functionality and, 761–762
- writing to files, 1204–1205

Windows Installer projects

- ClickOnce compared to Windows Installer, 567–568
- creating installers, 557–567
- for Enterprise Services, 1074–1075
- installing client from Web server, 566–567
- No Touch Deployment (NTD), 566–567
- options for deployment, 552
- project types available, 556
- simple client application, deployment project as
 - separate solution, 558–562
- simple client application, deployment project in same solution, 563–564
- simple Web application, 564–565
- Windows Installer overview, 556–557
- for Windows Services, 1287–1292

Windows Resource Localization Editor (winres.exe), 543

Windows Services. See also quote server application

- architecture, 1272–1275
- class library using sockets, 1276–1279
- creating as service, 1275–1292
- defined, 1271
- event logging, 1302–1308
- example services, 1272
- handler methods, 1274
- installing, 1287–1292
- interactive services, 1302
- monitoring and controlling services, 1292–1301
- namespace for, 1275
- net.exe utility, 1293–1294
- overview, 24
- performance monitoring, 1308–1313
- power events and, 1314
- project wizard for, 1281–1283
- QuoteService example, 1281–1286
- sc.exe utility, 1294–1295
- service configuration program, 1275
- Service Control Manager (SCM), 1273, 1274
- service control program, 1275
- service program, 1273–1274
- service start, 1285
- ServiceBase class, 1283–1284
- ServiceController class, 1295–1301
- service-main function, 1273–1274

- Services administration tool, 1272
- Services MMC snap-in, 1293
- TcpClient example, 1279–1281
- threading and, 1287
- troubleshooting, 1301–1313
- viewing all services on a system, 1272
- Visual Studio Server Explorer, 1295

Windows.Forms namespace, 188

winres.exe (Windows Resource Localization Editor), 543

worker threads, 352

world coordinates, 863

WorldCoordinatesToLineIndex() method, 889

WorldYCoordinateToLineIndex() method, 889

WriteAttributeInfo() method, 323

WriteMatches() method, 240

write-only properties, avoiding, 81, 94

WriteXml() method (DataSet class), 624–625, 690–691

writing to files

- binary files, 1208–1213

- overview, 1201

- text files, 1213–1219

- Windows Forms for, 1204–1205

WROX Web site, 895, 920, 1151

WS-Addressing specification, 1134

WS-AtomicTransactions specification, 1131

WS-BusinessActivity specification, 1131–1133

WS-Coordination specification, 1131

WSDL (Web Service Description Language)

- described, 983

- overview, 985–987

- type-definition section, 985–986

- viewing for Web services, 990

- as WCF contract, 1137

- as XML-compliant syntax, 985

WSDL.exe tool, 991

WS-Federation specification, 1127

WS-ReliableMessaging specification, 1129–1131

WS-SecureConversation specification, 1128–1129

WS-Security specification, 1128

X

xcopy deployment, 552, 554–555

XML. See also XML data type (SQL Server)

- ASP.NET server control syntax based on, 910

- comments, 68–70

- converting ADO.NET relational data to, 694–696

- converting ADO.NET single table data to, 689–694

- converting to ADO.NET data, 696–699

- DiffGram documents, 699–701
 - further information, 659
 - namespace for, 659, 660–661
 - Office applications supporting, 651
 - populating a DataSet from, 620
 - reading streamed XML, 664–670
 - relational data and, 651
 - .resX files for resources, 526, 527–528
 - serializing objects in, 701–711
 - standards supported in .NET, 660
 - transforming into HTML, 684–686
 - using MSXML 4.0 in .NET, 661–663
 - using the DOM in .NET, 672–678
 - writing output with DataSet, 623–625, 690–691
 - writing streamed XML, 670–672
 - WSDL compliance with, 985
 - XmlReader with ADO.NET, 593–594
 - XPathNavigators, 678–689
- XML Data Modification Language (XML DML), 654–655**
- XML data type (SQL Server)**
 - introduction of, 650
 - Office applications supporting XML, 651
 - querying data, 653–654
 - relational data and, 651
 - strongly typed XML, 656–657
 - tables with XML data, 651–653
 - XML Data Modification Language (XML DML), 654–655
 - XML indexes, 655–656
- XML DML (XML Data Modification Language), 654–655**
- XML schema definition (XSD) schemas. See XSD schemas**
- XML Web services, 23**
- XmlAttributeOverrides **class, 708–710**
- XmlAttribute **class, 708–710**
- XmlCharacterData **class, 672, 674**
- XmlDataDocument **class, 692–693**
- XmlDocument **class**
 - converting ADO.NET data to XML, 689–694
 - creating a document from scratch, 677–678
 - described, 661
 - DOM representation using, 673
 - inserting nodes, 675–678
 - using, 673–675
- XmlElement **class, 676**
- XmlLinkedNode **class**
 - classes extending, 673
 - described, 672
 - inheritance diagram for, 674
- XmlNode **class**
 - classes based on, 672
 - described, 661, 672
 - inheritance diagram for, 674
- XmlNodeReader **class, 664**
- XmlReader **class**
 - classes derived from, 664
 - Create method for, 665
 - described, 660
 - GetAttribute() method, 668
 - properties for controlling processing of nodes and values, 665
 - Read() method, 665
 - ReadElementString() method, 665–667
 - ReadStartElement() method, 665
 - ReadValueAs methods, 667
 - retrieving attribute data, 668
 - SAX versus, 664
 - simple example, 665
 - validating XML to an XSD schema, 668–670
- XmlReaderSettings **class, 665, 668, 670**
- XmlSerializer **class, 703**
- XmlTextReader **class**
 - described, 661, 664
 - EOF property, 666
- XmlTextWriter **class**
 - described, 661, 664
 - using, 670–671, 677
- XmlValidatingReader **class, 664**
- XmlWriter **class**
 - classes derived from, 664
 - described, 660
 - using, 670–671
- XmlWriterSettings **class, 670, 671**
- XPathDocument **class, 678**
- XPathException **class, 679**
- XPathExpression **class, 679, 682**
- XPathNavigator **class**
 - described, 678
 - move methods, 679–680
 - select methods, 680
 - using, 681–683
- XPathNavigators. See also specific namespaces**
 - defined, 678
 - System.Xml.XPath namespace for, 678–684
 - System.Xml.Xsl namespace for, 684–689
- XPathNodeIterator **class**
 - described, 678
 - properties and methods, 680–681
 - using, 682, 683–684

xsd command

xsd command

converting XSD files into code, 613
output for Products schema, 613–618
switches, 613

XSD (XML schema definition) schemas. See also *xsd* command

adding elements, 829
building with Visual Studio .NET, 827–832
DataColumn for, 831–832
DataRow for, 830–831
DataTable for, 829–830
EventArgs for, 832

generating from books.xml file, 668–669
overview, 612–613
Schema view versus XML view, 828
TestSchema.cs example, 828–832
validating XML to, 668–670
XmlReader class for validating, 669–670

xsd.exe tool, 702

Z

zero impact installation of private assemblies, 18