

# Index

## Symbols

!0, 288  
0 initialization, 82  
#if/#end blocks, 25  
#if/#endif blocks, 26–27, 30  
== operator, 162  
/CLR, 275–277

## A

accessing  
  native code from .NET code, 271  
  properties, 9  
accessors, 6  
AddMsg method, 133  
AllowPartiallyTrustedCallers  
  attribute, 180  
APIs, 150  
Application Data directory, 216  
ApplicationException class, 260  
applications  
  exceptions, 270  
  multiassembly, 177  
  setting storage, 217  
  storing settings, 213  
  validating user input, 224

array class, 50  
arraylike collections, 229  
ArrayList class, 229–234  
  disadvantages, 233  
  jagged, 232  
  multidimensional, 232  
arrays, 272  
as operator, 17–18  
  consistency, 21  
  using is operator instead, 22  
  when not to use, 22  
ASP.NET, 142  
  application config files, 214  
  validation, 225  
assemblies, 177  
  CLS-compliant, 181–186  
  code, 193  
  diagnostics, 212  
  loading, 177  
  multiple, 191  
  names, 180  
  partially trusted, 279  
  security checks, 193  
  server/client-side functionality, 192  
  smaller size, 190–192  
  unsafe code, 279  
  upgrading, 180  
  utility classes, 191

**AssemblyVersion**, 179

**atomic types**, 44

**attributes**

- AllowPartiallyTrustedCallers attribute, 180
- AttributeUsage attribute, 248
- Conditional attribute, 25–29
  - applying multiple*, 30
  - benefits over #if/#endif blocks*, 31
- custom, 143–144, 249
- DefaultSort attribute, 144
- DynamicCommand attribute, 249
- Serialization attribute, 148
- simplifying reflection, 246–247, 250–252
- we[WebMethod] attribute, 143

**AttributeUsage attribute**, 248

## B

**BadCastExceptions**, 23

**BadClass** objects, 114

**base classes**, 117–118, 125

- ICloneable interface, 167
- implementing interfaces, 127

**basic guarantee**, 266

**binary compatibility**, 9

**binary components**, 177–178, 253

**binding managers**, 222

**BitArray** class, 235

**blittable types**, 272

**boxing**, 68, 103–108

**Brushes** class, 101

## C

**C#**

- boxing/unboxing versus generics in C# 2.0, 287
- constraints, 288–289
- default parameters, 88
- ECMA standard, 293
- inlining, 189
- NUnit, 281
- online resources, 283

**C# 2.0**, 284–286

- generics, 285
- partial types, 291–292
- yield keyword, 291

**callbacks**, 129

**casting**

- compared with using as and is operators, 17–18
- foreach loop, 23
- user-defined types, 19

**CausesValidation** property, 227

**CheckState( )** function, 28–30

**CheckState( )** method, 26

**class keyword**, 39

**class objects**, 139

**class types**, 42–43

**classes**, 117

- collections, 235
- data binding, 2
- events, 137
- exposing functionality through interfaces, 139
- generic, 288
- public interfaces, 191
- string representation, 37

- substitutability, 167
  - using nonmemory resources, 109
  - visibility, 195
  - versus structs, 38
- clients, limiting ability to manipulate data, 140**
- Clone( ) function, 163**
- /CLR, 275–277**
- CLR (Common Language Runtime), 1**
- code access security (CAS), 278
  - loading assemblies, 177
  - P/Invoke declaration, 274
  - security, 193
- CLS**
- compliant assemblies, 181–186
  - operator overloading, 183
- CLS-compliant types, 186**
- code**
- access security, 265, 268–269
  - assemblies, 193
  - declarative programming, 147
  - don't write unless you have to, 205
  - efficiency, 186
  - integrating existing code into .NET development, 270
  - JITing, 187
  - malicious, 277
  - resurrecting objects, 114
  - testing, 281
  - unsafe, 279
- code access security, 278**
- codebase directories, 178**
- cohesion, 191**
- collections, 229**
- adding value types, 106
  - arrays, 229–234
  - type-safe, 239
  - which is best?, 229
- CollectionsBase class, 239**
- COM interop, 273**
- COM objects, 272**
- command handlers, 252**
- Common Application Data directory, 216**
- common constructor logic, filtering out, 88**
- Common Language Runtime. *See* CLR**
- communication protocols, 198**
- compacting the heap, 78**
- comparers, 144, 147**
- CompareTo( ) method, 144, 156**
- compile-time constants, 12**
- runtime compatibility, 14
  - updating values, 15
  - versions of an object in its serialized form, 15
  - versus runtime constants, 13
- Conditional attribute, 25–29**
- applying multiple, 30
  - benefits over #if/#endif blocks, 31
- conditional compilation, 26**
- config files, 181, 214–216**
- configuring data binding, 219**
- const keyword, 12, 17**
- constants, 12, 15**

- constraints, 288–289**
- constructor initializers, 92**
- constructors**
  - adding to MyClass type, 82
  - ApplicationException class, 260
  - chaining, 87–91
  - exception parameters, 262
  - initializers, 82
  - serialization, 151
- conversion operators**
  - changing conversion from implicit to explicit, 171
  - defining for types, 168
  - disadvantages, 172
- converting**
  - strings to numeric values, 223
  - types using the is operator, 23
  - value types into class types, 42–43
- creating**
  - binary components, 177–178
  - classes, 195
  - CLS-compliant assemblies, 181–184
  - conditional routines, 30
  - delegates, 130
  - exception classes, 258, 261
  - immutable types, 49
  - integer array lists, 120
  - P/Invoke declarations, 275
  - type-safe collections, 239
  - types, 56
- custom attributes, 143–144**
- custom output formats, 36**

## D

- data binding, 2, 217–221**
  - binding managers, 222
  - DataSets and DataGrids, 223
  - indexers, 7
- data members**
  - performance, 11
  - versus properties, 9
- DataBinding code, 2**
- DataGrids, 223**
- DataSet class, 77**
- DataSets, 223, 237–246**
- DataManager, 78**
- deallocation of resources, 80**
- DEBUG environment variable, 30**
- Debug.Assert method, 28**
- debugging**
  - #if/#end blocks, 25
  - #if/#endif blocks, 27
- declarative programming, 142–143**
  - avoiding repetitive code, 147
  - custom attributes, 143–144
- declaring**
  - indexers, 8
  - interfaces in non CLS-compliant assemblies, 185
  - partial types, 292
  - runtime constants, 12
  - variables natural in C#, 82
- default parameters, 88**
- DefaultSort attribute, 144**

**defining**

- APIs for a class, 121
- conversion operators for types, 168
- delegates, 130
- equality, 56
- outgoing interfaces with events, 131–135

**delegates**

- creating, 130
- type-safe callback definitions, 129

**derived classes, 126, 155****derived types, 49****deserialize method, 149****designs, 117****diagnostics, 209, 213**

- for each assembly distributed in an application, 211
- runtime, 210

**dictionary-based collections, 234****disposable objects, 95–96****Dispose( ) method, 93–94, 110, 113**

- cautions, 99
- throwing exceptions, 269

**disposing objects, 98–99****DynamicCommand attribute, 249****E****ECMA standard, 293****efficiency, 186, 198****enregistration, 188****enums, 0 as a valid choice, 52–54****equality, 62**

- defining, 56
- hash codes, 66

**Equals( ) method, 162**

- creating instances of, 56
- exceptions, 60
- overriding, 59, 62–63

**event handlers, 205–207****events**

- creating exception classes, 258, 261
- defining outgoing interfaces, 131–135
- event handlers, 137, 207
- reason for different classes, 260
- when to use, 260

**exception translation, 263****exceptions**

- guarantees, 266–269
- in wake of, 265

**explicit constructors, 53****expressing designs, 117****F****factory methods, 51****file system security, 279****finalizers, 79–80, 109**

- BadClass objects, 114
- throwing exceptions, 269

**flags, 54****flexibility of foreach loops, 74****foreach loops, 23, 70–74****framework library, 205**

freeing disposable objects, 98

#### functions

- CheckState( ) function, 28–30
- Clone( ) function, 163
- dependent on more than one
  - environment variable, 30
- disposable objects, 95
- GetType( ) function, 24
- isolating with Conditional attribute, 27
- static factory functions, 254
- UseCollection( ) function, 24
- virtual functions
  - impure*, 128
  - overriding*, 125

FXCop, 282

## G

GAC (Global Assembly Cache), 178

GC (Garbage Collector), 77

- finalizers, 80
- maximizing space, 78
- minimizing garbage, 100–102
- responsibility, 78

generic comparer, 147

generics, 285–287

GetEnumerator( ) method, 75

GetFormat( ) method, 37

GetHashCode( ) method, 63–67

GetList( ) method, 122

GetType( ) function, 24

Global Assembly Cache (GAC), 178

GotDotNet, 282

## H

#### handling

- data between managed and unmanaged layers, 273
- exceptions, 266–269

hash-based containers, 229

hash codes, 64–66

hashtable keys, 64

heap, compacting, 78

## I

IBindingList interface, 246

ICloneable interface, 163

- defining a protected copy
  - constructor, 166
  - reference type support, 164

ICollection interface, 229

IComparable interface

- CompareTo( ) method, 156
- implementing, 159

IComparer interface, 161–162

ICustomFormatter interface, 36

ICustomFormatter.Format( )  
method, 37

IDisposable interface

- Dispose( ) method, 93, 110, 113
- implementing, 110

IEnumerable interface, 236

IEnumerable interface, 75

#if/#end blocks, 25

#if/#endif blocks, 26–27, 30

- IFormatProvider interface**, 35
- IFormattable interface**, 35
- IFormattable.ToString( ) method**, 31–35
- IldAsm**, 282–283
- immutable types**, 44–46
  - creating, 49
  - modifying states, 48
- imperative programming**, 142
- implementing**
  - IComparable interface, 159
  - dispose pattern, 111
  - IDisposable interface, 110
  - IFormatProvider interface, 35
  - IFormattable interface, 35
  - interfaces, 122, 125–128
  - Object.Equals( ) method, 57–59
  - properties, 3
  - singleton pattern, 85
- IMyInterface**, 40
- indexers**, 7
  - multidimensional, 7, 235
  - this keyword, 8
- initializations**, 83
- initializers**, 82
  - as alternative to static constructors, 85
  - constructor initializers, 92
  - exceptions, 84
- initializing**
  - static member variables, 84–86
  - types that contain references, 54
- inlining**, 189
- interface methods**, 127
- interfaces**, 117–118
  - as alternatives to reflection, 256
  - CLS compliance, 184
  - const versions, 4
  - creating on value types, 108
  - defining
    - APIs for a class, 121
    - outgoing interfaces with events, 131–135
  - implementing, 125–128
  - nonconst versions, 4
  - as parameters and return values, 121
  - saving unboxing penalties for
    - structs, 124
  - web method interfaces, 198–202
  - what you can't do, 118
- internal classes**, 196
- internal state changes**, 46
- interop**, 270
  - /CLR, 277
  - performance costs, 271–274
- IntList class**, 120
- ISerializable interface**, 150
  - adding support, 151
  - downside, 153
- is operator**, 17–18
  - converting types, 23
  - when to use, 22
- isolated storage**, 280
- iterators**, 232, 289–291

**J–L**

Java reference types, 39

jagged arrays, 232

JIT compiler, 187–189

enrollment, 188

inlining property accessors, 11

Local Application Data directory, 216

loops, 70–74

**M**

malicious code, 277

marshalling strings, 273

memory

managing, 81

releasing, 78

unmanaged access, 278

methods

AddMsg method, 133

CheckState( ) method, 26

CompareTo( ) method, 144, 156

Debug.Assert method, 28

deserialize method, 149

Dispose( ) method, 93–94, 110, 113

*cautions, 99*

*throwing exceptions, 269*

Equals( ) method, 162

*creating instances of, 56*

*exceptions, 60*

*overriding, 59, 62–63*

factory methods, 51

GetEnumerator( ) method, 75

GetFormat( ) method, 37

GetHashCode( ) method, 63–67

GetList( ) method, 122

ICustomFormatter.Format( )  
method, 37

IFormattable.ToString( ) method,  
31–35

implementation, 118

inlining, 189

interface methods, 127

Object.Equals( ) method, 57

Object.GetHashCode( ) method, 65

Object.ReferenceEquals( ) method, 57

OnPaint( ) method, 100

operator== ( ) method, 56, 63

ReferenceEquals( ) method, 56

serialize method, 149

static Equals( ) method, 58

static ReferenceEquals( ) method, 58

System.Object.GetHashCode( )  
method, 64

System.Object.ToString( ) method,  
31–32

ToString( ) method, 105

Trace.WriteLine method, 28

ValueType.Equals( ) method, 58

versus properties, 3

**minimizing**

garbage, 100–102

performance costs of interop, 271–274

**modifying states of types, 48**

**MSIL, 10**

**multiassembly applications, 177**

**multicast delegates, 129**

**multidimensional arrays, 73, 231**

**multidimensional indexers, 7, 235**

**mutable reference types, 50–51**



**MyClass type**

- adding constructors, 82
- creating, 83

**MyType objects**

- converting from SecondType, 20
- NewType classes, 24
- writing functions usable with all object instances, 24

**N**

name property, 10

naming assemblies, 180

**.NET**

- assemblies, 177
- boxing, 103–108
- efficiency pitfalls, 187
- interop, 273
- isolated storage, 280
- malicious code, 277
- resource management, 77
- security, 278
- serialization, 148
- unboxing, 103–108

.NET FCL, 35

**.NET Framework**

- collections, 229–234
- data binding, 218–223
- diagnostics, 209–213
- framework library, 205
- hash codes, 64
- IldASM, 282–283
- properties, 2
- serializing objects, 156
- validations, 224–228
- value types versus reference types, 38

.NET Remoting, 198

new modifier, 172–175

NewType classes, 24

no-throw guarantee, 266, 269

null, converting using casts, 18

NUnit, 281

**O**

Object.Equals( ) method, 57

Object.GetHashCode( ) method, 65

Object.ReferenceEquals( ) method, 57

**objects**

- behavior, 40–42
- checking conditions, 25
- converting SecondType to MyType, 20
- determining equality, 56
- disposable, 94–95
- disposing
  - cautions, 99
  - resource leaks, 98
- finalizers, 109
- hash values, 64–66
- initialization, 92
- internal state changes, 46
- name modification, 70
- resurrected, 113
- state changes, 44
- type conversion, 18

OnPaint( ) method, 100

operator overloading, 183

operator== ( ) method, 56, 63

operators, 21

**optimization, 270**  
**op\_add, 183**  
**op\_equals, 183**  
**ordering relationships, 156–157**  
    IComparable interface, 159  
    IComparer Interface, 161–162  
**overloading operators, 183**  
**overriding**  
    Equals( ) method, 62–63  
    Object.ToString( ) method, 35  
    operator== ( ) method, 56  
    relational operators, 159  
    ToString( ) method, 32  
    virtual functions, 125  
    versus event handlers, 205–207

**P**  
**P/Invoke, 274–275**  
**parameterless constructors, 289**  
**partial keyword, 292**  
**partial types (C# 2.0), 291–292**  
**patterns, 111**  
**performance**  
    CompareTo( ) method, 157  
    costs associated with interop, 271–274  
    DataSets, 237  
    finalizers, 80  
    Garbage Collector, 100  
    pitfalls in .NET, 187  
    properties versus data members, 11  
    readonly keyword versus const  
        keyword, 17  
**persistence, 148**

**polymorphism, 168**  
**pragmas, 26**  
**properties, 1**  
    accessors, 6  
    indexers, 7  
    name, 10  
    NET.NET Framework, 2  
    performance, 11  
    returning reference types, 137–140  
    syntax, 7  
    translating definitions into MSIL, 10  
    versus data members, 9  
    versus methods, 3  
    versus public data members, 3  
**property directive, 10**  
**public data members versus**  
    **properties, 3**  
**public interfaces**  
    class visibility, 195  
    changing, 122

**Q–R**  
**RangeValidator, 225**  
**readonly keyword, 12, 17**  
**reference types, 38, 41**  
    array class, 50  
    equality among variables, 56  
    mutable, 50–51  
    returned by properties, 137–140  
    security concerns, 138–142  
    support for ICloneable interface, 164  
    versus value types, 41  
**ReferenceEquals( ) method, 56**

**reflection**, 246–247, 252  
    accessing data members, 255  
    alternatives, 256  
    cautions about overuse, 253  
    finding/using command handler  
        properties, 250  
    versus static factory functions, 254  
    when to use, 257

**reflexive property**, 58

**RegularExpression validator**, 225

**relational operators**, 156, 159

**releasing resources**, 93

**repetitive code**, 88

**replacing data members with  
    properties**, 8

**RequiredFieldValidator**, 225

**resource management (.NET)**, 77

**resources**  
    accessing without permissions, 278  
    deallocation, 80  
    leaks, 98  
    releasing, 93  
    writing management code, 109

**resurrected objects**, 113

**role-based security**, 278

**runtime constants**, 12  
    runtime compatibility, 14  
    updating values, 15  
    versions of an object in serialized  
        form, 15  
    versus compile-time constants, 13

**runtime diagnostics**, 209

**runtime type checking**, 17

## S

**SecondType objects, converting to  
    MyType**, 20

**security**  
    code access, 265, 268–269  
    code access (CAS), 278  
    file system, 279  
    isolated storage, 280  
    limiting user access to data  
        manipulation, 122  
    properties returning reference types,  
        137–140  
    role-based, 278  
    serialization, 153  
    validating user input (web  
        applications), 224

**serialization**, 148–151

**Serialization attribute**, 148

**serialize method**, 149

**single-dimension indexers**, 7

**singleton pattern**, 85–86

**size**  
    arrays, 231  
    types, 43

**source compatibility**, 11

**specifying custom formats**, 37

**StackTrace class**, 28

**static constructors**, 85

**static Equals( ) method**, 58

**static factory functions**, 254

**static member variables**, 84–86

**static ReferenceEquals( ) method**, 58

**storage**

- BitArray class, 235
- DataSets, 237
- information to control application
  - behavior at runtime, 214
- isolated, 280
- settings, 213, 216

**string operations, 103****StringBuilder class, 103****strings**

- converting to numeric values, 223
- marshalling, 273

**strong guarantee, 266–267****struct keyword, 39****structs**

- saving unboxing penalties, 124
- versus classes, 38

**substitutability, 167, 170****symmetric property, 58****System.Collections.CollecitonBase class, 119****System.Diagnostics.Trace class, 28****System.Object parameters, 17****System.Object.GetHashCode( ) method, 64****System.Object.ToString( ) method, 31–32****System.String class, 102****T****testing**

- code, 281
- for equality, 63

**this keyword, 8****thread safe, 44****throw statements, 260****thunking cost, 271****ToString( ) method, 105****Trace.WriteLine method, 28****TraceSwitch class, 210****transitive property, 58****translating exceptions, 263****try/finally block, 93–95****type-specific array classes, 229****types**

- atomic, 44
- blittable types, 272
- CLS-compliant, 186
- converting, 23
- converting to string instances, 105
- creating your own, 56
- defining
  - conversion operators, 168*
  - equality, 56*
- explicit constructors, 53
- exposing properties as class types, 122
- GetType( ) function, 24
- hashtable keys, 64
- immutable, 44–46, 49
- implementing interfaces, 122
- influence on behavior, 42
- limiting visibility, 194
- ordering relationships, 156–157
- partial (C# 2.0), 291–292
- persistence, 148
- serialization, 148–151
- size, 43
- thread safe, 44

- type-safe comparisons, 158
- uninitialized variables, 82
- value versus reference, 39–41
- visibility, 196

## U

- unboxing, 68, 103–108
- uninitialized variables, 82
- unmanaged code, 271
- unsafe code, 279
- upgrading assemblies, 180
- UseCollection( ) function, 24
- user-defined conversion operators, 21
- using statement, 95
- using statements
  - generating cleanup code, 96
  - try/finally blocks, 97
- utility classes, 191

## V

- validation, 224–228
- value types, 38, 41
  - adding to collections, 106
  - boxing, 104
  - containing reference types, 164
  - converting to class types, 42–43
  - creating, 43
  - creating interfaces on, 108
  - default values, 52–54
  - derived types, 49
  - equality among variables, 56
  - immutable versus atomic, 44

- overriding Equals( ) method, 59
- substituting for System.Object, 106
- versus reference types, 41

**values, determining at compile time, 15**

**ValueType.Equals( ) method, 58**

**Variables, uninitialized, 82**

**virtual functions**

- impure, 128
- overriding, 125

**visibility**

- classes, 195
- types, 196

**VS .NET diagnostics, 209**

**VS .NET Web Service wizard, 142**

## W–Z

- web applications, 224
- web method interfaces, 198–202
- web sites
  - C# Team FAQ, 283
  - GotDotNet, 282
- we[WebMethod] attribute, 143
- wrapper objects, 139
- wrapping COM objects, 272
- writing
  - constructors, 87–91
  - resource-management code, 109
  - throw statements, 260
  - try/finally blocks, 97
- XML serializer, 215**
- yield keyword, 291**

