Index

A

abbreviations, 135
abstract classes *versus* interfaces, 275–276
Abstract Factory pattern
  CreateCommand method, 365
  CreateConnection method, 365
  creation code, refactoring, 371–374
  data provider creation code, 365
  DataProviderFactory as, 371–372
  extracting creation methods to separate class, 368–370
  instantiation, 361
  multiple database engine support, 362–363
  provider object creation logic, 366–368
  split initialization from declaration refactoring, 364
  upcasting object declarations, 363–364
  variations, 375–376
abstract form, form helper classes, 343–345
abstract form inheritance, 341–343
abstract keyword, 275
abstract members, 272
abstract methods, BranchMaintenanceHelper, 351–352
AbstractAdoData class, 422–424
AbstractData class, 421–422
AbstractHelper, 343–344
abstraction, program to, 274–275
AbstractParentForm, 342
access level
  levels, 119
  reducing, 119–120
    gradual reduction, 122
AccountView example after separation of domain and persistence code, 255–256
acronyms, 135
actors, Rent-a-Wheels, 93–95
acyclic dependencies principle, 337
ad hoc unit testing, 72
AddParameter method, 205
analysis artifacts, 223–226
applications
  component-based, 105
  self-contained, *versus* reusable modules, 144–148
  tiered, 105
arrays, initializing, 403
artifacts, analysis artifacts, 223
ASP.NET
  master pages, 476–480
  single-file *versus* code-behind, 472–476
  skins, 466–467
  themes, 466–467
  user controls, 477
assemblies
  auto-wiring assembler, 381
  binary reuse
    encapsulation, 323–324
    intellectual property protection, 325
    memory, 324
    multilanguage reuse, 325
    security, 324–325
    versioning, 324
  coded assembler, 380–381
  metadata assembler, 381–382
  references, unused, 126–127
  Rent-a-Wheels, 353–355
Assert class (NUnit), 79–81
attribute-based mapping, 416
attribute classes, suffixes, 135
attribute values, HTML, 454
attributes, entity classes, 425–427
auto-implemented properties, 393–395
auto-wiring assembler, 381
automating transformations, 6–7

B

Beck, Kent, 73
behavioral patterns, 359
binary, 3
binary reuse
  encapsulation, 323–324
  intellectual property protection, 325
  memory, 324
  multilanguage reuse, 325
  security, 324–325
  versioning, 324
bottlenecks, 12
Branch class, 259–264
Branch Data class, 257–258
branch maintenance form code, 208–211
BranchData class, 259–264
  IData interface extension and, 420–421
BranchMaintenance class, 259–264
BranchMaintenance form, 352–353
BranchMaintenanceHelper implementing abstract methods, 351–352
btnChangeBranch_Click from Receive Form Event Handling Routine, 493
btnChangeBranch_Click from the FrmChangeBranch Form Event-Handling Routine, 496–497
btnRent_Click, 101
  event-handling routine, 102–103
btnSave_Click method
  decomposition, 53–55
  persistence-related code and, 50–52
bugs, duplicated code and, 175
businesses, 20
button save click event-handling code in branch maintenance form, 202–203

COPYRIGHTED MATERIAL

C

CalculateCircumferenceLength function, 163
calculating circumference
- length, function extraction, 162, 163–164, 164–165
- long method, 159–162

calculating radius, function extraction, 165–166
Calories Calculator, 23–24
- btnCalculate_Click method, 32–33
  - after method extraction, 34–35
- calories by gender, 33–34
- classes, new, 35–37
- DailyCaloriesRecommended method test, 80–81
- DistanceFromIdealWeight method, 33, 41–43
- domain classes, 408–412
- gender-specific methods, 44
- ideal weight, 27–29
- IdealBodyWeight method test, 79
- measurement struct, extracting, 407–412
- Patient class, 35–37
  - conditional logic, 39–40
  - Gender enum, 38
  - Gender property, 38
  - interface, 38–41
- patient class hierarchy, 43–48
- patient data persistence, 49–53
- patient history display, 407
- patient-history display, 57–61
- PatientHistoryXMLStorage class, 58–61
- persistent data, 30–31
- recommended daily calories, 24–27
- refactored version, 61–63
- ValidatePatientPersonalData method, 49
- weight by gender, 33–34

camel case capitalization style, 134
capitalization styles, 134–135
centralization, DI pattern, 384
change, 4–5
Change Branch Button click event, 493–497
character encoding, 447
Charge Button click event, 492–493
ChildForm, as startup object, 341–342
CIL (Common Intermediate Language), 13
Circle class, 245
- Move method refactoring and, 247

CircleCircumferenceLength class, 245–246
- object design conversion, 248–249

circumference calculation, long methods, 159–162
Class FrmChangeBranch Declaration, 494–495
Class Library Visual Studio template, 76
classes
- analysis artifacts, 223–226
- Assert, 79–81
- Branch, 259–264
- BranchData, 259–264
- BranchMaintenance, 259–264
- capitalization style, 135
- Circle class, 245
- CircleCircumferenceLength, 245–246
- code smells, 3
- complexity in code and, 8
- data class, 228
- entity class, 416
- inheritance, object composition and, 278–281
- interface inheritance and, 271–272
- large, 240
- as nouns, 226–230
- OOP, 218–219
  - constructors, 218
  - static members, 218
- partial, 339–345

clauses, guard clauses, 200
client programmers, 19
closing tags, HTML, 454
code
- commented, 111
- complexity, reasons for, 8–9
- readability, 9–10
- simplicity in, 8–9
- structured, 15–17
- transformations, 6
- unreachable, 111
- unstructured, 14–15
- unused, 111

code-behind refactoring, 476
code coverage tool, dead code and, 110
code reuse, copy and paste, 106
code smells, 2, 6
- comments, 162–163
- cyclic dependencies, 333
- data class, 228
- database-driven design, 230
- dead code, 110
- definition, 3
- document displays differently in different browsers, 439
- duplicated, 175–179
- duplicated code, 176
- event-handling blindness, 169
- fully qualified names outside using section, 123–124
- implicit imports, 321
- large class, 240
- lazy method, 173
- long method, 161–162
- magic literals, 177
- namespace, 325
- overexposure, 116
- procedural design, 244
- refused request, 282
- temporary variables, 188
  - superfluous, 192–193
- unrevealing names, 134
- unused references, 126–127
- XHTML document non-compliance, 441–442

Code Snippets (Visual Studio), 177
coded assembler, 380–381
collecting variables, 189–192
collections, initializing, 402–403
commented code, 111
- searches, 110

common misconceptions, 10–18
compilers, 13
- unused code and, 110

complexity in code, reasons for, 8–9
components, 143
- definition, 377

composition mistaken for inheritance, 281–287
computation-intensive code, IO-intensive code and, 12
concurrency, version control and, 87
conditional logic, 39–40
conditionals, 198–200
- converting to guard clauses, 200

configuration file, mapping, 416
console window, closing, 166
console.Read( ), 166
const keyword, 178
constant values, declaring, 178
constants
literal value SQL string replaced with a constant, 207
magic literals and, 178–179
containers
custom-typed container implementation using standard containers, 276–277
generic, 277
control, invisible, 114
convert procedural design to objects refactoring, 250–251
convert standard property to auto-implemented refactoring, 393–394
copy and paste, code reuse and, 106
copy-paste programming, duplicated code and, 177
CRC-cards (class-responsibility-collaborator), 231
brainstorming sessions and, 235–239
create property backing store refactoring, 394–395
CreateCommand method, 365
conditional logic, 367–368
CreateConnection method, 365
conditional logic, 367–368
creation methods, extracting to separate class, 368–370
creational patterns, 359
CRUD persistence pattern, Rent-a-Wheels, 389
CSS (cascading style sheets), 442–443
extract presentational markup to, 461–463
skins, 466–467
themes, 466–467
custom-typed container implementation using standard containers, 276–277
cyclic dependencies, 333
GUI and faxing service namespaces, 334, 335–337

D

DailyCaloriesRecommended method, 80
data class, 228
domain logic, 253
persistence logic, 253–254
replace row with, 241–244
Data class, newly defined Branch Data class, 257–258
data providers, creation code, extracting as method, 365
database
categories, 98
engine, 98
records, deleting, 361–362
relational database design, 104
tables, 98
database-driven design, 104, 230
DatabaseTime method, 399
DataProviderFactory as Abstract Factory, 371–372
DataProviderFactory with provider-related variables, 369–370
dead code, 109–110
code coverage tools and, 110
definition, 110
eliminating, 112–114
flavors, 111–112
sources, 110, 112–116
types of, 111–112
declarations
refactoring, initialization and, 187–188
split initialization from declaration refactoring, 364
declaring, temporary variables, location, 184–187
decomposing, 116
methods, 159–171
circumference calculation, 159–162
DeleteAllData utility, 399
dependencies, 114, 124–126
acyclic dependencies principle, 337
breaking cycles, 333–335
build process and, 330–333
cyclic, 333
distribution and, 330–333
inverting, 335–337
testability and, 330–333
Dependency Injection (DI) pattern, 376
auto-wiring assembler, 381
benefits of, 382–385
coded assembler, 380–381
component containers, 383–384
constructor-based, 380
containers, 383–384
heavyweight containers, 383–384
IoC (Inversion of Control), 376
lightweight containers, 383–384
metadata assembler, 381–382
modular architecture, 384
POCO (Plain Old CLR Object), 383–384
problem using, 376–379
property-based, 380
Rent-a-Wheels, 386–389
design, 2, 4
classes, analysis artifacts, 223–226
database-driven, 104, 230
design rot, 5
errors, 120–122
reuse, 359
design patterns, 357
Abstract Factory, 361
benefits of, 360
defining, 358–359
detached event handler, 114
Display button click event-handling routine, 505–509
DisplayCurrentRow method, 204
DOCTYPE declaration, 456
document presentation, 460–467
document type declaration, 440
DOM (Document Object Model), 444
Domain class, Vehicle Data class becomes Domain class, 258–259
domain classes, Calories Calculator, 408–412
domain code
data class, 252–253
logic, moving inside data class, 253
separating from persistence code, 254–257
separating from presentation code, 252–254, 258–259
downlevel browsers, 449

DTD (Document Type Definition), 440
compliance level, 456
Visual Studio validation for HTML, 448–449
duplicated code
bugs and, 175
code smell, 176
copy-paste programming and, 177
maintainability of code and, 175
methods and, 175
sources of, 176
duplicated code smell, 175–179
duplication
elimination by pulling up members, 301–308
elimination with inheritance, 295–301

E

ECMAScript, 444
eliminate dead code refactoring, 112–114
embedding, DI pattern, 384
encapsulation, 138–139, 155–156
assemblies, binary reuse, 323–324
encapsulate field refactoring (Visual Studio), 216–217
object orientation and, 116
objects (OOP), 214–216
encoding, XML and, 447–448
entities, 239–240
entity class, 416
attributes, 425–427
enums, capitalization style, 135
Equals method, 219–220
event arguments, suffixes, 135
event-driven programming, 105
event handlers
button save click event-handling code in branch maintenance form, 202–203
detached, 114
navigational button event handlers, 204
suffixes, 135
event handling, routine to delete database record, 361–362
event-handling blindness, 169
events, capitalization style, 135
exception classes, suffixes, 135
exception handling, program class with global exception-handling code, 206–207
exceptions, 82–84
excess of structure, 173
ExecuteNonQuery method, 205
ExpectedException attribute, 82–84
explaining temporary variables, 197–198
explicit imports, 123–126, 322
exposed elements
access level, reducing, 116–123
scope, reducing, 116–123
extensibility
fields, properties and, 150
interfaces, abstract classes, 150
extension methods, 395–402
extension wrapper, 399–402
Extract Class refactoring, 223
extract common content to master page refactoring, 477–480
extract interface, 292–294
extract method, 169–171
refactoring, 157–159, 193
Rent-a-Wheels, 179–180
extract namespace, 331–333
refactoring, 331–333
extract presentational markup to CSS refactoring, 461–463
extract style refactoring, 463–464
extract superclass refactoring, 298–301
extract user control refactoring, 481–484
extracting methods, 169–171
local variables and, 184

F

factories
MsSqlProviderFactory, 372–373
OleDbProviderFactory, 373
OracleProviderFactory, 373
Fiddler, 472
fields, properties and, 150
FileNotFoundException, 52
FillDataset method, 206
flavors of dead code, 111–112
form helper classes, abstract forms, 343–345
forms, abstract form inheritance, 341–343
fragile base class problem, 281
frames, 476
FrmChangeBranch_Load event-handling routine, 495–496
from maintenance, 497

G

Gamma, Erich, 73
garbage collection, 221
messages, 222–223
reference counting, 221–222
tracing, 222
GeneralMaintenanceForm code, 347–348
generic containers, 277
generic types, 276
generics, 308–312
Rent-a-Wheels, 312–318
GET method, 467, 468–471
GetTable method, 416
GoF (Gang of Four) design patterns, 359
guard clauses, 200
GUI automation code, separation of from database code, 207
GUI-based application, 104–105
GUI controls, 480

H

Hand Over button click event, 491
Helper interface code, 349–351
hidden classes, 240–265
hiding, overriding and, 151
hierarchy, upcasting object declarations, 363–364
HTML (Hypertext Markup Language)
attribute values, 454
closing tags, 454
CSS (cascading style sheets), 442–443
DOM, 444
ECMAScript, 444
history of, 438–439
legacy, 458–459
overlapping elements, 454
printing documents, 459–460
quirks mode, 441
REST, 444–446

upgrade to valid strict XHTML, 457–458
upgrade to well-formed XML, 454–456
XHTML, well-formed documents, 454–456
HTML Tidy, 459
HTTP (Hypertext Transfer Protocol)
Fiddler and, 472
REST and, 467–472
Hungarian notation, 136

I

IData interface, 419–420
BranchData class and, 420–421
IdealBodyWeight method, 79
identifiers, name source, 137
ignored return parameter, 115
ignored return value, 115
implicit imports, 321
importing
explicit imports, 123–126, 322
implicit imports, 321
unused elements, 114
information and implementation hiding, 116, 156–159
inheritance, 268–271
abstract form inheritance, 341–343
composition mistaken for, 281–287
duplication, eliminating, 295–301
hierarchies, name choices, 139–140
interface inheritance, 271
‘‘is a’’ principle, 282
multiple, 272
parent class child class link, 277–278
print system, 288–294, 301–303
delegation instead, 294–308
refactoring for, 288–294
relationships, 282
Rent-a-Wheels, 312–318
subclass and, 271
superclass and, 271
Inherited Form (Visual Studio), 340–341
initialization, split, 364
inline code, move to code-behind in ASP.NET page, 473–475
inline method refactoring, 173–175
inline temp refactoring, 193–194
inlining methods, 171–175
input point coordinates extraction, 166–169
instance public fields, capitalization style, 135
instantiation
Abstract Factory, 361
interfaces, 272
intellectual property protection, 325
IntelliJ, 66
interface inheritance, 271
classes and, 271–272
interfaces
versus abstract class, 275–276
abstract classes, 150
abstract members, 272
capitalization style, 135
IData, 419–420
identifiers, prefixes, 135
implementation, 272
instantiation, 272
OOP, 214
persistence layer, extracting, 419–425
public, 143–153
published, 143–153
Internet Media (MIME) type, 441
introduce explaining temporary variable refactoring, 198
inverting dependencies, 335–337
invisible control, 114
IO-intensive code, computation-intensive code and, 12
IoC (Inversion of Control), 376
IPrintDevice interface, extracted, 291

J

JIT (Just-in-Time) compilation, 13
JUnit, NUnit and, 73

K

keywords
abstract, 275
const, 178
out, 169, 170
ref, 169, 170
var, 392

L

large class, 240
lazy method code smell, 173
legacy HTML, 458–459
lifetime of objects (OOP), 221–223
LINQ (Language Integrated Query), 392
indirection, 406
object-relational mapping, 414–418
query example, 404–405
querying objects, 404–406
Rent-a-Wheels, 418–427
to SQL, 415–418
LINQ-to-SQL persistence class, 419
LinqData class, 424–425
literal value SQL string replaced with a constant, 207
literals, magic literals, 178–179
local variables, 184
capitalization style, 135
not read, 115
type inference, 391
upcasting object declarations, 363–364
logic, conditional, 39–40
looping variables, 189–192

M

magic literals, 177, 178–179
Rent-a-Wheels, 179–180
maintenance
From Maintenance, 497
To Maintenance, 497
maintenance programmers, 19
manual data entry in unit testing, 71–72
mapping, attribute-based, 416
mapping configuration file, 416
master pages (ASP.NET), 476–480
memory
assemblies, binary reuse and, 324
garbage collection, 221
messages, garbage collection, 222–223
metadata assembler, 381–382

methods
AddParameter, 205
capitalization style, 135
code smells, 3
complexity in code and, 8
DailyCaloriesRecommended, 80
DatabaseTime, 399
decomposition, 53
DisplayCurrentRow, 204
duplicated code and, 175
Equals, 219–220
ExecuteNonQuery, 205
extension methods, 395–402
extracting, 169–171
local variables and, 184
FillDataset, 206
GET, 467, 468–471
GetTable, 416
IdealBodyWeight, 79
inlining, 171–175
POST, 467, 468–471
PrepareDataObjects, 205
querying as, 195
related to behavior, 35
reorganization heuristics, 197
separating into smaller methods, 32
SplitReturningDelimiter, 397–398
misconceptions, debunking, 10–18
mock objects, 86
modular architecture, DI pattern, 384
module globals, 367
move class to namespace refactoring, 328–329
move declaration near reference refactoring, 185–187
move element to more enclosing region refactoring, 121–122
Move Field, 233–235
move initialization to declaration refactoring, 187–188
move inline code to code-behind in the ASP.NET page refactoring, 473–475
Move method, refactoring, 231–233
Circle class and, 247
move type to file refactoring, 338–339
MsSqlProviderFactory factory, 372–373
multiple inheritance, 272

N
namespaces, 319–320
capitalization style, 135
code smell, 325
default name, 321
extract, 331–333
move class to, 328–329
naming guidelines, 320
nested, 320
organization, 320, 325–330
maintainability, 326
reuse, 326–329
Rent-a-Wheels, 345
using directives, 321–323
naming guidelines, 129, 133–134
abbreviations, 135
acronyms, 135
capitalization styles, 134–135
Hungarian notation, 136
inheritance hierarchies, 139–140
namespaces, 320
spell-check, 135
suffixes, 135
unrevealing names, 134
word choice, 136–140
navigational button event handlers, 204
NCover, 86
nested namespaces, 320
NUnit framework, 73–74
Assert class, 79–81
asserts, 79–81
Class Library Visual Studio template, 76
color legend, 76
exceptions, 82–84
ExpectedException attribute, 82–84
installing, 74
JUnit and, 73
projects, new, 76
reusable features, 73
Run button, 74
samples, 74–76
Setup attribute, 81–82
Stop button, 74
TearDown attribute, 82
tests, 18
text fixtures, 76–78
writing tests, 78–79
NUnitForms GUI-Testing framework, 86

O
object composition, class inheritance and, 278–281
object design, CircleCircumferenceLength conversion, 248–249
object-mocking frameworks, 86
object orientation, encapsulation and, 116
object-oriented analysis, 129
object-oriented design, 17, 129
refactoring and, 17
object-relational impedance mismatch, 239, 414–415
objects
convert procedural design from, 250–251
declarations, upcasting, 363–364
initializing, 402–403
mock objects, 86
querying (LINQ), 404–406
objects (OOP), 214
as basic building blocks, 220
encapsulation and, 214–216
garbage collection, 221
messages, 222–223
reference, 221–222
tracing, 222
identity, 219–220
lifetime of, 221–223
root object, 221
state, retention, 218
obsolete elements, 116
OleDbProviderFactory factory, 373
OOP (object-oriented programming)
classes, 218–219
static members, 218
entities, 239–240
inheritance, 268–271
interfaces, 214
objects, 214
encapsulation, 214–216
garbage collection, 221–223
identity, 219–220
lifetime of, 221–223
root object, 221
state retention, 218
polymorphism, 273–276
relationships, 239–240
SRP (Single Responsibility Principle), 236–238

open-closed principle, 151–153
operations, as verbs, 226–230
optimization, 13
OracleProviderFactory factory, 373
organizing namespaces, 320
ORM (object-relational mapping) framework, 414–415
out keyword, 169, 170
overexposure, 116, 122–123
sources, 119–123
overlapping elements, HTML, 454
overriding, hiding and, 151
OvertimeIndex, 6

P

parameterized types, 276
parameters, capitalization style, 135
parent maintenance form, extracting, 345–347
partial classes, 339–345
Inherited Form, 340–341
Pascal case capitalization style, 134
PatientHistoryXMLStorage class, 58–61
patterns. ***See also*** **design patterns**
behavioral, 359
classifying, 359
creational, 359
elements
consequences, 360
name, 359
problem, 360
solution, 360
structural, 359
using, 360
peer programmers, 19
performance, 12–17
bottlenecks, 12
experimenting with, 13
refactoring and, 12
persistence
AccountView example after separation of domain and persistence code, 255–256
btnSave_Click method, 50–52
logic, moving inside data class, 253–254
.NET serialization, 412–414
ORM (object-relational mapping), 414
separation from presentation code, 259–265
XML file, 49–53
persistence code, separating domain code, 252–254
persistence layer interfaces extracting, 419–425
POCO (Plain Old CLR Object), 382, 383–384
polymorphism, 273–276
POST method, 467, 468–471
PrepareDataObjects method, 205
presentation code, separating domain code, 252–254
presentational markup, 438
pretty-print XHTML refactoring, 459–460
primary key, auto-incrementing, 98
print system, inheritance, 288–294, 301–303
print system, inheritance,
delegation instead, 294–308
PrintDevice abstract superclass, 295–297
printing, HTML documents, 459–460
private instance fields, capitalization style, 135
problem domain, 130
information gathering and, 131
interactions, designing, 132–133
prototype, building, 133
vocabulary, 131–132
identifier names source, 137
procedural design, 244–251
convert to objects, 250–251
Program class, as DI assembler, 386–389
program class with global exception-handling code, 206–207
program to an abstraction, 274–275
programmers
client, 19
maintenance, 19
peer, 19
programming
database-driven design, 104
event-driven, 105
properties
auto-implemented, 393–395
capitalization style, 135
querying as, 195
protected fields, capitalization style, 135
provider objects, creation logic, 366–368
public interfaces, 144–153
published interfaces, 144–153
definition, 145
modifying, 148–153
pull-down method refactoring, 44
pull-up method refactoring, 304–308
pulling up members, eliminating duplication, 301–308

Q

queries
LINQ, 404–405, 404–406
syntax, 404
query refactoring, temporary variables, 194–197
querying
as method, 195
as property, 195
quirks mode, 441

R

RAD (rapid application development), 105
radius, calculating, function extraction, 165–166
read coordinates code, extracting, 166–169
read-only property
get property, 115
set property, 115
readability of code, 9–10
Receive Button click event, 492
records (database), deleting, 361–362
reduce access level refactoring, 119–120
ref keyword, 169, 170
Refactor! for ASP (Developer Express), 68, 430
interface, 433–434
linked identifiers, 435–436
markers, 434–435
replace progress indicator, 436–438
target pickers, 436
invoking
cut and paste, 432–433

Refactor! for ASP (Developer Express) *(continued)*
keyboard shortcut, 432
mouse, 431
Smart Tags, 431
Refactor! for ASP.NET (Developer Express), 511–513
extract style refactoring, 463–464
rename style refactoring, 464–466
user controls, 481–484
Refactor! Pro (Developer Express), 67
refactoring
automating transformations, 6–7
benefits, 8–10
business people, 20
C# and, 21
code-behind, 476
convert procedural design to objects, 250–251
convert standard property to auto-implemented, 393–394
create property backing store, 394–395
definition, 2
to DI, 385
encapsulate field, Visual Studio, 216–217
explaining temporary variables, 198
explicit imports, 322
Extract Class, 224–226
Extract Class refactoring, 223
extract common content to master page, 477–480
extract interface, 292–294
extract method, 157–159
extract namespace, 331–333
extract presentational markup to CSS, 461–463
extract style refactoring, 463–464
extract superclass, 298–301
extract user control, 481–484
for inheritance, 288–294
initialization move to declaration, 187–188
inline method, 173–175
inline temp, 193–194
move class to namespace, 328–329
move element to a more enclosing region, 121
Move Field, 233–235
move inline code to code-behind in the ASP.NET page, 473–475
Move method, 231–233
move type to file, 338–339
object-oriented design and, 17
performance and, 12
pretty-print XHTML, 459–460
process, 2–3
pull up method, 304–308
remove unused references, 127
rename, 140–142
Visual Studio and, 142–143
rename style refactoring, 464–466
replace complex imperative C# query code with LINQ, 406
replace extension wrapper with extension method, 400–402
replace fully qualified names with explicit imports, 124–125
replace general-purpose reference with parameter type, 309–312
replace GET with POST, 468–471
replace inheritance with delegation, 283–287
replace programmatic data layer with LINQ to SQL, 417–418
replace row with data class, 241–244
replacing nested conditionals with guard clause, 200–201
replacing temp variable with query, 195–196
safe refactorings, 149–150
safe rename, 146–148
split initialization from declaration refactoring, 364
split temporary variables, 188–192, 190–192
techniques, 62
upgrade HTML markup to valid strict XHTML, 457–458
upgrade HTML markup to well-formed XML, 454–456
variable declaration near reference, 185–187
reference counting garbage collector, 221–222
references
removing unused, 126–127
variable declarations and, 185–187
relational database design, 104
relationships, 239–240
remove unused references refactoring, 127
rename refactoring, 140–142
Visual Studio and, 142–143
rename style refactoring, 464–466
renaming, safe rename, 146–148
Rent-a-Wheels
actors, 93–95
ASP.NET refactorings, 486–489
assemblies, reorganization, 353–355
C# code, 100–104
client interviews
desk receptionist, 91
maintenance personnel, 92–93
manager, 90–91
parking lot attendant, 91–92
CRUD persistence pattern, 389
database model, 98–100
tinyint values, 99–100
Dependency Injection, 386–389
duplication, removing, 203–211
extract method, 179–180
generic types, 312–318
hygiene, 127–128
inheritance, 312–318
introduction, 89
LINQ and, 418–427
magic literals, 179–180
main application window, 97
method reorganization, 201–211
namespaces, 345
reorganization, 353–355
.NET Framework, duplication, eliminating, 385
objects, 257–265
parent maintenance form, extracting, 345–347
patterns, 385–389
prototype, 98–104
refactoring to patterns, 385–389
rename refactoring, 153–154
safe rename refactoring, 153–154
team meeting, 97–98
use cases, 93–95
vehicle states, 95–97
in operation, 96
super state, 96

vocabulary, 132
Windows Designer problems, resolving, 347–353
replace complex imperative C# query code with LINQ refactoring, 406
replace extension wrapper with extension method refactoring, 400–402
replace fully qualified names with explicit imports refactoring, 124–125
replace general-purpose reference with parameter type refactoring, 309–312
replace GET with POST refactoring, 468–471
replace inheritance with delegation refactoring, 283–287
replace magic literal with constant refactoring, 178–179
replace nested conditional with guard clause refactoring, 200–201
replace programmatic data layer with LINQ to SQL refactoring, 417–418
replace row with Data class refactoring, 241–244
replace temp with query refactoring, 195–196
ReSharper (JetBrains), 66–67
REST (Representational State Transfer), 444–446
HTTP and, 467–472
RestoreDatabaseData utility, 399
return parameter, ignored, 115
return value, ignored, 115
reusable modules, *versus* self-contained applications, 144–148
reuse-release equivalence, 326–328
Rich Internet Applications, 467
root data class
AbstractAdoData, 422–424
AbstractData, 421–422
root element, XML, 454
root object (OOP), 221
rotting design, 5
row, replace with data class, 241–244
rule of least surprise, 9

S

safe refactorings, 149–150
safe rename, 146–148
sample applications, Calories Calculator, 23–24
btnCalculate_Click method, 32–33, 34–35
calories by gender, 33–34
DailyCaloriesRecommended method test, 80–81
DistanceFromIdealWeight method, 33, 41–43
gender-specific methods, 44
ideal weight, 27–29
IdealBodyWeight method test, 79
new classes, 35–37
Patient class, 35–37
patient class hierarchy, 43–48
Patient class interface, 38–41
patient data persistence, 49–53
patient-history display, 57–61
PatientHistoryXMLStorage class, 58–61
persistent data, 30–31
recommended daily calories, 24–27
refactored version, 61–63
ValidatePatientPersonalData method, 49
weight by gender, 33–34
saving data, 49–57
scope
levels, 118
reduction, gradual, 123
searches, commented code, 110
self-contained applications, *versus* reusable modules, 144–148
separation of GUI automation code from database code, 207
services, definition, 377
Setup attribute, 82
SGML, 438, 439
short-term benefits, 17
shortcut keys, 68
simplicity in code, 8–9
skins, 466–467
Smalltalk, 2
Smart Tag, 84–85
software
binary, 3
efficiency, 3
performance, 3
quality, 3
timeliness, 3
usability, 3
user requirements, 3
solution domain vocabulary, identifier name source, 137
sources of duplicated code, 176
spaghetti code, 156
spell-check, 135
split initialization from declaration refactoring, 364
split temporary variables, 190–192
SplitReturningDelimiter extension method, 397–398
SQL
LINQ to, 415–418
literal value SQL string replaced with a constant, 207
SqlConnection, enhanced, 399–400
SRP (Single Responsibility Principle), 236–238
SSI (server-side includes), 477
statements, using, 114
static methods, implementing string functionality, 396–397
static public fields, capitalization style, 135
strict XHTML, 449–451
structural markup, 438
structural patterns, 359
structured code, 15–17
subclasses, inheritance and, 271
subroutines, 156
benefits, 156
suffixes, 135
superclass
extracting, 312–313
inheritance and, 271
syntax
queries, 404
XML, 440

T

tag soup, 438
teams, 18, 19
TearDown attribute, 82
temporary variables, 183–184
code smells, 192–193
declaration, location, 184–187
eliminating, 192–194

temporary variables *(continued)*
explaining temporary variables, 197–198
query refactoring, 194–197
split, 188–192, 190–192
test-driven approach, 84–85
TestDriven.NET (Visual Studio), 85
testing, DI pattern, 384
tests, unit tests, 2
working without, 18
themes, 466–467
Tim (apprentice programmer), 89
tinyint values, 99–100
to maintenance, 497
tracing garbage collector, 222
transformations, 6
automating, 6–7
refactoring, 2
type inference, 392
local variables, 391
type safety, variables, 391
typed container problem, 276–277

U
Unicode, 447
unit testing, 70. *See also* NUnit framework
ad hoc unit testing, 72
manual data entry, 71–72
NCover, 86
NUnit, 73–74
installing, 74
NUnitForms GUI-Testing framework, 86
object-mocking frameworks, 86
test-driven approach, 84–85
test fixture, 76–78
test project, creating, 76
TestDriven.NET (Visual Studio), 85
unit testing frameworks, 73
writing tests, 78–79
unit tests, 2, 31
working without, 18
unreachable code, 111
unrevealing names, 134
unstructured code, 14–15
unused code, 111
compilers and, 110
unused elements, importing, 114
unused references, removing, 126–127
upcasting object declarations, 363–364
upgrade HTML markup to valid strict XHTML refactoring, 457–458
upgrade HTML markup to well-formed XML refactoring, 454–456
uplevel browsers, 449
uppercase capitalization style, 134, 135
use cases, Rent-a-Wheels, 93–95
user, closing console window, 166
user controls, 481–485
custom server controls, 485
user input, reading, 166
using directives, 321–323
using statements, 114

V
valid XHTML documents, 456–458
valid XML documents, 440
value semantics, 220
var keyword, 392
variables
collecting, 189–192
initializing, declaration refactoring and, 187–188
local, 184
looping, 189–192
refactoring, split temporary, 188–192
split initialization from declaration refactoring, 364
temporary, 183–184
declaration location, 184–187
eliminating, 192–194
explaining, 197–198
query refactoring, 194–197
type safety, 391
upcasting, 364
Vehicle Data class becomes Domain class, 258–259
Vehicle Fleet Administration form
Delete button, 498–499
fields, 501–502
form load event-handling routine, 499–501
navigation buttons, 502–503
New button, 499
Reload button, 499
save button, 503–505
_VehiclesAndRates class code, 486–487
VehiclesAndRates.aspx code, 488
version control
as backup system, 87
concurrency and, 87
versioning, 324
versioning policies, 149
Visual Studio
Code Snippets, 177
DTD validation for HTML, 448–449
Encapsulate Field refactoring, 70
encapsulate field refactoring, 216–217
Extract Interface refactoring, 70
Extract Method refactoring, 70
Inherited Form, 340–341
Promote Local Variable to Parameter refactoring, 70
refactoring features, 68–70
Remove and Sort (Usings), 70
Remove Parameters refactoring, 70
Remove Unused Usings, 70
Rename refactoring, 70
rename refactoring, 142–143
Reorder Parameters, 70
Smart Tag, 84–85
Sort Usings, 70
XHTML and, 446–447
vocabulary document, 132

W
W3C (World Wide Web Consortium), 444
WCF (Windows Communication Foundation) framework, 467
Web Content Form, 477–480
well-formed XML documents, 440
window, console, closing, 166
Windows Forms, 104
Rent-a-Wheels, 345
Windows Forms Designer, 105
word choice in naming, 136–140
World Wide Web Consortium (W3C), 444
wrappers, extension wrapper, 399–402
write-only property
get property, 115
set property, 115

X

XHTML, 439–442
  document type declaration, 440
  DTD (Document Type Definition), 440
  namespace declaration, 457
  printing, 459–460
  strict, 449–451
  valid strict, upgrading to, 457–458
  validity, 456–458
  Visual Studio and, 446–447
  well-formed documents, 454–456

XML, 439
  encoding and, 447–448
  root element, 454
  syntax, 440
  valid documents, 440
  validity, 440
  well-formed documents, 440
    upgrade to, 454–456