eaa.book Page 517 Monday, October 7, 2002 8:23 AM

Index

A
ACID (atomicity, consistency, isolation, and durability), 71–76
  business and system transactions, 74–76
  reducing transaction isolation for liveness, 73–74
  transactional resources, 72–73
Active Record, 160–64
  example
    simple person (Java), 162–64
  how it works, 160–61
  when to use it, 161–62
ADO.NET data sets, 148–51
Advice, some technology-specific, 100–103
  Java and J2EE, 100–101
  .NET, 101–2
  stored procedures, 102–3
  Web services, 103
Affinity, server, 85
Albums and tracks (Java), 264–67
Albums, transferring information about, 407–11
Application Controller, 379–86
  example
    state model Application Controller (Java), 382–86
  further reading, 382
  how it works, 380–81
  when to use it, 381–82
Application Controller, state model, 382–86
Application server concurrency, 78–80
Applications, evolution of layers in enterprise, 18–19
Architectural patterns, 33–38
Architectural patterns, data source, 143–81
  Active Record, 160–64
  Data Mapper, 165–81
  Row Data Gateway, 152–59
  Table Data Gateway, 144–51
ASP.NET server page (C#), 357–60
Association Table Mapping, 248–61
  examples
    employees and skills (C#), 250–53
    using direct SQL (Java), 253–56
    using single query for multiple employees (Java), 256–61
  how it works, 248–49
  when to use it, 249

B
Base patterns, 465–510
  Gateway, 466–72
  Layer Supertype, 475
  Mapper, 473–74
  Money, 488–95
  Plugin, 499–503
  Record Set, 508–10
  Registry, 480–85
  Separated Interface, 476–79
  Service Stub, 504–7
  Special Case, 496–98
  Value Object, 486–87

Behavioral patterns, object-relational, 183–214
  Identity Map, 195–99
  Lazy Load, 200–214
  Unit of Work, 184–94
Behavioral problem, 38–39
Boosters, complexity, 24
Boundaries, working with distribution, 91–92
Brown layers, 104
Business and system transactions, 74–76
Business logic, 20

C
C#
  ASP.NET server page, 357–60
  collection of references, 244–47
  concrete players, 296–301
  employees and skills, 250–53
  integral key, 221–22
  page handler with code behind, 340–43
  Person Gateway, 146–48
  players and their kin, 287–92
  revenue recognition with Table Module, 129–32
  simple null objects, 498
  single table for players, 280–81
  using ADO.NET data sets, 148–51
  using ghosts, 206–14
  Web service, 395–400
Cases, Special, 496–98
  example
    simple null objects (C#), 498
  further reading, 497
  how it works, 497
  when to use it, 497
Class, money, 491–95
Class Table Inheritance, 285–92
  example
    players and their kin (C#), 287–92
  further reading, 287
  how it works, 285–86
  when to use it, 286–87
Client Session State, 456–57
  how it works, 456–57
  when to use it, 457
Coarse-Grained Lock, 438–48
  examples
    root Optimistic Offline Lock (Java), 447–48
    shared Optimistic Offline Lock (Java), 441–46
    shared Pessimistic Offline Lock (Java), 446–47
  how it works, 438–41
  when to use it, 441
Committed, read, 73
Complexity boosters defined, 24
Compound key (Java), 224–35
Concrete players (C#), 296–301
Concrete Table Inheritance, 293–301
  example
    concrete players (C#), 296–301
  how it works, 293–95
  when to use it, 295–96
Concurrency, 63–80
  application server, 78–80
  application server concurrency, 78–80
  concurrency problems, 64–65
  execution contexts, 65–66
  isolation and immutability, 66–67
  off line, 75
  offline, 63
  optimistic and pessimistic concurrency controls, 67–71
  patterns for off line concurrency control, 76–78
Concurrency controls
  optimistic and pessimistic, 67–71
    ACID (atomicity, consistency, isolation, and durability), 71–76
    deadlocks, 70–71
    preventing inconsistent reads, 68–69
    transactions, 71
  patterns for offline, 76–78
Concurrency patterns, offline, 415–53
Connections, database, 50–52
Contexts, execution, 65–66
Controller, Front, 344–49
  example
    simple display (Java), 347–49
  further reading, 347

eaa.book Page 519 Monday, October 7, 2002 8:23 AM

how it works, 344–46
when to use it, 346
Controller, Page, 333–43
examples
page handler with code behind (C#), 340–43
simple display with JSP view (Java), 335–37
simple display with servlet controller (Java), 335–37
using JSP as handler (Java), 337–40
how it works, 333–34
when to use it, 334–35
Controller patterns, input, 61
Controllers
simple display with, 335–37
state model Application, 382–86
using JSP as view with separate, 355–57
Controllers, Application, 379–86
example
state model Application Controller (Java), 382–86
further reading, 382
how it works, 380–81
when to use it, 381–82
Controls
optimistic and pessimistic concurrency, 67–71
ACID (atomicity, consistency, isolation, and durability), 71–76
deadlocks, 70–71
preventing inconsistent reads, 68–69
transactions, 71
patterns for offline concurrency, 76–78
Correctness, 65
Custom tags, JSP and, 374–78

D
Data
immutable, 67
reading in, 40–41
record, 83
Data holder for domain object (Java), 158–59
Data Mapper, 165–81
domain layer with, 421–25
examples
creating empty objects (Java), 179–81
separating finders (Java), 176–79
simple database mapper (Java), 171–76
how it works, 165–70
when to use it, 170–71
Data sets, ADO.NET, 148–51
Data source
architectural patterns, 143–81
Active Record, 160–64
Data Mapper, 165–81
Row Data Gateway, 152–59
Table Data Gateway, 144–51
for Domain Models, 98–99
logic, 20
Table Modules, 98
for Transaction Scripts, 97–98
Data source layers, down to, 97–100
data source for Domain Models, 98–99
data source for Transaction Scripts, 97–98
data source Table Modules, 98
presentation layers, 99–100
Data Transfer Objects, 401–13
examples
serializing using XML (Java), 411–13
transferring information about albums (Java), 407–11
further reading, 407
how it works, 401–6
when to use it, 406
Database connections, 50–52
Database mapper, simple, 171–76
Database Session State, 462–64
how it works, 462–63
when to use it, 464
Databases
loading objects from, 281–84
mapping to relational, 33–53
architectural patterns, 33–38
behavioral problem, 38–39
building mapping, 47–49
database connections, 50–52

eaa.book Page 520 Monday, October 7, 2002 8:23 AM

Databases, mapping to relational, *continued*
reading in data, 40–41
some miscellaneous points, 52–53
structural mapping patterns, 41–47
using metadata, 49–50
Deadlocks, 70–71
Department hierarchy, serializing, 274–77
Dependent Mapping, 262–67
example
albums and tracks (Java), 264–67
how it works, 262–63
when to use it, 263–64
Dependents, finding person's, 325–26
Dirty reads, 74
Display, simple, 347–49
Distributed objects, allure of, 87–88
Distribution boundaries, working with, 91–92
Distribution, interfaces for, 92–93
Distribution patterns, 387–413
Data Transfer Objects, 401–13
Remote Facade, 388–400
Distribution strategies, 87–93
allure of distributed objects, 87–88
interfaces for distribution, 92–93
remote and local interfaces, 88–90
where you have to distribute, 90–91
working with distribution boundaries, 91–92
DNA layers, Microsoft, 105
Domain layer with Data Mappers (Java), 421–25
Domain layers, starting with, 96–97
Domain logic, 20
organizing, 25–32
making choices, 29–30
Service Layers, 30–32
patterns, 109–41
Domain Model, 116–24
Service Layer, 133–41
Table Module, 125–32
Transaction Script, 110–15
Domain Model, 116–24
data source for, 98–99
example
revenue recognition (Java), 120–24
further reading, 119–20
how it works, 116–19
when to use it, 119
Domain objects, data holder for, 158–59
Domain objects (Java), 475

E
EAI (Enterprise Application Integration), 468
Embedded Value, 268–71
example
simple value objects (Java), 270–71
further reading, 270
how it works, 268
when to use it, 268–69
Employees and skills (C#), 250–53
Employees, using single query for multiple, 256–61
Enterprise Application Integration (EAI), 468
Enterprise applications, evolution of layers in, 18–19
Examples
albums and tracks (Java), 264–67
ASP.NET server page (C#), 357–60
collection of references (C#), 244–47
concrete players (C#), 296–301
data holder for domain object (Java), 158–59
domain objects (Java), 475
employees and skills (C#), 250–53
finding person's dependents (Java), 325–26
gateway to proprietary messaging service (Java), 468–72
Id Generator (Java), 501–3
implicit Pessimistic Offline Lock (Java), 451–53
integral key (C#), 221–22
lazy initialization (Java), 203
methods for Identity Map (Java), 198–99
money class (Java), 491–95
multitable find (Java}, 243–44
Person Gateway (C#), 146–48
person record (Java), 155–58
players and their kin (C#), 287–92

eaa.book Page 521 Monday, October 7, 2002 8:23 AM

revenue recognition (Java), 120–24, 138–41
revenue recognition with Table Module (C#), 129–32
root Optimistic Offline Lock (Java), 447–48
sales tax service (Java), 505–7
separating finders (Java), 176–79
serializing department hierarchy in XML (Java), 274–77
serializing using XML (Java), 411–13
shared Optimistic Offline Lock (Java), 441–46
shared Pessimistic Offline Lock (Java), 446–47
simple database mapper (Java), 171–76
simple display (Java), 347–49
simple display with servlet controller (Java), 335–37
simple lock manager (Java), 431–37
simple null objects (C#), 498
simple person (Java), 162–64
simple Query Object (Java), 318–21
simple transform (Java), 363–64
simple value objects (Java), 270–71
single table for players (C#), 280–81
single-valued reference (Java), 240–43
singleton registry (Java), 483–84
state model Application Controller (Java), 382–86
swapping Repository strategies (Java), 326–27
thread-safe registry (Java), 484–85
transferring information about albums (Java), 407–11
two-stage XSLT (XSLT), 371–74
Unit of Work with object registration (Java), 190–94
using ADO.NET data sets (C#), 148–51
using compound key (Java), 224–35
using direct SQL (Java), 253–56
using ghosts (C#), 206–14
using Java session been as Remote Facade (Java), 392–95
using JSP as handler (Java), 337–40
using key table (Java), 222–24
using metadata and reflection(Java), 309–15
using single query for multiple employees (Java), 256–61
using value holder (Java), 205–6
virtual proxy (Java), 203–5
Web service (C#), 395–400
Execution contexts, 65–66

F
Facade, Remote, 388–400
examples
using Java session been as Remote Facade (Java), 392–95
Web service (C#), 395–400
how it works, 389–92
when to use it, 392
Fields, Identity, 216–35
examples
integral key (C#), 221–22
using compound key (Java), 224–35
using key table (Java), 222–24
further reading, 221
how it works, 216–20
when to use it, 220–21
Find, multitable, 243–44
Finders, separating, 176–79
Foreign Key Mapping, 236–47
examples
collection of references (C#), 244–47
multitable find (Java}, 243–44
single-valued reference (Java), 240–43
how it works, 236–39
when to use it, 239–40
Front Controller, 344–49
example
simple display (Java), 347–49
further reading, 347
how it works, 344–46
when to use it, 346

G
Gateway, 466–72
example
gateway to proprietary messaging service (Java), 468–72
how it works, 466–67

eaa.book Page 522 Monday, October 7, 2002 8:23 AM

Gateway, *continued*
  Person, 146–48
  when to use it, 467–68
Gateway, Row Data, 152–59
  examples
    data holder for domain object (Java), 158–59
    person record (Java), 155–58
  how it works, 152–53
  when to use it, 153–55
Gateway, Table Data, 144–51
  examples
    Person Gateway (C#), 146–48
    using ADO.NET data sets (C#), 148–51
  further reading, 146
  how it works, 144–45
  when to use it, 145–46
Gateway to proprietary messaging service (Java), 468–72
Generator, Id, 501–3
Ghosts, 202, 206–14
Globally Unique IDentifier (GUID), 219
GUID (Globally Unique IDentifier), 219

H
Handlers
  page, 340–43
  using JSP as, 337–40
Hierarchy, serializing department, 274–77
Holder, using value, 205–6

I
Id Generator (Java), 501–3
Identity Field, 216–35
  examples
    integral key (C#), 221–22
    using compound key (Java), 224–35
    using key table (Java), 222–24
  further reading, 221
  how it works, 216–20
  when to use it, 220–21
Identity Map, 195–99
  example
    methods for Identity Map (Java), 198–99
  how it works, 195–97
  methods for, 198–99
  when to use it, 198
Immutability, isolation and, 66–67
Immutable data, 67
Implicit Lock, 449–53
  example
    implicit Pessimistic Offline Lock (Java), 451–53
  how it works, 450–51
  when to use it, 451
Implicit Pessimistic Offline Lock (Java), 451–53
Inconsistent reads, 64
  preventing, 68–69
Inheritance, 45–47
Inheritance, Class Table, 285–92
  example
    players and their kin (C#), 287–92
  further reading, 287
  how it works, 285–86
  when to use it, 286–87
Inheritance, Concrete Table, 293–301
  example
    concrete players (C#), 296–301
  how it works, 293–95
  when to use it, 295–96
Inheritance Mappers, 302–4
  how it works, 303–4
  when to use it, 304
Inheritance, Single Table
  example
    single table for players (C#), 280–81
  how it works, 278–79
  loading objects from databases, 281–84
  when to use it, 279–80
Initialization, lazy, 203
Input controller patterns, 61
Integral key (C#), 221–22
Interfaces
  for distribution, 92–93
  remote and local, 88–90
  Separated, 476–79
    how it works, 477–78
    when to use it, 478–79
Isolated threads, 66

Isolation
  and immutability, 66–67
  reducing transaction for liveness, 73–74

J
J2EE, Java and, 100–101
J2EE layers, core, 104
Java
  albums and tracks, 264–67
  creating empty objects, 179–81
  data holder for domain object, 158–59
  domain layer with Data Mappers, 421–25
  domain objects, 475
  finding person's dependents, 325–26
  gateway to proprietary messaging service, 468–72
  Id Generator, 501–3
  and J2EE, 100–101
  JSP and custom tags, 374–78
  methods for Identity Map, 198–99
  money class, 491–95
  multitable find, 243–44
  person record, 155–58
  revenue recognition, 113–15, 120–24, 138–41
  root Optimistic Offline Lock, 447–48
  sales tax service, 505–7
  separating finders, 176–79
  serializing department hierarchy in XML, 274–77
  serializing using XML, 411–13
  shared Optimistic Offline Lock, 441–46
  shared Pessimistic Offline Lock, 446–47
  simple database mapper, 171–76
  simple display, 347–49
  simple display with JSP view, 335–37
  simple display with servlet controller, 335–37
  simple person, 162–64
  simple Query Object, 318–21
  simple transform, 363–64
  simple value objects, 270–71
  single-valued reference (Java), 240–43
  singleton registry, 483–84
  state model Application Controller, 382–86
  swapping Repository strategies, 326–27
  thread-safe registry, 484–85
  transferring information about albums, 407–11
  Unit of Work with object registration, 190–94
  using compound key, 224–35
  using direct SQL, 253–56
  using Java session bean as Remote Facade, 392–95
  using JSP as handler, 337–40
  using JSP as view with separate controller, 355–57
  using key table, 222–24
  using metadata and reflection, 309–15
  using single query for multiple employees, 256–61
  using value holder, 205–6
Java session bean, using as Remote Facade (Java), 392–95
JSP
  using as handler, 337–40
  using as view, 355–57
JSP and custom tags (Java), 374–78
JSP view, simple display with, 335–37

K
Key Mapping, Foreign, 236–47
Key table, 222–24
Keys
  compound, 224–35
  integral, 221–22
Kin, players and their, 287–92

L
Late transactions, 72
Layer Supertype, 475
  example
    domain objects (Java), 475
  how it works, 475
  when to use it, 475

eaa.book Page 524 Monday, October 7, 2002 8:23 AM

Layering, 17–24
  choosing to run your layers, 22–24
  evolution of layers in enterprise applications, 18–19
  schemes, 103–6
  three principal layers, 19–22
Layers
  Brown, 104
  choosing to run your, 22–24
  core J2EE, 104
  down to data source, 97–100
    data source for Domain Models, 98–99
    data source for Transaction Scripts, 97–98
    data source Table Modules, 98
    presentation layers, 99–100
  Marinescu, 105
  Microsoft DNA, 105
  Nilsson, 106
  presentation, 99–100
  Service, 30–32
  starting with domain, 96–97
  three principal, 19–22
Layers, evolution of, 18–19
Layers, Service, 133–41
  example
    revenue recognition (Java), 138–41
  further reading, 137
  how it works, 134–37
  when to use it, 137
Lazy initialization (Java), 203
Lazy Load, 200–214
  examples
    lazy initialization (Java), 203
    using ghosts (C#), 206–14
    using value holder (Java), 205–6
    virtual proxy (Java), 203–5
  how it works, 200–203
  when to use it, 203
Liveness, 65
  reducing transactions isolation for, 73–74
Load, Lazy, 200–214
  examples
    lazy initialization (Java), 203
    using ghosts (C#), 206–14
    using value holder (Java), 205–6
    virtual proxy (Java), 203–5
  how it works, 200–203
  when to use it, 203
Loading, ripple, 202
LOBs (large objects), serialized, 272–77
  example
    serializing department hierarchy in XML (Java), 274–77
  how it works, 272–73
  when to use it, 274
Local interfaces, remote and, 88–90
Lock manager, simple, 431–37
Locking
  optimistic, 67
  pessimistic, 67
Locks
  root Optimistic Offline, 447–48
  shared Optimistic Offline, 441–46
  shared Pessimistic Offline, 446–47
Locks, Coarse-Grained, 438–48
  examples
    root Optimistic Offline Lock (Java), 447–48
    shared Optimistic Offline Lock (Java), 441–46
    shared Pessimistic Offline Lock (Java), 446–47
  how it works, 438–41
  when to use it, 441
Locks, Implicit, 449–53
  example
    implicit Pessimistic Offline Lock (Java), 451–53
  how it works, 450–51
  when to use it, 451
Locks, implicit Pessimistic Offline, 451–53
Locks, Optimistic Offline, 416–25
  example
    domain layer with Data Mappers (Java), 421–25
  how it works, 417–20
  when to use it, 420–21
Locks, Pessimistic Offline, 426–37
  example
    simple lock manager (Java), 431–37

how it works, 427–31
when to use it, 431
Logic
business, 20
data source, 20
domain, 20
organizing domain, 25–32
making choices, 29–30
Service Layers, 30–32
presentation, 19–20
Logic patterns, domain, 109–41
Domain Model, 116–24
Service Layer, 133–41
Table Module, 125–32
Transaction Script, 110–15
Long transactions, 72
Lost updates, 64

M
Manager, simple lock, 431–37
Map, Identity, 195–99
example
methods for Identity Map (Java), 198–99
how it works, 195–97
when to use it, 198
Mapper, 473–74
how it works, 473
when to use it, 474
Mapper, Data, 165–81
examples
creating empty objects (Java), 179–81
separating finders (Java), 176–79
simple database mapper (Java), 171–76
how it works, 165–70
when to use it, 170–71
Mapper, simple database, 171–76
Mappers, domain layer with Data, 421–25
Mappers, Inheritance, 302–4
how it works, 303–4
when to use it, 304
Mapping, Association Table, 248–61
examples
employees and skills (C#), 250–53
using direct SQL (Java), 253–56
using single query for multiple employees (Java), 256–61
how it works, 248–49
when to use it, 249
Mapping, building, 47–49
Mapping, Dependent, 262–67
example
albums and tracks (Java), 264–67
how it works, 262–63
when to use it, 263–64
Mapping, Foreign Key, 236–47
examples
collection of references (C#), 244–47
multitable find (Java}, 243–44
single-valued reference (Java), 240–43
how it works, 236–39
when to use it, 239–40
Mapping, Metadata
example
using metadata and reflection(Java), 309–15
how it works, 306–8
when to use it, 308–9
Mapping patterns
object-relational metadata, 305–27
Metadata Mapping, 306–15
Query Object, 316–21
Repository, 322–27
structural, 41–47
inheritance, 45–47
mapping relationships, 41–45
Mapping relationships, 41–45
Mapping to relational databases, 33–53
architectural patterns, 33–38
behavioral problem, 38–39
building mapping, 47–49
database connections, 50–52
reading in data, 40–41
some miscellaneous points, 52–53
structural mapping patterns, 41–47
using metadata, 49–50
Marinescu layers, 105
Messaging service, gateway to, 468–72
Metadata and reflection, using, 309–15
Metadata Mapping, 306–15
example
using metadata and reflection (Java), 309–15

Metadata Mapping, *continued*
how it works, 306–8
when to use it, 308–9
Metadata mapping patterns, object-relational, 305–27
Metadata Mapping, 306–15
Query Object, 316–21
Repository, 322–27
Metadata, using, 49–50
Microsoft DNA layers, 105
Migration, session, 85
Model, Domain, 116–24
example
revenue recognition (Java), 120–24
further reading, 119–20
how it works, 116–19
when to use it, 119
Model View Controller (MVC), 330–32
Models, data source for Domain, 98–99
Modules, data source Table, 98
Modules, Table, 125–32
example
revenue recognition with Table Module (C#), 129–32
how it works, 126–28
when to use it, 128
Money, 488–95
example
money class (Java), 491–95
how it works, 488–90
when to use it, 490–91
Money class (Java), 491–95
Multiple employees, using single query for, 256–61
Multitable find (Java), 243–44
MVC (Model View Controller), 330–32
how it works, 330–32
when to use it, 332

N
.NET, 101–2
Nilsson layers, 106
Null objects, simple, 498

O
Object registration, 186
Object registration, Unit of Work with, 190–94
Object-relational behavioral patterns, 183–214
Identity Map, 195–99
Lazy Load, 200–214
Unit of Work, 184–94
Object-relational metadata mapping patterns, 305–27
Metadata Mapping, 306–15
Query Object, 316–21
Repository, 322–27
Object-relational structural patterns, 215–84
Association Table Mapping, 248–61
Class Table Inheritance, 285–92
Concrete Table Inheritance, 293–301
Dependent Mapping, 262–67
Embedded Value, 268–71
Foreign Key Mapping, 236–47
Identity Field, 216–35
Inheritance Mappers, 302–4
serialized LOBs (large objects), 272–77
Single Table Inheritance, 278–84
Object, simple Query, 318–21
Objects
allure of distributed, 87–88
creating empty, 179–81
domain, 475
loading from databases, 281–84
simple null, 498
simple value, 270–71
Objects, Data Transfer, 401–13
examples
serializing using XML (Java), 411–13
transferring information about albums (Java), 407–11
further reading, 407
how it works, 401–6
when to use it, 406
Objects, Query, 316–21
example
simple Query Object (Java), 318–21
further reading, 318
how it works, 316–17
when to use it, 317–18
Objects, Value, 486–87
how it works, 486–87
when to use it, 487
Offline concurrency, 63, 75

eaa.book Page 527 Monday, October 7, 2002 8:23 AM

Offline concurrency control, patterns for, 76–78
Offline concurrency patterns, 415–53
  Coarse-Grained Lock, 438–48
  Implicit Lock, 449–53
  Optimistic Offline Lock, 416–25
  Pessimistic Offline Lock, 426–37
Offline Lock, implicit Pessimistic, 451–53
Offline Lock, Optimistic, 416–25
  example
    domain layer with Data Mappers (Java), 421–25
  how it works, 417–20
  when to use it, 420–21
Offline Lock, Pessimistic, 426–37
  example
    simple lock manager (Java), 431–37
  how it works, 427–31
  when to use it, 431
Offline Lock, root Optimistic, 447–48
Offline Lock, shared Optimistic, 441–46
Offline Lock, shared Pessimistic, 446–47
Optimistic and pessimistic concurrency controls, 67–71
Optimistic locking, 67
Optimistic Offline Lock, 416–25
  example
    domain layer with Data Mappers (Java), 421–25
  how it works, 417–20
  root, 447–48
  shared, 441–46
  when to use it, 420–21

P
Page Controller, 333–43
  examples
    page handler with code behind (C#), 340–43
    simple display with JSP view (Java), 335–37
    simple display with servlet controller (Java), 335–37
    using JSP as handler (Java), 337–40
  how it works, 333–34
  when to use it, 334–35
Page handler with code behind, 340–43
Patterns
  architectural, 33–38
  base, 465–510
    Gateway, 466–72
    Layer Supertype, 475
    Mapper, 473–74
    Money, 488–95
    Plugin, 499–503
    Record Set, 508–10
    Registry, 480–85
    Separated Interface, 476–79
    Service Stub, 504–7
    Special Case, 496–98
    Value Object, 486–87
  data source architectural, 143–81
    Active Record, 160–64
    Data Mapper, 165–81
    Row Data Gateway, 152–59
    Table Data Gateway, 144–51
  distribution, 387–413
    Data Transfer Objects, 401–13
    Remote Facade, 388–400
  domain logic, 109–41
  input controller, 61
  mapping structural, 41–47
    inheritance, 45–47
    mapping relationships, 41–45
  object-relational behavioral, 183–214
    Identity Map, 195–99
    Lazy Load, 200–214
    Unit of Work, 184–94
  object-relational metadata mapping, 305–27
    Metadata Mapping, 306–15
    Query Object, 316–21
    Repository, 322–27
  object-relational structural, 215–84
    Association Table Mapping, 248–61
    Class Table Inheritance, 285–92
    Concrete Table Inheritance, 293–301
    Dependent Mapping, 262–67
    Embedded Value, 268–71
    Foreign Key Mapping, 236–47
    Identity Field, 216–35
    Inheritance Mappers, 302–4
    Serialized LOBs (large objects), 272–77
    Single Table Inheritance, 278–84

eaa.book Page 528 Monday, October 7, 2002 8:23 AM

Patterns, *continued*
  offline concurrency, 415–53
    Coarse-Grained Lock, 438–48
    Implicit Lock, 449–53
    Optimistic Offline Lock, 416–25
    Pessimistic Offline Lock, 426–37
  session state, 455–64
    Client Session State, 456–57
    Database Session State, 462–64
    Server Session State, 458–61
  view, 58–60
  Web presentation, 329–86
    Application Controller, 379–86
    Front Controller, 344–49
    MVC (Model View Controller), 330–32
    Page Controller, 333–43
    Template View, 350–60
    Transform View, 361–64
    Two Step View, 365–78
Person Gateway (C#), 146–48
Person record (Java), 155–58
Person, simple, 162–64
Person's dependents, finding, 325–26
Pessimistic concurrency controls, optimistic and, 67–71
Pessimistic locking, 67
Pessimistic Offline Lock, 426–37
  example
    simple lock manager (Java), 431–37
  how it works, 427–31
  implicit, 451–53
  shared, 446–47
  when to use it, 431
Phantoms, 73
Plain old Java objects (POJOs), 392
Players
  concrete, 296–301
  single table for, 280–81
Players and their kin (C#), 287–92
Plugin, 499–503
  example
    Id Generator (Java), 501–3
  how it works, 499–500
  when to use it, 500
POJOs (plain old Java objects), 392
Presentation
  layers, 99–100
  logic, 19–20
  Web, 55–61
Presentation patterns, Web, 329–86
  Application Controller, 379–86
  Front Controller, 344–49
  MVC (Model View Controller), 330–32
  Page Controller, 333–43
  Template View, 350–60
  Transform View, 361–64
  Two Step View, 365–78
Problems
  behavioral, 38–39
  concurrency, 64–65
Procedures, stored, 102–3
Process-per-request, 78
Process-per-session, 78
Processes defined, 66
Proxy, virtual, 203–5
Putting it all together, 95–106
  down to data source layers, 97–100
  miscellaneous layering schemes, 103–6
  some technology-specific advice, 100–103
  starting with domain layers, 96–97

Q
Query Object, 316–21
  example
    simple Query Object (Java), 318–21
  further reading, 318
  how it works, 316–17
  when to use it, 317–18
Query Object, simple, 318–21
Query, using single, 256–61

R
Read
  inconsistent, 64
  repeatable, 73
Read committed, 73
Read uncommitted, 74
Reads
  dirty, 74
  preventing inconsistent, 68–69
  Temporal, 69
  unrepeatable, 73

eaa.book Page 529 Monday, October 7, 2002 8:23 AM

Record data, 83
Record Set, 508–10
  how it works, 508–10
  when to use it, 510
Records, Active, 160–64
  example
    simple person (Java), 162–64
  how it works, 160–61
  when to use it, 161–62
References
  collection of, 244–47
  single-valued, 240–43
Reflection, using metadata and, 309–15
Registration
  object, 186
  Unit of Work with object, 190–94
Registry, 480–85
  examples
    singleton registry (Java), 483–84
    thread-safe registry (Java), 484–85
  how it works, 480–82
  singleton, 483–84
  thread-safe, 484–85
  when to use it, 482–83
Relational databases, mapping to, 33–53
  architectural patterns, 33–38
  behavioral problem, 38–39
  building mapping, 47–49
  database connections, 50–52
  reading in data, 40–41
  some miscellaneous points, 52–53
  structural mapping patterns, 41–47
  using metadata, 49–50
Relationships, mapping, 41–45
Remote and local interfaces, 88–90
Remote Facade, 388–400
  examples
    using Java session been as Remote Facade (Java), 392–95
    Web service (C#), 395–400
  how it works, 389–92
  when to use it, 392
Remote Facade, using Java session bean as, 392–95
Repeatable read, 73
Repository, 322–27
  examples
    finding person's dependents (Java), 325–26
    swapping Repository strategies (Java), 326–27
  further reading, 325
  how it works, 323–24
  when to use it, 324–25
Repository strategies, swapping, 326–27
Request transactions, 72
Requests, 65
Resources, transactional, 72–73
Revenue recognition (Java), 113–15, 120–24
Revenue recognition problem, 112–13
Revenue recognition with Table Module (C#), 129–32
Ripple loading, 202
Root Optimistic Offline Lock (Java), 447–48
Row Data Gateway, 152–59
  examples
    data holder for domain object (Java), 158–59
    person record (Java), 155–58
  how it works, 152–53
  when to use it, 153–55

S
Safety, 65
Sales tax service (Java), 505–7
Schemes, miscellaneous layering, 103–6
SCM (source code management), 420
Scripts, data source for Transaction, 97–98
Scripts, Transaction, 110–15
  example
    revenue recognition (Java), 113–15
  how it works, 110–11
  revenue recognition problem, 112–13
  when to use it, 111–12
Separate controller, using JSP as view with, 355–57
Separated Interface, 476–79
  how it works, 477–78
  when to use it, 478–79

eaa.book Page 530 Monday, October 7, 2002 8:23 AM

Serializable, transactions are, 73
Serialized LOBs (large objects), 272–77
  example
    serializing department hierarchy in XML (Java), 274–77
  how it works, 272–73
  when to use it, 274
Serializing using XML (Java), 411–13
Server affinity, 85
Server concurrency, application, 78–80
Server page, ASP.NET, 357–60
Server Session State, 458–61
  how it works, 458–60
  when to use it, 460–61
Servers, stateless, 81
Service Layer, 30–32, 133–41
  example
    revenue recognition (Java), 138–41
  further reading, 137
  how it works, 134–37
  when to use it, 137
Service Stub, 504–7
  example
    sales tax service (Java), 505–7
  how it works, 504–5
  when to use it, 505
Services, gateway to proprietary messaging, 468–72
Services, Web, 103, 395–400
Servlet controller, simple display with, 335–37
Session migration, 85
Session state, 81, 83–86
Session State
  Client, 456–57
    how it works, 456–57
    when to use it, 457
  Database, 462–64
    how it works, 462–63
    when to use it, 464
  Server, 458–61
    how it works, 458–60
    when to use it, 460–61
Session state
  value of statelessness, 81–83
  ways to store, 84–86
  ways to store session state, 84–86
Session state patterns, 455–64
  Client Session State, 456–57
  Database Session State, 462–64
  Server Session State, 458–61
Sessions defined, 66
Shared Optimistic Offline Lock (Java), 441–46
Shared Pessimistic Offline Lock (Java), 446–47
Simple display (Java), 347–49
Simple person (Java), 162–64
Simple transform (Java), 363–64
Single Table Inheritance
  example
    single table for players (C#), 280–81
  how it works, 278–79
  loading objects from databases, 281–84
  when to use it, 279–80
Singleton registry (Java), 483–84
Skills, employees and, 250–53
Source code management (SCM), 420
Source layers, down to data, 97–100
Special Case, 496–98
  example
    simple null objects (C#), 498
  further reading, 497
  how it works, 497
  when to use it, 497
SQL, using direct, 253–56
State
  session, 81, 83–86
  ways to store session, 84–86
State model Application Controller (Java), 382–86
Stateless servers, 81
Stored procedures, 102–3
Strategies
  distribution, 87–93
    allure of distributed objects, 87–88
    interfaces for distribution, 92–93
    remote and local interfaces, 88–90
    where you have to distribute, 90–91
    working with distribution boundaries, 91–92
  swapping Repository, 326–27

eaa.book Page 531 Monday, October 7, 2002 8:23 AM

Structural mapping patterns., 41–47
  inheritance, 45–47
  mapping relationships, 41–45
Structural patterns, object-relational, 215–84
  Association Table Mapping, 248–61
  Class Table Inheritance, 285–92
  Concrete Table Inheritance, 293–301
  Dependent Mapping, 262–67
  Embedded Value, 268–71
  Foreign Key Mapping, 236–47
  Identity Field, 216–35
  Inheritance Mappers, 302–4
  serialized LOBs (large objects), 272–77
  Single Table Inheritance, 278–84
Stub, Service, 504–7
  example
    sales tax service (Java), 505–7
  how it works, 504–5
  when to use it, 505
System transactions, business and, 74–76

T

Table Data Gateway, 144–51
  examples
    Person Gateway (C#), 146–48
    using ADO.NET data sets (C#), 148–51
  further reading, 146
  how it works, 144–45
  when to use it, 145–46
Table Inheritance, Class, 285–92
Table Inheritance, Concrete, 293–301
  example
    concrete players (C#), 296–301
  how it works, 293–95
  when to use it, 295–96
Table Inheritance, Single
  example
    single table for players (C#), 280–81
  how it works, 278–79
  loading objects from databases, 281–84
  when to use it, 279–80
Table Mapping, Association, 248–61
  examples
    employees and skills (C#), 250–53
    using direct SQL (Java), 253–56
    using single query for multiple employees (Java), 256–61
  how it works, 248–49
  when to use it, 249
Table Modules, 125–32
  data source, 98
  example
    revenue recognition with Table Module (C#), 129–32
  how it works, 126–28
  when to use it, 128
Tables, key, 222–24
Tags, JSP and custom, 374–78
Tax service, sales, 505–7
Technology-specific advice, some, 100–103
  Java and J2EE, 100–101
  .NET, 101–2
  stored procedures, 102–3
  Web services, 103
Template View, 350–60
  examples
    ASP.NET server page (C#), 357–60
    using JSP as view with separate controller (Java), 355–57
  how it works, 351–54
  when to use it, 354–55
Temporal Reads, 69
Thread-safe registry (Java), 484–85
Threads
  defined, 66
  isolated, 66
Together, putting it all, 95–106
  down to data source layers, 97–100
  miscellaneous layering schemes, 103–6
  some technology-specific advice, 100–103
  starting with domain layers, 96–97
Tracks, albums and, 264–67
Transaction isolation, reducing for liveness, 73–74

eaa.book Page 532 Monday, October 7, 2002 8:23 AM

Transaction Script, 110–15
  example
    revenue recognition (Java), 113–15
  how it works, 110–11
  revenue recognition problem, 112–13
  when to use it, 111–12
Transaction Scripts, data source for, 97–98
Transactional resources, 72–73
Transactions, 66, 71
  business and system, 74–76
  late, 72
  long, 72
  request, 72
  system, 74–76
Transform, simple, 363–64
Transform View, 361–64
  example
    simple transform (Java), 363–64
  when to use it, 362–63
Two-stage XSLT (XSLT), 371–74
Two Step View, 365–78
  examples
    JSP and custom tags (Java), 374–78
    two-stage XSLT (XSLT), 371–74
  how it works, 365–67
  when to use it, 367–71

U
Uncommitted, read, 74
Unit of Work, 184–94
  example
    Unit of Work with object registration (Java), 190–94
  how it works, 184–89
  when to use it, 189–90
Unit of Work with object registration (Java), 190–94
Unrepeatable reads, 73
Updates, lost, 64

V
Value holder, using, 205–6
Value Object, 486–87
  how it works, 486–87
  when to use it, 487
Value objects, simple, 270–71
Values, Embedded, 268–71
  example
    simple value objects (Java), 270–71
  further reading, 270
  how it works, 268
  when to use it, 268–69
View patterns, 58–60
View, simple display with JSP, 335–37
Views, Template, 350–60
  examples
    ASP.NET server page (C#), 357–60
    using JSP as view with separate controller (Java), 355–57
  how it works, 351–54
  when to use it, 354–55
Views, Transform, 361–64
  example
    simple transform (Java), 363–64
  how it works, 361–62
  when to use it, 362–63
Views, Two Step, 365–78
  examples
    JSP and custom tags (Java), 374–78
    two-stage XSLT (XSLT), 371–74
  how it works, 365–67
  when to use it, 367–71
Virtual proxy (Java), 203–5

W
Web presentation, 55–61
  input controller patterns, 61
  view patterns, 58–60
Web presentation patterns, 329–86
  Application Controller, 379–86
  Front Controller, 344–49
  MVC (Model View Controller), 330–32
  Page Controller, 333–43
  Template View, 350–60
  Transform View, 361–64
  Two Step View, 365–78
Web service (C#), 395–400
Web services, 103

eaa.book Page 533 Monday, October 7, 2002 8:23 AM

Work, Unit of, 184–94
  example
    Unit of Work with object registration (Java), 190–94
  how it works, 184–89
  when to use it, 189–90

X
XML
  serializing department hierarchy in, 274–77
  serializing using, 411–13
XSLT, two stage, 371–74