

Index

NUMERICS

2PC (2-Phase-Commit), transaction management, 233
80/20 rule, Pareto Principle, 11

A

AbstractExcelView interface, 391
abstraction, testing considerations, 440–441
AbstractPdfView interface, 391
AbstractPoolingTargetSource class, 356
AbstractXslView interface, 391
access
 access code, transaction management, 245
 classes, 19
 data access
 clustering challenges, 461–462
 layers, 40–41, 285
 methods, transaction management, 232
 solutions, persistence strategies, 263–264
 transaction management methods, 232
 EJB problems, 103
 file, 175
 JDBC access, via templates, 298–300
 registry access, RMI, 311
 SQL-based, 40
ActionForm class, 504
Active Record, service layer limitations, 35
ad hoc applications
 examples of, 59–61
 overview, 48–50
 testing, 414
addExpectedSetAttribute () method, 427
admin.properties file, 183
after advice, AOP, 192
afterPropertiesSet () method, 163
Agile Manifesto Web site, 25
agile methodologies, 86
aliases, 149
alternate configuration, testing considerations, 442–443
AOP (Aspect Oriented Programming)
 advantages of, 88–89
 after advice, 192
 before advice, 192
 AOP Alliance, 207
 AOSD (Aspect-Oriented Software Development), 228
 application testing, 209
 around advice, 192
 AspectJ implementation, 199–200
 aspects, 191, 196
 AspectWerkz implementation, 201
 Chain of Responsibility pattern, 194–195
 checked exception problem, 227
 code duplication, 188
 crosscutting concerns, 189–191

debugging considerations, 210
 Decorator pattern, 194–195
 design issues, 207–211
 discussed, 2
 field access, 191
 implementation strategies, 197–198
 interception, 88, 193
 JBoss implementation, 201–203
 join points, 191
 method interception, 189
 method invocation, 191
 naming conventions, 225
 Nanning Aspects implementation, 207
 Observer pattern, 194
 performance affects, 210
 pointcuts, 191, 196
 programmatic usage, 219–220
 references, 227–228
 security checks, 188
 Spring Framework and, 145, 203–205
 target objects, 193
 technologies, list of, 187
 testing considerations, 425
 throws, 191
 transaction management, 465–466
 weaving, 192

Apache

Apache Ant, 26
 Apache Avalon Web site, 130
 ApacheBench tool, 480
 JMeter tool, 479
 OJB (ObjectRelationalBridge), 270
 Web site, 26

APIs, third-party, 47

ApplicationContext interface, 177
ApplicationContextAware interface, 179
applicationContext.xml file, 147

applications

ad hoc
 examples of, 59–61
 overview, 48–50
 testing, 414
 application context concept, Spring Framework, 175–177
 application events, support, 175
 application identity, identity management, 281
 application servers
 EJB limitations, 98
 need for, determining, 62–63
 architecture
 selection considerations, 524
 web tier design, 368–372
 central registries within, 177
 component-based, 86

applications (continued)

- development, productivity goals
 - approaches to, 15–16
 - architectural impacts, 21–23
 - business requirements, 28
 - code generation alternatives, 17–20
 - empirical process benefits, 28–29
 - methodologies, 23–25
 - open source solutions, 22
 - problems with, 14–15
 - source control systems, 26
 - template usage, 24–25
 - testing tools, 25–26
- internal distribution, 309
- root web application context, 177
- single database, transaction management, 257–258
- testability considerations, 416–418
- testing, using AOP, 209

architectures

- applications
 - selection considerations, 524
 - web tier design, 368–372
- architectural impacts
 - complexity issues, 66–70
 - productivity goals, 21–23
 - web tier design, 364–365
- classic
 - examples of, 54–57
 - overview, 42–44
- co-located, 68
- data access layer, 40–41
- distributed, 67
- JCA (Java Connection Architecture), 41
- legacy systems, 41
- local
 - examples of, 57–59
 - overview, 44–46
- service layers
 - discussed, 31
 - in lightweight containers, 34–35
 - limitations, 35
 - .NET serviced components, 33–34
 - remote client support, 40
 - SFSBs (stateful session beans), 32
 - SLSBs (stateless session beans), 32
- simplicity considerations, 6–7
- traditional, 38
- validation, 28
- Web interfaces, 38–39

archives

- EAR (Enterprise Application Archive), 325
- RAR (Resource Adapter Archive), 41
- WAR (Web Application Archive), 325, 404

around advice, AOP, 192

articles and online resources, AOP, 227–228

Aspect Oriented Programming. See AOP

AspectJ

- advantages, 199
- AspectJ in Action: Practical Aspect-Oriented Programming* (Ramnivas Laddad), 227
- disadvantages, 200

aspects, AOP, 191, 196

AspectWerkz Web site, 201

ASP.NET

- overview, 409–410
- Tapestry and, 407

- `assertTrue()` method, 431

- Aston, Philip (*J2EE Performance Testing with BEA WebLogic Server*), 485

attributes

- class-level, 222
- Jakarta Commons Attributes, 223
- method-level, 222
- `TransactionAttributeSource` class, 223–224

authentication

- declarative security, 359
- overview, 335

auto-generated testing, 444

autoproxying, 218–219

autowiring

- dependency checks and, 159–160
- overview, 136

B

backward compatibility, 105

- `BaseAction` class, 378

- `BeanFactory` interface, 149–150

- `BeanFactoryAware` interface, 177

- `BeanFactoryPostProcessor` interface, 184

Bean-Managed Persistence (BMP), 264

Bean-Managed Transactions (BMT), 9

before advice, AOP, 192

benchmarking, performance considerations, 464–465, 479–480

binary formats, web tier design, 402

black box testing, 415

blueprints, 71

BMP (Bean-Managed Persistence), 264

BMT (Bean-Managed Transactions), 91

boilerplate code, 19

bold text values, 152

Burke, Bill (JBoss), 111

business interfaces

- RMI, 313–315
- service endpoints, 331–332

business layers, DAO, 285

business objects

- clustering challenges, 458
- DAO (Data Access Object) and, 285–287
- interface-implementation separation, 126–127
- lightweight containers and, 50
- managing, 96–97
- `MyBusinessObject` class, 129
- POJOs, 48
- web tier design, 369

business requirements, portability, 28

business service layers. See service layers

byte code, 266

C

caching

- EHCACHE product, 267
- OSCache product, 267
- performance considerations, 472–473
- persistence strategies, 266–267
- pluggable cache strategies, 282
- query results, 275
- second-level, 278
- SwarmCache product, 267
- Tangosol Coherence technology, 93

Cactus testing framework, 101

callbacks, 162–163, 177–178

- career-driven development, complexity issues, 74**
- case studies, testing, 437–439**
- catch block, 56, 59
- central registries, 177**
- central transaction infrastructure, transaction management, 241**
- CGLIB (Code Generation Library), 198**
- Chain of Responsibility pattern, AOP, 194–195**
- checked exception problem, AOP, 227**
- class loaders**
 - custom, 198
 - server problems, 100
- classes. See also interfaces**
 - AbstractPoolingTargetSource, 356
 - access, 19
 - ActionForm, 504
 - attributes, 222
 - BaseAction, 378
 - CommonsPoolInvokerInterceptor, 356
 - ContextUtil, 222
 - DaoManager, 492
 - DataAccessException, 295, 300
 - DataIntegrityViolationException, 295
 - DataSourceTransactionManager, 253–255
 - DefaultAdvisorAutoProxyCreator, 218–219
 - DefaultInventoryManager, 158, 166
 - DriverManagerDataSource, 169–170
 - EJBException, 91
 - HibernateTemplate, 147, 304–305
 - HibernateTransactionManager, 256–257
 - interfaces versus, 27
 - IntroductionAdvisors, 203
 - InventoryManager, 156
 - InventoryManagerFactory, 158
 - JdbcTemplate, 298–300
 - JdoTemplate, 303
 - jdoTransactionManager, 255–256
 - JndiObjectFactoryBean, 254
 - JtaTransactionManager, 253
 - LocalSessionFactoryBean, 173–175
 - MappingSqlQuery, 300
 - MethodInterceptor, 205
 - MockInventoryManager, 158
 - MockProductManager, 158
 - MyBusinessObject, 129
 - MyDataAccessObject, 252
 - myInventoryManager, 159
 - myProductManager, 159
 - OpenSessionInViewFilter, 305
 - OpenSessionInViewInterceptor, 305
 - OptimisticLockingFailureException, 295
 - OrderManager, 286
 - PersistenceManager, 264, 279–280
 - PersistenceManagerFactory, 273
 - PetStoreLogic, 60–61
 - PlatformTransactionManager, 218, 243
 - Product, 155–156
 - ProductManager, 155
 - ProductManagerFactory, 158
 - ProxyFactoryBean, 213–217, 248
 - RdbmsOperation, 300–301
 - RegexpMethodPointcut, 216
 - RegexpMethodPointcutAdvisor, 216
 - RemoteException, 45, 98, 310, 314
 - RemoteOrderService, 512
 - ResultSet, 59
 - ServerEndpointSupport, 511
 - service endpoints, 333
 - ServicedComponent, 33
 - ServletContextListener, 273
 - ServletFilter, 273
 - SessionFactory, 168, 174
 - SomeObject, 216
 - SQLException, 274
 - SqlUpdate, 300
 - StoredProcedure, 300
 - Swing, event-driven frameworks, 404
 - testing, 415
 - TextStyle, 151
 - ThreadLocalTargetSource, 357
 - TransactionAttributeSource, 223–224
 - TransactionFactoryBean, 217–218
 - TransactionInterceptor, 209
 - TransactionProxyFactoryBean, 217–218
 - TransactionTemplate, 240, 243–244
 - Transport, 420
 - ValueHolder, 284
 - ViewItemAction, 505
 - ViewItemController, 505
 - WebApplicationContextUtils, 148, 334, 387
 - WebLogSystem, 386
 - XmlBeanFactory, 154–155
- ClassFilter interface, 206**
- classic architectures**
 - examples of, 54–57
 - overview, 42–44
- client-side proxy, RMI, 312**
- close() method, 170, 254
- Clover tool**
 - discussed, 26
 - testing options, 444–445
- clustering services**
 - deployment, 455
 - lightweight containers, 123
 - overview, 92–94
 - performance considerations, 456–460
- CMT (Container Managed Transactions)**
 - declarative transaction management services, 90–92
 - disadvantages of, 234–235
 - discussed, 3
- CocoBase O/R mapping tool, 239, 269**
- code**
 - coding practices, testing considerations, 431–433
 - coding styles, IoC, 138–139
 - complex, solutions to, 20–21
 - duplication, 15, 188
 - generation alternatives, 17–20
 - hand-authored, 20
 - management, lightweight containers, 123
 - optimization, performance considerations, 471
 - reusability, lightweight container benefits, 125
- Code Generation Library (CGLIB), 198**
- collections support, Spring Framework representation, 136**
- co-located architectures, 68**
- command-driven controllers, 394–397**
- commercial containers, 92**
- Commons Attributes, Jakarta, 223**
- CommonsPoolInvokerInterceptor class, 356**
- communication, complexity issues, 75**
- compile-time byte code modification, persistence strategies, 266**

complexity

- architectural causes, 66–70
 - career-driven development, 74
 - communication issues, 75
 - complex code, solutions to, 20–21
 - cultural causes of, 71–75
 - deployment issues, 67
 - distribution and scalability, 68–69
 - EJB programming model, 102
 - error handling issues, 67
 - industry expectations, 77
 - object distribution, 66–70
 - organizational requirements, 73
 - overcoming, 77–78
 - patterns, 71
 - performance issues, 67
 - platform capabilities, 70
 - political changes, 74–75
 - semantic issues, 70
 - tools and, 72–73
 - unnecessary code, 65
- component interface, SLSBs, 321**
- component-based applications, 86**
- component-level testing, 417**
- compression, 335**
- concatenation, strings, 474**
- conceptual versus implementation-level metadata, 221**
- concerns, AOP, 191**
- concurrency libraries, thread management, 352–353**
- concurrency support, thread management, 349–353**
- Concurrent Programming Java: Design Principles and Patterns (Doug Lea), 351**
- configuration, lightweight containers, 122**
- connection factories, persistence strategies, 272–273**
- connections**
- database connection pools, 95
 - `getConnection()` method, 169
- consistency, lightweight containers, 125**
- Constructor Injection**
- IoC, 132–134
 - Spring Framework and, 160–162
- Container Managed Transactions. See CMT**
- containers**
- commercial, 92
 - dedicated, EJB, 324
 - HiveMind, 111
 - in-container testing, 417
 - IoC (Inversion of Control), 135–138
 - lightweight
 - accessing, 51
 - advantages of, 52–53, 125–126
 - business objects and, 50
 - clustering services, 123
 - code management, 123
 - configuration services, 122
 - consistency issues, 125
 - customization and extensibility services, 123
 - dependency resolution services, 122
 - disadvantages of, 53
 - EJB versus, 125–126
 - enterprise services, 123
 - implementation issues, 53–54
 - lifecycle management service, 122
 - lookup services, 122
 - object pooling services, 123
 - pluggability issues, 124
 - remoting services, 123
 - sample application, 61–62
 - service layers, 34–35
 - thread management services, 123
 - uses for, 122
 - value-added services, 123
 - PicoContainer, 137
- ContextUtil class, 222**
- control logic, web tier design, 366**
- controllers**
- command-driven, 394–397
 - form-driven, 397–399, 504–505
 - web tier design, 374
- CORBA language, 83**
- coverage analysis, testing considerations, 438, 444–447**
- create() method, 319**
- Criteria Queries, Hibernate, 283**
- crosscutting concerns, AOP, 189–191**
- cultural causes of complexity, 71–75**
- currentTransactionStatus() method, 441**
- custom class loaders, 198**
- customization, lightweight containers, 123**

D

DAO (Data Access Object)

- business layers, 285
- business objects and, 285–287
- data access layers, 285
- delete operations, 290
- design issues, 289–292
- discussed, 41
- exception handling, 292
- iBatis, 292
- infrastructure issues, 292–293
- JDBC-based, 288
- load operations, 290
- method types, 289–290
- `OrderDao` class, 296
- portable interfaces, 288
- read-only, 289
- store operations, 290
- types of, 288–289
- vendor extensions, leveraging, 288
- web layers, 285

DaoManager class, 492

data access

- clustering challenges, 461–462
- layers, 40–41, 285
- methods, transaction management, 232
- solutions, persistence strategies, 263–264
- transaction management methods, 232

Data Access Object. See DAO

data binding, XML, 462

data holders, 89

data mappers

- defined, 262
- service layer limitations, 35

Data Transfer Objects (DTOs), 27

DataAccessException class, 295, 300

databases

- configuration, performance considerations, 477–478
- connection pools, 95
- RDBMS
 - SQL-based access to, 40
 - tables, 27
- updates, clustering challenges, 461

- DataIntegrityViolationException **class**, 295
 - DataSourceTransactionManager **class**, 253–255
 - datastore identity, identity management**, 281
 - debugging considerations, AOP**, 210
 - declarative security, thread management**, 359–360
 - declarative transactions**
 - demarcation, 234
 - management services, 90–92
 - declarative validation**, 377
 - Decorator pattern, AOP**, 194–195
 - decoupling**
 - loose coupling, 27
 - overview, 292
 - testing techniques, 415
 - dedicated containers, EJB**, 324
 - default isolation code**, 240
 - DefaultAdvisorAutoProxyCreator **class**, 218–219
 - DefaultInventoryManager **class**, 158, 166
 - delete operations, DAO**, 290
 - dependency checking**
 - autowiring and, 159–160
 - Spring Framework representation, 136
 - Dependency Injection, IoC**, 130–131, 135
 - Dependency Lookup, IoC**, 128–130
 - dependency resolution**
 - lightweight containers, 122
 - Spring Framework, 159–160
 - deployment**
 - complexity issues, 67
 - deployment clustering, 455
 - EJB problems, 100
 - tuning options, performance considerations, 476–478
 - design issues**
 - AOP (Aspect Oriented Programming), 207–211
 - DAO (Data Access Object), 289–292
 - markup generation, 408
 - OOD (object-oriented design), 348
 - Struts framework, 375–379
 - web tier design
 - application architecture, 368–372
 - architectural issues, 364–365
 - binary formats, 402
 - business objects, 369
 - clustering challenges, 457–458
 - command-driven controllers, 394–397
 - control logic, 366
 - controllers, 374
 - discussed, 363
 - event-driven frameworks, 404–408
 - form-driven controllers, 397–399
 - interceptors, 374
 - IoC-based middle tier, 371–373
 - MVC approach, 366
 - naive approaches, 365
 - open source frameworks, 368
 - portals, 403–404
 - portlets, 403–404
 - request-driven frameworks, 368, 374–377
 - servlets, 366–368
 - singletons, 369
 - SLSBs (stateless session beans), 369
 - static templates, 406
 - validation, 374
 - view technologies, 401–402
 - WebLogSystem class, 386
 - wizard-style workflows, 399
 - destroy() **method**, 163
 - diagnostics, performance options**, 484–485
 - DII (Dynamic Invocation Interface)**, 328
 - disassociation, persistence strategies and**, 280
 - disconnected objects**, 35
 - DispatcherServlet **interface**, 373
 - DisposableBean **interface**, 177
 - distribution**
 - architectures, 67
 - scalability and, complexity issues, 68–69
 - transaction requirements, transaction management, 258
 - doEdit() **method**, 403
 - doHelp() **method**, 403
 - domain arguments, finder methods with**, 288
 - domain object behaviors, persistence strategies**, 268
 - doSecurityCheck() **method**, 189–190
 - doUpdate() **method**, 351
 - doView() **method**, 403
 - DriverManagerDataSource **class**, 169–170
 - DTOs (Data Transfer Objects)**, 27
 - dynamic byte code generation**, 198
 - Dynamic Invocation Interface (DII)**, 328
 - dynamic mock objects, unit testing**, 428–429
 - dynamic proxies**
 - AOP implementation strategy, 197–198
 - service endpoints, 328
 - DynaMock Web site**, 429
- ## E
- EAI (Enterprise Application Integration)**, 326
 - EAR (Enterprise Application Archive)**, 325
 - EasyMock Web site**, 428–429
 - Eclipse plug-ins**, 26, 483
 - efficiency considerations, EJB**, 97
 - EHCACHE product, Hibernate**, 267
 - 80/20 rule, Pareto Principle**, 11
 - EIS (Enterprise Information System)**, 40
 - EJB BMT (EJB Bean Managed Transactions)**, 236
 - EJB (Enterprise JavaBeans)**
 - application server limitations, 98
 - benefits of, 126
 - business object management, 96–97
 - complex programming models, 102
 - container services, 3–4
 - dedicated containers, 324
 - deployment problems, 100
 - discussed, 97
 - efficiency considerations, 97
 - EJB 3.0 version, 105, 225
 - encapsulation, 87
 - enterprise concept problems, 103
 - instance pooling, 94–95
 - interception model, 195–196
 - lightweight containers versus, 125–126
 - myths and fallacies, 105–107
 - overuse, 102
 - portability problems, 104
 - productivity loss, 104
 - proliferation of classes, 98
 - remote, 319–323
 - resource pooling, 95
 - restricted access problems, 103
 - security management, 95–96
 - singleton implementation problems, 103
 - specifications, 1–2, 81
 - strengths/weaknesses, 1–3

EJB (Enterprise JavaBeans) (continued)

- testing problems, 100–102
- thread management, 94
- tool support, 82, 104–105
- when to use, 107–109

EJBContext interface, 84

ejbCreate() method, 129

EJBException class, 91

EJBLocalObject interface, 98

EJBObject interface, 98

EJBUtil interface, 58

EL (Expression Language), 377, 384

EMCA Web site, 112

empirical process benefits, productivity goals, 28–29

encapsulation, 87

encryption, 335

endpoints

- business interfaces, 331–332
- classes, 333
- discussed, 328
- exposure, 330–331
- servlets, 332–334

Enterprise Application Archive (EAR), 325

Enterprise Application Integration (EAI), 326

enterprise concept problems, EJB, 103

Enterprise Information System (EIS), 40

Enterprise JavaBeans. See EJB

Enterprise Objects Framework (EOF), 269

enterprise services, lightweight containers, 123

entity beans

- clustering services, 92–94
- object orientation and, 27
- persistence strategies, 269–271

environment variables, 47

EOF (Enterprise Objects Framework), 269

equivalence partitioning, testing considerations, 430–431

ERROR constant, 382

error handling, complexity issues, 67

event-driven frameworks, web tier design, 404–408

evidence-based approaches, performance considerations, 478–482

exception handling, DAO infrastructure issues, 292

execute() method, 305

executeUpdate() method, 277

exposing resources, 168

Expression Language (EL), 377, 384

extensibility, lightweight containers, 123

Extreme Programming (XP), 86

F

factory beans

- overview, 171
- ProxyFactoryBean class, 213–217
- TransactionFactoryBean class, 217–218

fail() method, 432

fake objects, 27

field access, AOP, 191

files

- access to, 175
- admin.properties, 183
- applicationContext.xml, 147
- getFile() method, 180
- finally blocks, 59, 244
- find() method, 305
- finder methods, with domain arguments, 288
- findForward() method, 375

FireStorm Web site, 16

Fleury, Marc (JBoss), 111

formBackingObject() method, 399

form-driven controllers, 397–399, 504–505

Fowler, Martin (Patterns of Enterprise Application Architecture), 35, 67, 231, 261, 455

frameworks

- Cactus testing, 101
- EOF (Enterprise Objects Framework), 269
- event-driven, web tier design, 404–408
- open source, web tier design, 404–408
- request-driven, web tier design, 368, 374–377
- Spring Framework
 - advantages, 39
 - AOP support, 145, 203–205
 - application context concept, 175–177
 - callbacks, 162–163, 177–178
 - complex property value support, 164–165
 - Constructor Injection, 160–162
 - dependency resolution, 159–160
 - factory bean support, 171–172
 - file resources, 180–182
 - IoC (Inversion of Control), 135–137
 - JDBC abstraction, 145
 - JNDI resource availability, 172–173
 - local connection factory, 173–174
 - motivations for, 144
 - overview, 146–147
 - resource setup, 165–169
 - sub-frameworks, 145–146
 - template resources, 24
 - transaction management, 145, 239–243
 - Web applications, 147–149
 - Web site, 10–11
 - Struts, 375–379

FreeMarker Web site, 401

G

GC (garbage collection), 347–348

generational garbage collection, 347

GET request, 398

getAliases() method, 149

getBalance() method, 429

getBank() method, 431

getBean() method, 149, 157, 214

getCallerPrincipal() method, 359

getCatalogHome() method, 58

getConnection() method, 169, 254

getFile() method, 180

getHeadline() method, 473

getInputStream() method, 180

getInstance() method, 420

getJdbcTemplate() method, 300

getJdoTemplate() method, 303

getMessage() method, 178–179

getObject() method, 171

getObjectType() method, 171

getPersistenceManager() method, 303

getPort() method, 328

getServletContext() method, 178

getSession() method, 282

getSqlMapTemplate() method, 302, 501

getStatus() method, 55

getStockServicePort() method, 328

getter() method, 283

getWebApplicationContext() method, 334

glass box testing, 415

global objects, 406

Grinder performance tool, 480

GUI tools, Java Server Faces, 407

GZIP compression, 335

H

hand-authored code, 20

handlers, requests, 388

hash tables, 343

Hibernate

Criteria Queries, 283

EHCache product, 267

HibernateTemplate class, 147, 304–305

HibernateTransactionManager class, 256–257

lazy loading feature, 284, 292

Open Session in View pattern, 292

O/R mapping solutions, 111

pluggable cache strategies, 282

snapshot comparisons, 284

Web site, 264

high-level transaction management, 231–232

HiveMind container, 111

horizontal scalability, 94, 453

HTML interfaces, web tier design, 364–365

HttpServletRequest interface, 382

HttpServletResponse interface, 382

Hunter, Jason (Java Enterprise Best Practices), 345

Husted, Ted (JUnit in Action), 412, 449

I

iBatis

discussed, 112

SQL Maps, 264, 275–278

Web site, 263, 278

IDE tools, proliferation of classes, 98

identity management, persistence strategies, 281

identity maps, persistence strategies, 263

impedance mismatch

defined, 23

persistence strategies and, 262

implementation

AOP (Aspect Oriented Programming), 197–198

implementation-level versus conceptual metadata, 221

interface-implementation separation, business objects, 126–127

IoC strategies, 128–132

lightweight containers, 53–54

local architectures, 47

reference implementations, 112

simplicity considerations, 6–7

singleton implementation problems, 103

indexes, database configuration, 478

industry expectations, complexity issues, 77

infrastructure issues, DAO, 292–293

inheritance, performance, 475

InitializingBean interface, 177

INPUT constant, 382

instance creation, thread management, 344

instance fields, JDO, 279

instance pooling

overview, 94–95

performance considerations, 468

thread management, 347–349

integration testing, 414

interception

AOP, 88, 193

interception model, EJB, 195–196

transaction management, 246–249

web tier design, 374

interface-implementation separation, business objects, 126–127

interfaces. See also classes

AbstractExcelView, 391

AbstractPdfView, 391

AbstractXslView, 391

ApplicationContext, 177

ApplicationContextAware, 179

BeanFactory, 149–150

BeanFactoryAware, 177

BeanFactoryPostProcessor, 184

business interfaces

RMI, 313–315

service endpoints, 331–332

classes versus, 27

ClassFilter, 206

DII (Dynamic Invocation Interface), 328

DispatcherServlet, 373

DisposableBean, 177

EJBContext, 84

EJBLocalObject, 98

EJBObject, 98

EJBUtil, 58

HttpServletRequest, 382

HttpServletResponse, 382

InitializingBean, 177

ListableBeanFactory, 150

MethodInvocation, 215

MethodMatcher, 206

MyCollaborator, 129

OrderDao, 296

OrderServer, 513

Pointcut, 206

portable, 288

proxyInterfaces, 214

ResourceLoader, 180–181

ResourceLoaderAware, 180

TargetSource, 355

ThreadLocalInvokerStatus, 358

TransactionAttribute, 240

TransactionTemplate, 240

UserTransaction, 236–237

View, 391

internal distribution, applications, 309

IntroductionAdvisors class, 203

InventoryManager class, 156

InventoryManagerFactory class, 158

invoke() method, 215

IoC (Inversion of Control)

coding styles, 138–139

Constructor Injection, 132–134

Dependency Injection, 130–131, 135

Dependency Lookup, 128–130

discussed, 121–122, 127

implementation strategies, 128–132

language-based, 130

PicoContainer, 137

portability issues, 137–138

push dependencies, 442

Setter Injection, 131–133

IoC (Inversion of Control) (continued)

- Spring Framework container, 135–137
- testability, 139
- testing considerations, 425
- web tier design, 371–373
- `isDebugEnabled()` **method**, 474
- `isDirty()` **method**, 211
- `isInfoEnabled()` **method**, 481
- isolation level, transaction management**, 240
- `isRejected()` **method**, 55
- `isRunTime()` **method**, 206
- `isSingleton()` **method**, 149
- italic text values**, 152
- iterative development, testing**, 433

J

- Jakarta, Commons Attributes**, 223
- Java Connection Architecture (JCA)**, 41, 233
- Java Data Objects**. *See* **JDO**
- Java Enterprise Best Practices (Jason Hunter)**, 345
- Java language improvements**, 83
- Java Message Service (JMS)**, 236, 360
- Java Performance Tuning (Jack Shirazi)**, 485
- Java Server Faces GUI tool**, 407
- Java Transaction API (JTA)**, 232
- JavaBeans**
 - non-XML bean definition formats, 154–155
 - populating, via XML, 151–153
- JAX-RPC specification**, 326
- JBoss**
 - advantages, 201–202
 - Bill Burke, 111
 - disadvantages, 201–202
 - Marc Fleury, 111
- JCA (Java Connection Architecture)**, 41, 233
- JDBC**
 - abstraction, 145
 - access, via templates, 298–300
 - JDBC-based DAOs, 288
 - `JdbcTemplate` class, 298–300
 - operation objects, 298
 - persistence strategies, 273–274
- JDO (Java Data Objects)**
 - discussed, 270
 - instance fields, 279
 - `JdoTemplate` class, 303
 - named queries, 279
 - persistence strategies, 278–280
- `JdoTemplate` **class**, 303
- `JdoTransactionManager` **class**, 255–256
- Jester testing tool**, 448
- JMeter tool, Apache**, 479
- JMS (Java Message Service)**, 236, 360
- JNDI resource availability**, 172–173
- `JndiObjectFactoryBean` **class**, 254
- join points**, 191
- JOTM Web site**, 63
- JSTL (JSP Standard Tag Library)**, 377
- JTA (Java Transaction API)**, 232
- `JtaTransactionManager` **class**, 253
- Jtest tool**, 444
- J2EE Performance Testing with BEA Web Logic Server (Peter Zadrozny, Philip Aston and Ted Osborne)**, 485
- JUnit in Action (Ted Husted and Vincent Massol)**, 412, 449
- JUnit testing tool**, 26, 436

L

- Laddad, Ramnivas (AspectJ in Action: Practical Aspect-Oriented Programming)**, 227
- language improvements, Java**, 83
- language-based IoC**, 130
- language-level support, thread management**, 344
- latency**, 453
- Law of Demeter, testing considerations**, 423–424
- layers**
 - data access layers
 - DAO (Data Access Object), 285
 - overview, 40–41
 - service layers
 - discussed, 31
 - in lightweight containers, 34–35
 - limitations, 35
 - .NET serviced components, 33–34
 - remote client support, 40
 - SFSBs (stateful session beans), 32
 - SLSBs (stateless session beans), 32
- lazy loading feature**, 284, 292
- Lea, Doug (Concurrent Programming Java: Design Principles and Patterns)**, 351
- learning curves, complexity issues**, 73
- legacy systems**, 41
- library challenges, testing considerations**, 420–421
- licensing costs, complexity issues**, 73
- lifecycle management, lightweight containers**, 122
- lightweight containers**
 - accessing, 51
 - advantages of, 52–53, 125–126
 - business objects and, 50
 - clustering services, 123
 - code management, 123
 - configuration services, 122
 - consistency issues, 125
 - customization and extensibility services, 123
 - dependency resolution services, 122
 - disadvantages of, 53
 - EJB versus, 125–126
 - enterprise services, 123
 - implementation issues, 53–54
 - lifecycle management service, 122
 - lookup services, 122
 - object pooling services, 123
 - pluggability issues, 124
 - reasons for, 124–125
 - remoting services, 123
 - sample application, 61–62
 - service layers, 34–35
 - thread management services, 123
 - uses for, 122
 - value-added services, 123
- lightweight transaction infrastructure**, 238–239
- linear returns, clustering challenges**, 457
- `ListableBeanFactory` **interface**, 150
- listeners and startup servlets**, 516–517
- load operations, DAO**, 290
- load testing**, 479
- `loadLineItems()` **method**, 289
- `loadOrder()` **method**, 289
- `loadOrderAsDisassociated()` **method**, 291
- `loadOrderAsTransactional()` **method**, 291
- `loadProducts()` **method**, 276

local architectures

- examples of, 57–59
- overview, 44–46

local SLSBs, 90

LocalSessionFactoryBean **class**, 173–175

locking

- lock splitting, 351
- thread management and, 345–346

logging levels

- benchmarking, 479
- performance considerations, 477

logical separation, Web interfaces, 38–39**LOGIN constant, 382****lookup functionality**

- lightweight containers, 122
- testing considerations, 418

M**maintainability, performance considerations, 9****major collection, garbage collection, 348**

makeTransient() **method**, 281

mandatory propagation code, 240**mapping requests, 389**

MappingSqlQuery **class**, 300

mapRow() **method**, 301

markitecture, 71–72**markup generation, web tier design, 408****Massol, Vincent (*JUnit in Action*), 412, 449****Mastering Enterprise JavaBeans (Ed Roman), 107–108**

matches() **method**, 206

Maven Web site, 26**MDBs (Message-Driven Beans), 360****memory management, thread management, 347–348****messages**

- message sources, 178–179
- support, 175

metadata

- conceptual versus implementation-level, 221
- source-level, 84–85, 220–221

MethodInterceptor **class**, 205

MethodInvocation **interface**, 215

MethodMatcher **interface**, 206

methodologies, application development productivity goals, 23–25**methods**

- addExpectedSetAttribute(), 427
- afterPropertiesSet(), 163
- assertTrue(), 431
- attributes, 222
- close(), 170, 254
- create(), 319
- currentTransactionStatus(), 441
- destroy(), 163
- doEdit(), 403
- doHelp(), 403
- doSecurityCheck(), 189–190
- doUpdate(), 351
- doView(), 403
- ejbCreate(), 129
- execute(), 305
- executeUpdate(), 277
- fail(), 432
- find(), 305
- findForward(), 375
- formBackingObject(), 399
- getAliases(), 149

- getBalance(), 429
- getBank(), 431
- getBean(), 149, 157, 214
- getCallerPrincipal(), 359
- getCatalogHome(), 58
- getConnection(), 169, 254
- getFile(), 180
- getHeadline(), 473
- getInputStream(), 180
- getInstance(), 420
- getJdbcTemplate(), 300
- getJdoTemplate(), 303
- getMessage(), 178–179
- getObject(), 171
- getObjectType(), 171
- getPersistenceManager(), 303
- getPort(), 328
- getServletContext(), 178
- getSession(), 282
- getSqlMapTemplate(), 302, 501
- getStatus(), 55
- getStockServicePort(), 328
- getter(), 283
- getWebApplicationContext(), 334
- interception, AOP, 189
- invocation, AOP, 191
- invoke(), 215
- isDebugEnabled(), 474
- isDirty(), 211
- isInfoEnabled(), 481
- isRejected(), 55
- isRunTime(), 206
- isSingleton(), 149
- loadLineItems(), 289
- loadOrder(), 289
- loadOrderAsDisassociated(), 291
- loadOrderAsTransactional(), 291
- loadProducts(), 276
- makeTransient(), 281
- mapRow(), 301
- matches(), 206
- new(), 102, 353
- onBindandValidate(), 399
- perform(), 378
- placeOrder(), 466
- private(), 475
- proceed(), 205, 215
- processAction(), 403
- query(), 298
- redirect(), 427
- replay(), 429
- requiresNoSecurityCheck(), 193
- saveorUpdateCopy(), 285
- setApplicationContext(), 178
- setBeanFactory(), 178
- setDataSource(), 169
- setInventoryManager(), 156
- setOrderDao(), 296
- setReadOnly(), 241
- setResourceLoader(), 178
- setReturnValue(), 429
- setRollbackOnly(), 91, 209, 226, 248, 441
- setter(), 283
- setTransactionManager(), 244
- setUp(), 417, 438
- setupAddParameter(), 427

methods (continued)

- `specialMethod()`, 216
- static, testing considerations, 420
- `storeLineTimes()`, 289
- `storeOrder()`, 286, 289
- `tearDown()`, 433, 438
- testing, 431
- `toString()`, 474, 481
- `UnicastRemoteObject()`, 311
- `UnsupportedOperationException()`, 439
- `update()`, 300
- `updateInventory()`, 286
- `updateProduct()`, 276
- `verify()`, 427–428

Microsoft Transaction Server (MTS), 82

minor collection, garbage collection, 347–348

mock objects, unit testing, 426–430

MockEJB Web site, 417

MockInventoryManager class, 158

MockProductManager class, 158

MTS (Microsoft Transaction Server), 82

multi-threaded service objects, 94

mutation testing tools, 447–448

MVC approach, web tier design, 366

MyBusinessObject class, 129

MyCollaborator interface, 129

MyDataAccessObject class, 252

myInventoryManager class, 159

myProductManager class, 159

N

named queries, JDO, 279

naming conventions, 225

NamingException object, 130

Nanning Aspects

AOP and, 207

Jon Tirsén, 111

Web site, 207

NanoContainer Web site, 137

negative testing, 430

.NET

challenges, 83–85

serviced components, 33–34

never propagation code, 240

new() method, 102, 353

NONE constant, 382

not supported propagation code, 240

O

Oberg, Richard (J2EE development), 111

Object Data Management Group (ODMG), 269

object orientation. See OO

object pooling, lightweight containers, 123

object-oriented design (OOD), 348

ObjectRelationalBridge (OBJ), 270

objects

business objects

clustering challenges, 458

DAO (Data Access Object) and, 285–287

interface-implementation separation, 126–127

lightweight containers and, 50

managing, 96–97

`MyBusinessObject` class, 129

POJOs, 48

web tier design, 369

creation, thread management and, 353

DAO (Data Access Object)

business layers, 285

business objects and, 285–287

data access layers, 285

delete operations, 290

design issues, 289–292

discussed, 41

exception handling, 292

iBatis, 292

infrastructure issues, 292–293

JDBC-based, 288

load operations, 290

methods types, 289–290

`OrderDao` class, 296

portable interfaces, 288

read-only, 289

store operations, 290

types of, 288–289

vendor extensions, leveraging, 288

web layers, 285

disconnected, 35

distribution

complexity issues, 66–70

performance considerations, 455–456

DTOs (Data Transfer Objects), 27

`EJBLocalObject` interface, 98

`EJBObject` interface, 98

EOF (Enterprise Objects Framework), 269

fake, 27

`formBackingObject()` method, 399

`getObject()` method, 171

`getObjectType()` method, 171

global, 406

JDO (Java Data Objects)

discussed, 270

instance fields, 279

`JdoTemplate` class, 303

named queries, 279

persistence strategies, 278–280

mock objects, unit testing, 426–430

`MyDataAccessObject` class, 252

proliferation of, EJB problems, 98

query objects, persistence strategies, 263

SOAP (Simple Object Access Protocol), 325–327

target objects, AOP, 193

`UnicastRemoteObject()` method, 311

visit, 406

WebObjects Web site, 404

Observer pattern, AOP, 194

ODMG (Object Data Management Group), 269

OJB (ObjectRelationalBridge), 270

onBindandValidate() method, 399

online resources. See resources

OO (object orientation)

advantages of, 26–27

examples of, 7–8

lightweight container benefits, 126

OOD (object-oriented design), 348

Open Session in View pattern, Hibernate, 292

open source frameworks, web tier design, 368

OpenSessionInViewFilter class, 305

OpenSessionInViewInterceptor class, 305

OpenSymphony, OSCache product, 267

operation objects, JDBC, 298

`OptimisticLockingFailureException` class, 295

O/R mapping

- clustering challenges, 459
- CocoBase mapping tool, 239, 269
- mapping solutions, 111
- ODMG (Object Data Management Group), 269
- productivity goals and, 23
- TopLink mapping tool, 239, 269
- when to use, 265–266

Oracle Technology Network (OTN), 55

`OrderDao` interface, 296

`OrderManager` class, 286

`OrderServer` interface, 513

organizational requirements, complexity issues, 73

orthogonal join point model, 208

Osborne, Ted (*J2EE Performance Testing with BEA WebLogic Server*), 485

OSCache product (OpenSymphony), 267

OTN (Oracle Technology Network), 55

overusing EJB, 102

P

parameter maps, SQL Maps, 275

Pareto Principle, 80/20 rule, 11

password properties, security, 337

patterns, complexity issues, 71

Patterns of Enterprise Application Architecture (Martin Fowler), 35, 67, 231, 261, 455

PDF files, 402

`perform()` method, 378

performance. *See also* testing

AOP and, 210

Apache JMeter tool, 479

ApacheBench tool, 480

architectural choices, 454

benchmarking, 464–465, 479–480

caching, 472–473

clustering challenges, 456–460

code optimization, 471

complexity issues, 67

data access, 461–462

database configuration, 477–478

declarative and programmatic middleware models, 465–467

deployment clustering, 455

deployment-tuning options, 476–478

diagnostics, 484–485

discussed, 451

evidence-based approaches to, 478–482

goals for, 453–454

Grinder tool, 480

inheritance, 475

instance pooling, 468

latency, 453

maintainability considerations, 9

object distribution, 455–456

portability versus, 463

presentation tier technologies, 462

profiling, 480–484

resources for, 485

response time, 452

result summaries, 468–471

scalability and, 452–453

servers, 476–477

threading models, 467–468

vertical scaling, 453

WEB (Web Application Stress) tool, 479

XML usage, 462

persistence strategies

BMP (Bean-Managed Persistence), 264

caching, 266–267

compile-time byte code modification, 266

connection factories, 272–273

data access solutions, 263–264

data mappers, 262

disassociation and, 280

domain object behaviors, 268

entity beans, 269–271

`getPersistenceManager()` method, 303

identity management, 281

identity maps, 263

impedance mismatch, 262

JDBC, 273–274

JDO (Java Data Objects), 278–280

lazy loading feature, 284

O/R mapping, 265–266

parameter maps, 275

`PersistenceManager` class, 264, 279–280

query languages, 263

resource handling, 272–273

runtime byte code generation, 266

savepoints, 274

snapshot comparisons, 284

SQL Maps, 275–278

state management, 280–281

transactional instances, 290

transparent persistence, 262, 266, 287–288

unit-of-work semantics, 262–263

`PersistenceManager` class, 264, 279–280

`PersistenceManagerFactory` class, 273

`PetStoreLogic` class, 60–61

PicoContainer, NanoContainer Web site, 137

placeholder keys, 184

`placeOrder()` method, 466

platform capabilities, complexity issues, 70

`PlatformTransactionManager` class, 218, 243

pluggability

cache strategies, 282

lightweight containers, 124–125

testability considerations, 415

plug-ins, Eclipse, 26, 483

pointcuts

AOP, 191, 196

Pointcut interface, 206

POJOs

business objects, 48

discussed, 47

testing, 440

transaction management, 247

political changes, complexity issues, 74–75

portability

business requirements, 28

EJB application problems, 104

IoC (Inversion of Control), 137–138

performance versus, 463

portable interfaces, DAO, 288

portals, web tier design, 403–404

portlets, web tier design, 403–404

ports, web services, 327–328

POST request, 398

post-processors, 182–184

presentation tier technologies, performance considerations, 462

`private()` method, 475

`proceed()` method, 205, 215

`processAction()` method, 403

`Product` class, 155–156

productivity

application development goals

Agile Manifesto Web site, 25

approaches to, 15–16

architectural impacts, 21–23

business requirements, 28

code generation alternatives, 17–20

empirical process benefits, 28–29

methodologies, 23–25

open source solutions, 22

problems, 14–15

source control systems, 26

template usage, 24–25

testing tools, 25–26

productivity loss, EJB, 104

`ProductManager` class, 155

`ProductManagerFactory` class, 158

profiling, performance options, 480–484

programmatically transaction demarcation, 234

programmatically usage, AOP, 219–220

programming styles, testability considerations, 415–416

proliferation of objects, EJB problems, 98

propagation behaviors, transaction management, 240

property values, Spring Framework support, 164–165

`ProxyFactoryBean` class, 213–217, 248

`proxyInterfaces` interface, 214

push dependencies, IoC, 442

Q

queries

languages, persistence strategies, 263

objects, persistence strategies, 263

optimization options, 478

`query()` method, 298

results, caching, 275

R

RAR (Resource Adapter Archive), 41

RDBMS

SQL-based access to, 40

tables, 27

`RdbmsOperation` class, 300–301

read committed isolation code, 240

read uncommitted isolation code, 240

read-only transactions

DAOs, 289

transaction management, 24

read-write instance variables, 94

reassociation, state management, 280

recording state, mock object testing, 428

`redirect()` method, 427

refactoring

support, productivity goals, 25–26

testing, 412

reference implementations, 112

`RegexMethodPointcut` class, 216

`RegexMethodPointcutAdvisor` class, 216

registry access, RMI, 311

regression testing, 412

regular expressions, naming conventions, 225

relational data, 40–41

reliability, complexity issues, 73

remote client support, 40

Remote Method Invocation. See RMI

remote transaction propagation, 237–238, 318

`RemoteException` class, 45, 98, 310, 314

`RemoteOrderService` class, 512

remoting services

internal distribution, 309

lightweight containers, 123

remote EJBs

accessing, 319–323

deploying, 324–325

remote transaction propagation, 318

Resin Enterprise support, 335

RMI (Remote Method Invocation)

business interfaces, 313–315

client-side proxy, 312

registry access, 311

remote sessions, 311

`RemoteException` class, 310, 314

RMI invoker, 315–316

sample application, 493–494

SLSBs, 92

state management, 318–319

web services

accessing, 327–331

WSDL-based, 325–327

wire protocols, 317–318

repeatable read isolation code, 240

`replay()` method, 429

replication, clustering challenges, 456

request-driven frameworks, web tier design, 368, 374–377

requests

GET, 398

handlers, 388

mapping, 389

POST, 398

required propagation code, 240

requires new propagation code, 240

`requiresNoSecurityCheck()` method, 193

Resin Enterprise support, remoting services, 335

Resource Adapter Archive (RAR), 41

resource pooling

overview, 95

thread management, 346

`ResourceLoader` interface, 180–181

`ResourceLoaderAware` interface, 180

resources

accessing, 168

AOP, 227–228

exposing, 168

performance options, 485

resource handling, persistence strategies, 272–273

setup, Spring Framework support, 165–169

testing, 448–449

thread management, 344

response time, performance, 452

restricted access problems, EJB, 103

result summaries, performance considerations, 468–471

`ResultSet` class, 59

reusable code, 125

RMI (Remote Method Invocation)

business interfaces, 313–315

client-side proxy, 312

registry access, 311

- remote sessions, 311
 - RemoteException class, 310, 314
 - RMI invoker, 315–316
 - role-based security, 95–96, 359**
 - Roman, Ed (*Mastering Enterprise JavaBeans*), 107**
 - root web application context, 177**
 - routing, clustering challenges, 456**
 - runtime byte code generation, persistence strategies, 266**
- S**
- saveOrUpdateCopy() **method, 285**
 - savepoints, 274**
 - scalability**
 - defined, 452
 - distribution and, complexity issues, 68–69
 - horizontal, 94, 453
 - performance and, 452–453
 - second-level caching, 278**
 - security**
 - AOP (Aspect Oriented Programming), 188
 - authentication, 335, 359
 - declarative security, thread management, 359–360
 - doSecurityCheck() **method, 189–190**
 - EJB (Enterprise JavaBeans), 95–96
 - password properties, 337
 - requiresNoSecurityCheck() **method, 193**
 - role-based, 95–96, 359
 - user credentials, 95–96
 - username properties, 337
 - semantics, complexity issues, 70**
 - serializable isolation code, 240**
 - ServerEndpointSupport **class, 511**
 - servers**
 - MTS (Microsoft Transaction Server), 82
 - performance considerations, 476–477
 - service endpoints**
 - business interfaces, 331–332
 - classes, 333
 - discussed, 328
 - exposure, 330–331
 - servlets, 332–334
 - service layers**
 - discussed, 31
 - in lightweight containers, 34–35
 - limitations, 35
 - .NET serviced components, 33–34
 - remote client support, 40
 - SFSBs (stateful session beans), 32
 - SLSBs (stateless session beans), 32
 - ServicedComponent **class, 33**
 - ServletContextListener **class, 273**
 - ServletFilter **class, 273**
 - servlets**
 - endpoints, 332–334
 - listeners and startup, 516–517
 - web tier design, 366–368
 - SessionFactory **class, 168, 174**
 - sessions, getSession() method, 282**
 - setApplicationContext() **method, 178**
 - setBeanFactory() **method, 178**
 - setDataSource() **method, 169**
 - setInventoryManager() **method, 156**
 - setOrderDao() **method, 296**
 - setReadOnly() **method, 241**
 - setResourceLoader() **method, 178**
 - setReturnValue() **method, 429**
 - setRollbackOnly() **method, 91, 209, 226, 248, 441**
 - Setter Injection, IoC, 131–133**
 - setter() **method, 283**
 - setTransactionManager() **method, 244**
 - setUp() **method, 417, 438**
 - setUpAddParameter() **method, 427**
 - SFSBs (stateful session beans)**
 - discussed, 32
 - remote sessions, 318
 - shared instances, thread management, 348–349**
 - shared resources, 324–325**
 - Shirazi, Jack (*Java Performance Tuning*), 485**
 - Simple Object Access Protocol (SOAP), 325–327**
 - simplicity considerations, architecture and implementation, 6–7**
 - single database applications, transaction management, 257–258**
 - singletons**
 - implementation problems, 103
 - isSingleton() **method, 149**
 - testing considerations, 418–419
 - web tier design, 369
 - sites. See Web sites**
 - sketching approach, testing considerations, 434**
 - SLSBs (stateless session beans)**
 - clustering services, 92–94
 - component interface, 321
 - declarative transaction management services, 90–92
 - discussed, 32
 - importance of, 89
 - local, 90
 - remoting services, 92
 - thread management, 343
 - transaction management, 235–236
 - web tier design, 369
 - snapshot comparisons, 284**
 - SOAP (Simple Object Access Protocol), 325–327**
 - SomeObject **class, 216**
 - source control systems**
 - productivity goals, 26
 - testing considerations, 437
 - source-level attributes, transaction management, 498–499**
 - source-level metadata, 84–85, 220–221**
 - specialMethod() **method, 216**
 - specifications, EJB, 1–2, 81**
 - Spring Framework**
 - advantages, 39
 - AOP support, 145, 203–205
 - application context concept, 175–177
 - callbacks, 162–163, 177–178
 - complex property value support, 164–165
 - Constructor Injection, 160–162
 - dependency resolution, 159–160
 - factory bean support, 171–172
 - file resources, 180–182
 - IoC (Inversion of Control), 135–137
 - JDBC abstraction, 145
 - JNDI resource availability, 172–173
 - local connection factory, 173–174
 - motivations for, 144
 - overview, 146–147
 - resource setup, 165–169
 - sub-frameworks, 145–146
 - template resources, 24

Spring Framework (continued)

transaction management, 145, 239–243
Web applications, 147–149
Web site, 10–11

SQL

SQL Maps, iBatis, 264, 275–278
SQL-based access, RDBMS, 40
SQLException class, 274
SqlUpdate class, 300

standards, competition versus command, 112–116

startup and listener servlets, 516–517

state management

persistence strategies, 280–281
remoting services, 318–319

stateful session beans. See SFSBs

stateless session beans. See SLSBs

static methods, testing considerations, 420

static stubs, service endpoints, 328

static templates, web tier design, 406

store operations, DAO, 290

StoredProcedure class, 300

storeLineTimes() method, 289

storeOrder() method, 286, 289

Strategy design pattern, testing considerations, 421–423

strings

concatenation, 474
String values, 164–165

Struts framework, web tier design, 375–379

stubbing, unit testing techniques, 425–426

SUCCESS constant, 382

support

application events, 175
concurrency, thread management, 349–353
EJB (Enterprise JavaBeans), 82, 104–105
language-level, thread management, 344
messages, 175
for remote clients, 40
supports propagation code, 240

SwarmCache product, 267

Swing classes, event-driven frameworks, 404

synchronization, thread management, 350–351

T

Tangosol Coherence caching technology, 93

Tapestry event-drive framework, 405–407

target objects, AOP, 193

TargetSource interface, 355

TDD (test driven development)

arguments against, 434–435
benefits of, 433

tearDown() method, 433, 438

technical challenges, testing, 413

templates

application development productivity goals, 24–25
getJdbcTemplate() method, 300
getJdoTemplate() method, 303
getSqlMapTemplate() method, 302, 501
HibernateTemplate class, 147, 304–305
JDBC access via, 298–300
JdbcTemplate class, 298–300
JdoTemplate class, 303
Spring Framework resources, 24
static, web tier design, 406
TransactionTemplate class, 240, 243–244

test driven development. See TDD

testing. See also performance

ad hoc applications, 414
alternate configuration, 442–443
AOP (Aspect Oriented Programming), 425
APIs, environment-specific, 424
applications, using AOP, 209
auto-generated, 444
black box, 415
Cactus testing framework, 101
case studies, 437–439
classes, 415
Clover product, 444–445
coding practices, 431–433
component-level, 417
coverage analysis, 438, 444–447
decoupling techniques, 415
do's and don'ts, 415
effective writing considerations, 430–433
equivalence partitioning, 430–431
glass box, 415
implementation details, 424
importance of, 412–413
in-container testing, 417
integration, 414
IoC (Inversion of Control), 139, 425
iterative development, 433
Jester tool, 448
Jtest tool, 444
JUnit tests, 26, 436
Law of Demeter, 423–424
library challenges, 420–421
load, 479
lookup strategies, 418
methods, 431
mutation tools, 447–448
negative tests, 430
POJOs, 440
productivity goals, 26
refactoring, 412
regression, 412
resources for, 448–449
singletons, 418–419
sketching approach, 434
source control considerations, 437
static methods, 420
Strategy design pattern, 421–423
TDD (test driven development)
 arguments against, 434–435
 benefits of, 433
technical challenges, 413
test coverage, 413
testability, ensuring, 415–419
transaction abstraction, 440–441
unit testing
 discussed, 411
 goals, 414–415
 mock objects, 426–430
 stubbing techniques, 425–426
untestable applications, 416–418
white box, 415

text values, bold and italic, 152

TextStyle class, 151–152

third-party APIs, 47

this value, 98

thread management

- assumptions behind, 342
- concurrency libraries, 352–353
- concurrency support, 349–353
- declarative security, 359–360
- instance creation, 344
- instance pooling, 347–349
- language-level support, 344
- lightweight containers, 123
- lock splitting, 351
- locking and, 345–346
- object creation and, 353
- performance considerations, 467–468
- resource management, 344
- resource pooling, 346
- shared instances, 348–349
- SLSBs (stateless session beans), 343
- synchronization, 350–351

thread pooling, 3, 346

`ThreadLocalInvokerStatus` interface, 358

`ThreadLocalTargetSource` class, 357

throws, AOP, 191**Tiles tag library, 377****timestamping, 283**

Tirsen, Jon (Nanning Aspects), 111

tools

- ApacheBench, 480
- Clover, 26, 444–445
- complexity issues, 72–73
- IDE, proliferation of classes, 98
- JMeter, 479
- JUnit, 26
- testing, application development, 25–26
- XDoclet, 98

TopLink O/R mapping tool, 239, 269

`toString()` method, 474, 481

TPS (transactions per second), 479**traditional architectures, 38****transactional instances, persistence strategies, 290**

`TransactionAttribute` interface, 240

`TransactionAttributeSource` class, 223–224

`TransactionFactoryBean` class, 217–218

`TransactionInterceptor` class, 209

`TransactionProxyFactoryBean` class, 217–218

transactions

- abstraction, testing considerations, 440–441
- declarative
 - demarcation, 234
 - management services, 90–92
- transaction management
 - access code, 245
 - AOP (Aspect Oriented Programming), 465–466
 - business layer facades, 235
 - central transaction infrastructure, 241
 - CMT (Container Managed Transactions), 234–235
 - data access methods, 232
 - `DataSourceTransactionManager` class, 253–255
 - declarative transaction demarcation, 234
 - distributed transaction requirements, 258
 - EJB BMT (EJB Bean Managed Transactions), 236
 - `HibernateTransactionManager` class, 256–257
 - high-level, 231–232
 - interceptors, 246–249
 - isolation level, 240
 - `jdoTransactionManager` class, 255–256

- JTA (Java Transaction API), 232
- `JtaTransactionManager` class, 253
- lightweight infrastructure, 238–239
- `PlatformTransactionManager` class, 218, 243
- POJOs, 247
- programmatic transaction demarcation, 234
- propagation behaviors, 240
- read-only transactions, 241
- remote transaction propagation, 237–238
- `setTransactionManager()` method, 244
- single database applications, 257
- SLSBs (stateless session beans), 235–236
- source-level attributes, 498–499
- Spring Framework, 145
- strategies, 251–255
- 2PC (2-Phase-Commit), 233
- UML diagram, 242

transactions per second (TPS), 479

`TransactionStatus` class, 226, 241

`TransactionTemplate` class, 240, 243–244

transfer objects, disadvantages of, 27**transparent persistence, 262, 266, 287–288**

`Transport` class, 420

try block, 59

2PC (2-Phase-Commit), transaction management, 233

U**UML diagram, transaction management, 242**

`UnauthorizedException` object, 227

`UnicastRemoteObject()` method, 311

unit testing

- discussed, 411
- goals, 414–415
- mock objects, 426–430
- stubbing techniques, 425–426

unit-of-work semantics, persistence strategies, 262–266**unnecessary code, complexity issues, 65**

`UnsupportedOperationException()` method, 439

`update()` method, 300

`updateInventory()` method, 286

`updateProduct()` method, 276

user credentials, security management, 95–96**username properties, security, 337**

`UserTransaction` interface, 236–237

V**validation**

- architectures, 28
- declarative, 377
- web tier design, 374

value-added services, lightweight containers, 123

`ValueHolder` class, 284

variables, environment, 47**Velocity Web site, 401****vendor extensions, leveraging, 288**

`verify()` method, 427–428

vertical scaling, performance, 453

`View` interface, 391

view technologies, web tier design, 401–402

`ViewItemAction` class, 505

`ViewItemController` class, 505

Virtual Shopping Mall, OTN, 55–57**visit objects, 406****Visual Basic language, 23**

W

WAR (Web Application Archive), 325, 404

WAS (Web Application Stress) tool, 479

weaving, AOP, 192

web

- applications, Spring Framework, 147–149
 - interfaces, 38–39
 - web layers, DAO, 285
 - web tier design
 - application architecture, 368–372
 - architectural issues, 364–365
 - binary formats, 402
 - business objects, 369
 - clustering challenges, 457–458
 - command-driven controllers, 394–397
 - control logic, 366
 - controllers, 374
 - discussed, 363
 - event-driven frameworks, 404–408
 - form-driven controllers, 397–399
 - interceptors, 374
 - IoC-based middle tier, 371–373
 - markup generation, 408
 - MVC approach, 366
 - naive approaches to, 365
 - open source frameworks, 368
 - portals, 403–404
 - portlets, 403–404
 - request-driven frameworks, 368, 374–377
 - servlets, 366–368
 - singletons, 369
 - SLBs (stateless session beans), 369
 - static templates, 406
 - Struts framework, 375–379
 - validation, 374
 - view technologies, 401–402
 - WebLogSystem class, 386
 - WebWork2 framework, 381–385
 - wizard-style workflows, 399
 - web views, lazy loading feature, 292
- Web Service Description Language (WSDL), 325–327**
- web services**
- accessing, 327–331
 - ports, 327–328
 - WSDL-based, 325–327
 - XML-based, 85
- Web sites**
- Agile Manifesto, 25
 - Apache, 26

- Apache Avalon, 130
 - AspectWerkz, 201
 - DynaMock, 429
 - EasyMock, 428–429
 - EMCA, 112
 - FireStorm, 16
 - FreeMarker, 401
 - Hibernate, 264
 - iBatis, 263, 278
 - JOTM, 63
 - Maven, 26
 - MockEJB, 417
 - Nanning Aspects, 207
 - NanoContainer, 137
 - Spring Framework, 10–11
 - Tapestry, 405
 - Velocity, 401
 - WebMacro, 401
 - WebObjects, 404
 - WingS, 404
- WebApplicationContextUtils class, 148, 334, 387**
- WebLogSystem class, 386**
- WebMacro Web site, 401**
- WebObjects Web site, 404**
- WebWork2, web tier design, 381–385**
- white box testing, 415**
- wildcard syntax, naming conventions, 225**
- WingS Web site, 404**
- wire protocols, remoting services, 317–318**
- wizard-style workflows, web tier design, 399**
- WSDL (Web Service Description Language), 325–327**

X

XDoclet tool, 98

XML

- data binding, 462
 - performance considerations, 462
 - populating JavaBeans via, 151–153
 - web services, 85
- XmlBeanFactory class, 154–155**
- XP (Extreme Programming), 86**
- XWork, WebWork2, 382–383**

Z

Zadrozny, Peter (*J2EE Performance Testing with BEA WebLogic Server*), 485