

Index

SYMBOLS

- * (asterisk), in ant-style paths, 93
- { } (braces), JDBC SQL escape syntax, 175

A

- abstract attribute, bean element, 74
- abstract parent bean definition (template), 73
- abstract schema advice, 125
- AbstractAccessDecisionManager class, Acegi, 383–384
- AbstractAutoProxyCreator class, 133, 143–144
- AbstractController class
 - definition of, 459
 - example of, 423–424, 464–466
- AbstractDependencyInjectionSpringContextTests class, 107
- AbstractEnterpriseBean class, 407
- AbstractExcelView class, 517
- abstraction for services, 14–18, 67–71
- AbstractJExcelView class, 516–517
- AbstractJmsMessageDrivenBean class, 411
- AbstractMessageDrivenBean class, 411
- AbstractPoiExcelView class, 516–517
- AbstractPoolingTargetSource interface, 150
- AbstractScript class, 365
- AbstractScriptFactory interface, 365
- AbstractSecurityInterceptor class, Acegi, 385, 387
- AbstractSessionBean class, 407
- AbstractStatefulSessionBean class, 410, 412
- AbstractStatelessSessionBean class, 407, 408, 412
- AbstractTransactionalDataSourceSpringContextTests class, 107
- AbstractTransactionalSpringContextTests class, 107
- AbstractView class, 493–496, 520
- AbstractWizardFormController class, 474–477
- AbstractXsltView class, 514
- AcceptHeaderLocaleResolver class, 455
- access control list (ACL), 368, 371, 396
- AccessDecisionManager interface, Acegi, 383–384
- AccessDecisionVoter interface, Acegi, 383–384
- accessors, for lightweight remoting
 - for Burlap, 310–312
 - configuring, 307–308
 - definition of, 306
 - for Hessian, 310–312
 - for HTTP invoker, 314–315
 - for RMI, 318–320
 - for RMI-IIOP, 323
 - for Web Services with JAX-RPC, 326–328
- account manager example, 115–118
- Acegi Security
 - authentication using, 369, 377–383
 - authorization using, 383–386
 - business services layer security using, 368, 370–371
 - container adapters, 381
 - definition of, 13, 368–369, 594–595
 - domain object security using, 368, 371–372, 386–388
 - example of, 389–395
 - integration with JAAS, 373
 - Servlet Specification and, 380–381
 - web request security using, 368, 369–370
 - when to use, 396
- ACID requirements of transactions, 218–220
- ACL (access control list), 368, 371, 396
- AclEntry interface, Acegi, 386
- AclManager interface, Acegi, 386–388
- AclManager repository, 371
- AclProvider interface, Acegi, 386–387
- AclProviderManager class, Acegi, 386
- Action class, 537–538
- ActionServlet class, 539
- ActionSupport class, 538
- addAdvice property, AdvisedSupport class, 136
- addAdvisor property, AdvisedSupport class, 136
- advice, AOP
 - abstract schema advice, 125
 - before advice, 122–123
 - after returning advice, 123–124
 - custom advice types, 125
 - definition of, 12, 120
 - method interceptors, 120–121
 - throws advice, 124–125
 - typed advice, 125
 - where to apply (pointcuts), 125–131
 - wrapped in an Advisor, 131
- Advice interface, 121–122
- Advised interface, 146, 160
- AdvisedSupport class, 132–133, 135–137
- advisorAdapterRegistry property, AdvisedSupport class, 137
- advisorChainFactory property, AdvisedSupport class, 136
- Advisors, AOP, 130–131
- after returning advice, 123–124
- AfterInvocationManager interface, Acegi, 385–386

afterPropertiesSet() method, InitializingBean Interface

`afterPropertiesSet()` **method**, InitializingBean **interface**, **65, 67**

`afterReturning()` **method**, AfterReturningAdvice **interface**, **123**

AfterReturningAdvice **interface**, **122, 123–124**

`afterThrowing()` **method**, ThrowsAdvice **interface**, **124**

aggregate function (count) methods, DAO, 24

agile J2EE, 8

`allowDirtyBack` **property**, AbstractWizardFormController **class**, **475**

`allowDirtyForward` **property**, AbstractWizardFormController **class**, **475**

Alur, Deepak (Core J2EE Patterns), 172, 424

ant-style paths for resource location strings, 93

AOP Alliance API, 120–121

AOP (Aspect-Oriented Programming) framework

advantages of, 118–119

advice, 119–125

advisors, 130–131

alternatives to Spring's AOP framework, 163–169

benefits of, 13–14

books about, 597

debugging using, 159–161

definition of, 5, 11–12, 37, 578

disadvantages of, 119

example using, 114–118

goals of, 113–114

guidelines for, 170, 592

integration with, 20–21

interceptor chain, 118, 153

introduction, 12, 153–156

MethodInvocation, exposing, 157

multiple AOP framework interfaces, 161

pointcuts, 125–131

proxy-based, 12, 34, 114, 118

target object, 118–119, 147–153

testing using, 159–161

AOP proxies

autoproxying, 141–145, 170, 591

configuration of, 132–137

creating using ProxyFactoryBean interface, 137–140

definition of, 132

double proxying, 161

examining and modifying at runtime, 145–146

exposing current proxy, 156–157

lifecycle of, 140–141

programmatic creation of, 146–147

serializable, 162–163

types of, 158–159

AopProxyFactory **interface**, **158**

`aopProxyFactory` **property**, ProxyConfig **class**, **135**

AopUtils **class**, **145**

Apache Avalon, 4

Apache Cactus tool (Jakarta), 106, 421

Apache Commons DBCP, 199, 252

Apache Commons Pool, 150

Apache Jakarta Tapestry. See Tapestry web framework

Apache OJB (Object relational Bridge)

DAO support class for, 173

definition of, 255

integration with, 19

object query language used by, 257

O/R mapping and, 299–300

transaction manager for, 247–248

Apache Tomcat server, 552, 574, 575

APIs, simplified use of, 5

application

components of, handling, 97–99

structuring, guidelines for, 589–592

application context

accessing beans, 52–53

declarative usage of, 93–95

definition of, 48–49, 578

definitions of

in multiple files, 95–96

programmatic, 110

in Properties format, 108–109

XML, 52, 53, 57

event publishing and subscription with, 89–91

integration testing with, 106–108

interface hierarchy for, 50

loading, 50–52

as message source, 87–89

as resource loader, 84–87

singleton access of, 99

testing controllers using, 484–485

when to use, 49

application framework. See also Spring Framework; web frameworks

advantages of, 2–3

definition of, 1

lightweight frameworks, 3–4

non-invasive, 6

application objects

as named services, 7

as POJOs, 7

application server

running JDBC framework in, 199–201

testing applications deployed in, 108

transaction management and, 200–201, 220, 249–251

when to use, 581

ApplicationContext **interface**

definition of, 48–50

retrieving from ServletContext, 535

ApplicationContextAware **interface**, **65, 67**

ApplicationEvent **class**, **50, 89–91**

ApplicationEventPublisher **interface**, **50, 90**

ApplicationEventPublisherAware **interface**, **65**

ApplicationListener **interface**, **91**

`applyBeanPropertyValues()` **method**, Autowire

CapableBeanFactory **interface**, **48**

`applyCommonInterceptorsFirst` **property**,

AbstractAutoProxyCreator **class**, **143**

appserver

running JDBC framework in, 199–201

testing applications deployed in, 108

transaction management and, 200–201, 220, 249–251

when to use, 581

architecture

layers of, 21–23, 583–588

replaceability of layers in, 7

around advice, AOP, 118, 121
arrays, data binding and, 463
aspect association types, AspectJ, 167
AspectJ AOP framework
 advantages of, 168
 AOP Alliance and, 120
 disadvantages of, 168
 integration with, 20–21, 164–167
 using with Spring's AOP framework, 168–169
AspectJ in Action (Laddad), 170, 597
Aspect-Oriented Programming (AOP) framework.
 See also AOP proxies
 advantages of, 118–119
 advice, 119–125
 advisors, 130–131
 alternatives to Spring's AOP framework, 163–169
 benefits of, 13–14
 books about, 597
 debugging using, 159–161
 definition of, 5, 11–12, 37, 578
 disadvantages of, 119
 example using, 114–118
 goals of, 113–114
 guidelines for, 170, 592
 integration with, 20–21
 interceptor chain, 118
 introduction, 12, 153–156
 MethodInvocation, exposing, 157
 multiple AOP framework interfaces, 161
 pointcuts, 125–131
 proxy-based, 12, 114, 118
 target object, 118–119, 147–153
 testing using, 159–161
Aspect-Oriented Programming (AOP) framework (module)
 interceptor chain, 153
 proxy-based, 34
aspects (concerns), 12, 13
AspectWerkz AOP framework, 20–21, 169
asterisk (*), in ant-style paths, 93
asynchronous messaging. See JMS (Java Message Service)
atomicity of transactions, 218–219
attachments, email, 360
attribute-driven autoproxying, 144
auditing, 13
Aurora MVC, 595
authentication
 Acegi Security
 example for, 391–392
 features for, 369, 377–383
 definition of, 367, 368
 JAAS features for, 373
 Servlet Specification features for, 375–376
Authentication interface, Acegi, 377, 378
authentication repositories, 368
AuthenticationDao interface, Acegi, 377, 379
AuthenticationEvent class, 372
AuthenticationManager interface, Acegi, 369, 377, 378, 385
AuthenticationProcessingFilter module, Acegi, 380

AuthenticationProvider interface, Acegi, 377, 378, 379
authorization
 Acegi Security
 example for, 392–395
 features for, 383–386
 definition of, 367
 JAAS features for, 373–374
 Servlet Specification features for, 376–377
auto wiring, 542–543
autoproxying, 141–145, 170, 591
autowire() method, AutowireCapableBean Factory interface, 48
autowireBeanProperties() method, AutowireCapableBeanFactory interface, 48
AutowireCapableBeanFactory interface, 48, 50
autowiring dependencies, 61–63, 590–591
Avalon (Apache), 4

B

BadSqlGrammarException class, JDBC, 181, 182
BasicAclAfterInvocationProvider module, Acegi, 388
BasicAclEntryAfterInvocationCollection FilteringProvider module, Acegi, 388
BasicAclEntryVoter module, Acegi, 388
BasicAclProvider class, Acegi, 386–387
BasicProcessingFilter module, Acegi, 380
batch updates, JDBC, 208–209, 214
BatchSqlUpdate class, JDBC, 209
Bauer, Christian (Hibernate in Action), 269
bean element
 abstract attribute, 74
 class attribute, 54–55
 definition of, 53
 destroy-method attribute, 66–67
 factory-method attribute, 54–55
 id attribute, 54
 init-method attribute, 65, 66–67
 lazy-init attribute, 57
 name attribute, 54
 nesting, 61
 singleton attribute, 55–56
bean factory. See also application context
 accessing beans, 52–53
 definition of, 47, 578
 definitions of
 in multiple files, 95–96
 programmatic, 110
 in Properties format, 108–109
 XML, 52, 53, 57
 integration testing with, 106–108
 interface hierarchy for, 47–48
 loading, 50–52
 singleton access of, 99
 when to use, 49
bean post-processors, 65, 66, 75–80, 141
BeanFactory interface, 47, 50
BeanFactoryAware interface, 65, 67
BeanFactoryPostProcessor interface, 76–77
BeanFactoryUtils class, 48

beanMapping tag

beanMapping **tag**, 330

BeanNameAutoProxyCreator **interface**

definition of, 80, 141–142

transaction management and, 237–238

when to use, 145

BeanNameAware **interface**, 65, 354

beanNames **property**, BeanNameAutoProxyCreator **interface**, 142

BeanNameUrlHandlerMapping **class**, 442

BeanNameViewResolver **class**, 451

BeanPostProcessor **interface**, 75, 141

beans

accessing, 52–53

creation mechanism for, 54–55

defining, 53–56

definitions, reusing, 72–74

dependencies

autowiring, 61–63, 590–591

manual, 61

specifying, 58–63

validating, 64

destruction process for, 66–67

factory beans, 69–71, 99–102

identifier for, 54

inheriting, 591

initialization process for, 56–57, 64, 66–67

lifecycle of, managing, 64–67

multiple components referring to, 97–99

nesting, 61

non-singleton beans, 55–56, 67, 140–141

singleton beans, 55–56, 99, 140

beans **element**

default-autowire attribute, 62

default-lazy-init attribute, 57

definition of, 53

Beanshell scripts

definition of, 363–365

when to use, 582–583

before advice, 122–123

before() **method**, BeforeAdvice **interface**, 122

BeforeAdvice **interface**, 122–123

Begin, Clinton (*IBATIS SQL Maps Developer Guide*), 261

Bergsten, Hans (*ONJava articles*), 496

Binary Large Object (BLOB)

definition of, 203–204

example of, 204–206

mapping in Hibernate, 285–287

bindOnNewForm **property**, SimpleFormController **class**, 468

BindStatus **class**, 499–501

BLOB (Binary Large Object)

definition of, 203–204

example of, 204–206

mapping in Hibernate, 285–287

Boner, Jonas (“*Spring and AspectWerkz – A Happy Marriage*”), 169

books. *See* publications

braces {}, JDBC SQL escape syntax, 175

Brown, Simon (*Pro JSP Third Edition*), 496

bulk update methods, 24

Burlap, remoting using

accessing a service, 310–312

definition of, 306, 309

exception handling, 308

exporting a service, 312–313

support classes for, 309–310

when to use, 310, 333

BurlapProxyFactoryBean **class**, 309, 310–312

BurlapServiceExporter **class**, 309

Business Interface pattern, 403

business services layer

definition of, 23, 27, 583

guidelines for, 585–587

security for, 27, 368, 370–371

ByteArrayPropertyEditor **class**, 102

C

CachedRowSet **interface**, JDBC, 188–189

cacheSeconds **property**, WebContentInterceptor **class**, 448

caching

AOP used for, 13

as crosscutting concern, 113–114

views, 522

Cactus tool (Jakarta Apache), 106, 421

CallableStatement **interface**, JDBC, 174

CannotAcquireLockException **class**, JDBC, 181

Canoo WebTest tool, 106, 485

CasProcessingFilter **module**, Acegi, 380

Caucho Technology. *See* Burlap, remoting using; Hessian, remoting using

Cayenne O/R mapping project, 302, 596

CGLIB proxy, 145, 158–159

Cglib2AopProxy **interface**, 158

chaining

interceptors, 118, 153

view resolvers, 452–454

change detection, O/R mapping, 258, 268

Character Large Object (CLOB)

definition of, 203–204

mapping in Hibernate, 285–287

checkboxes, data binding and, 463

child bean definitions, 73, 74

ChildBeanDefinition **class**, 110

class attribute, bean element, 54–55

class weaving AOP, 114

ClassEditor **class**, 57, 102

classes. *See also specific classes*

programming to interfaces instead of, 7

self-documenting, 9

ClassFilter **interface**, 126–127

classpath: prefix for resource location string, 85, 92

ClasspathResource **class**, 85, 92

classpaths, using as resources, 84–85, 92

ClasspathXmlApplicationContext **interface**

constructors taking resource location paths, 92

resource access and, 85

client-server remoting, 305

CLOB (Character Large Object)

definition of, 203–204

mapping in Hibernate, 285–287

CMT (container-managed transactions), 219, 224

code reuse, Spring Framework support for, 6–7

- Colyer, Adrian** (*Eclipse AspectJ: Aspect-Oriented Programming with AspectJ and the Eclipse AspectJ Development Tools*), 170, 597
- Command design pattern**, 12
- `commandClass` property, `SimpleFormController` class, 468
- `commandName` property, `SimpleFormController` class, 469
- comma-separated value (CSV)**, 493–494
- Commons DBCP** (Apache), 199, 252
- Commons Pool** (Apache), 150
- `CommonsPathMapUrlHandlerMapping` class, 444–445
- communication resource manager (CRM)**, 220
- components of application, handling**, 97–99
- concerns (aspects)**, 12, 13. *See also* **Aspect-Oriented Programming (AOP) framework**
- concurrency control for transactions**, 220
- `ConfigAttribute` interface, `Acegi`, 383–384
- `ConfigurableBeanFactory` interface, 48, 50
- configuration**
- externalizing, 7, 9
 - guidelines for, 589–592
- `Connection` interface, **JDBC**, 174
- connection pooling**
- definition of, 199–200
 - transaction data source declaration and, 251–254
- connection wrappers, with application servers**, 200–201
- `ConnectionFactory` interface, 340
- Constructor Injection**
- definition of, 9, 589
 - example of, 10, 43–44
 - specifying dependencies and, 56
 - when to use, 45–47
- `constructor-arg` element, 58, 63
- `ConstructorArgumentValues` class, 110
- container**. *See* **application context**; **bean factory**;
- Inversion of Control (IoC) container**
- container adapters**, `Acegi`, 381
- container-managed transactions (CMT)**, 219, 224
- `contentType` property, `AbstractView` class, 520
- `contextClass` parameter, `DispatcherServlet` class, 432
- `ContextClosedEvent` event, 90
- `contextConfigLocation` parameter, `DispatcherServlet` class, 432
- `ContextHolder` class, 381–383
- `ContextLoader` class, 93–94, 95, 416–419
- `ContextLoaderListener` class, 93
- `ContextLoaderPlugIn` class, 540
- `ContextLoaderServlet` class, 93–94, 434
- `ContextRefreshEvent` event, 89
- `ContextSingletonBeanFactoryLocator` class, 99, 414–416, 419–420
- control flow pointcuts**, 129
- `ControlFlowPointcut` class, 129, 162
- Controller interface**, 429, 435, 440
- controller objects**. *See also* **Controller interface**
- definition of, 28, 426
 - example of, 464–474
 - JSF features for, 531
 - provided by Spring, 429
 - Spring MVC features for, 530–531
 - Struts features for, 530
 - Tapestry features for, 531
 - testing, 481–485
 - types of, 458–461
 - WebWork features for, 530
 - for wizard-style forms, 474–477
- `CookieLocaleResolver` class, 455
- CORBA IIOP protocol**, 323–324
- Core J2EE Patterns** (Alur, Crupi, Malks), 172, 424
- COS package**
- sending mail, 356
 - uploading files, 478–479
- count (aggregate function) methods, DAO**, 24
- `createAopProxy()` method, `AopProxyFactory` interface, 158
- credentials**, 367
- criteria queries, Hibernate**, 257
- CRM (communication resource manager)**, 220
- `CronTriggerBean` class, 352
- crosscutting concerns**, 113–114
- cross-platform programming model, support for**, 31
- Crupi, John** (*Core J2EE Patterns*), 172, 424
- CSV (comma-separated value)**, 493–494
- curly braces ({}), JDBC SQL escape syntax**, 175
- `currentInvocation()` method, `ExposeInvocationInterceptor` class, 157
- `currentProxy()` method, `AopContext` class, 156
- custom advice types**, 125
- `CustomBooleanEditor` class, 102
- `CustomDateEditor` class, 79–80, 103
- `CustomEditorConfigurer` interface, 78–80
- `CustomNumberEditor` class, 102
- `CustomSQLExceptionCodesTranslation` class, **JDBC**, 201–202
- `customTargetSourceCreators` property, `AbstractAutoProxyCreator` class, 143

D

- DAO (Data Access Object) interface**
- Hibernate, 270–273
 - iBATIS SQL Maps, 263–265
 - JDO API, 289–291
 - O/R mapping, 259–260
- DAO (Data Access Object) interface layer**, 23, 24–26, 583
- DAO (Data Access Object) pattern**, 172–173
- `DaoAuthenticationProvider` class, `Acegi`, 377, 379
- data access abstraction**, 5, 34, 579
- data access layer**. *See also* **database layer**
- definition of, 583–585
 - for sample application, 177–182
- Data Access Object (DAO) interface**
- Hibernate, 270–273
 - iBATIS SQL Maps, 263–265
 - JDO API, 289–291
 - O/R mapping, 259–260
- Data Access Object (DAO) interface layer**, 23, 24–26, 583
- Data Access Object (DAO) pattern**, 172–173

data binding, MVC

- definition of, 461–463
- JSF features for, 528
- Spring MVC features for, 526–527
- Struts features for, 526
- Tapestry features for, 527
- WebWork features for, 525–526

data definition language (DDL), 178, 190

data disconnection, 588

data mappers, 256–257

data source declaration for transactions

- choosing, 253–254
- JNDI data sources, 253
- local pooled data sources, 252–253
- local unpooled data sources, 251–252

DataAccessException class, 15–16, 177, 180–182, 259

DataAccessResourceFailureException class, JDBC, 181, 182

database. See also iBATIS SQL Maps; O/R (Object-Relational) mapping

- connection pooling, 199–200, 251–254
- connection wrappers, with application servers, 200–201
- SQL queries, JDBC
- features for, 175
 - with `JdbcTemplate` class, 183–189, 204–206, 214
 - with RDBMS operation classes, 189–194, 206–209
- statement wrappers, with application servers, 200–201

database layer (EIS tier), 23, 583. See also data access layer

DataIntegrityViolationException class, JDBC, 181, 259

DataRetrievalFailureException class, JDBC, 181

DataSource interface, JDBC, 68–69, 174, 175, 178–180

DataSourceTransactionManager class

- definition of, 243–245, 266
- example of, 222–224, 266–267
- when to use, 225, 226–227, 230

DataSourceUtils class, 244

DDL (data definition language), 178, 190

debugging, using AOP, 159–161

declarative transaction demarcation

- `BeanNameAutoProxyCreator` interface, 237–238
- definition of, 225, 227, 233
- guidelines for, 242
- performance of, 242
- `ProxyFactoryBean` interface and transaction interceptor, 233–235
- source-level metadata
 - using commons attributes, 238–241
 - using Java 5.0 annotations, 241–242
- `TransactionProxyFactoryBean` interface, 235–236

declarative transaction management, 13, 219, 222, 224

declarative usage of application contexts, 93–95

Decorator design pattern, 12

DefaultAdvisorAutoProxyCreator interface

- definition of, 80, 141, 142
- security and, 370
- when to use, 145

default-autowire attribute, beans element, 62

DefaultIntroductionAdvisor class, 155

DefaultJdoDialect interface, 297

default-lazy-init attribute, beans element, 57

DefaultPointcutAdvisor class, 131

DefaultTransactionAttribute class, 226

DefaultTransactionDefinition class, 226, 231

deferred close mode, Hibernate Open Session in View, 283

DelegatingIntroductionInterceptor class, 155

DelegationActionProxy class, 539–542

DelegationRequestProcessor class, 539–542

delete methods, DAO, 24

dependencies

- autowiring, 61–63, 590–591
- manual, 61
- self-documenting, 9
- specifying, 56, 58–63
- validating, 64

Dependency Injection

- with AspectJ AOP, 164–167
- books about, 597
- Constructor Injection
 - definition of, 9, 589
 - example of, 10, 43–44
 - specifying dependencies and, 56
 - when to use, 45–47
- definition of, 8–9, 37, 43
- example of, 10, 43–45
- Method Injection
 - definition of, 9, 589
 - example of, 44–45
 - testing and, 45
 - when to use, 44, 590
- migration to other lightweight containers using, 6
- Setter Injection, 9
 - definition of, 589
 - example of, 10, 41–43
 - specifying dependencies and, 56
 - when to use, 45–47, 590
- types of, 9, 589

Dependency Lookup, 11

design, Object Oriented

- books about, 597
- crosscutting concerns not addressed by, 114
- guidelines for, 589
- Spring Framework support for, 7

design patterns

- book about, 597
- Command design pattern, 12
- Decorator design pattern, 12
- inappropriate use of, 2
- Observer design pattern, 12

Design Patterns (Gamma, Helm, Johnson, Vlissides), 597

destroy() method, DisposableBean interface, 66, 67

destroy-method attribute, bean element, 66–67

DestructionAwareBeanPostProcessor interface, 66, 75

detached objects

- Hibernate, 268
- JDO, 288

dirty checking domain objects, 113–114

dirty reads, 219, 229
dispatcher, 427, 430
Dispatcher Servlet (Front Controller), 424, 430
 DispatcherServlet **class**, 430, 431–434
 DisposableBean **interface**, 66, 67
distributed (global) transactions, 221
distributed objects, when to use, 2, 582
DNS (domain name system), 389, 395, 396
document type definition (DTD), for bean definitions, 53
document-based views, 515–519
Domain Driven Design (Evans), 597
domain name system (DNS), 389, 395, 396
domain objects, security for, 368, 371–372, 386–388
 doSubmitAction() **method**, SimpleForm
 Controller **class**, 470
double proxying, 161
 DriverManagerDataSource **interface**, JDBC, 179,
 251–252
DTD (document type definition), for bean definitions, 53
dynamic objects, 13
dynamic pointcuts, 127, 129–130
dynamic proxies, J2SE, 12
 DynamicDestinationResolver **class**, 343
 DynamicMethodMatcherPointcut **class**, 129–130
 DynamicScript **interface**, 362
DynAOP, 120

E

EasyMock library, 105, 483
Eclipse AspectJ: Aspect-Oriented Programming with AspectJ and the Eclipse AspectJ Development Tools (Colyer), 170, 597
Eclipse, Spring IDE for, 595
EIS (Enterprise Information System) tier, 23, 583
EJB CMT, transaction manager for, 250
EJB Expert Group, 4
EJBs
 accessing
 with Home object lookups, 400–402
 with Stateless Session Bean lookups, 402–406
 without Spring, 399–400
 disadvantages of, 2, 3–4
 evolution of, Spring Framework and, 29–30
 future of, 398, 599
 implementing
 Message Driven Beans, 411–412
 Stateful Session Beans, 410–411
 Stateless Session Beans, 407–410
 using XDoclet, 412
 inappropriate use of, 2
 local EJBs, 402, 403–404
 local stateless session EJBs, pooling as alternative to,
 151
 for providing services to objects, 12
 remote EJBs, 402–403, 404–406
 remoting by, using, 28, 307, 308
 simplified use of, 34–35
 singleton container access for, 99, 413–420
 Spring Framework and, 38, 581–582
 testing, 420–422
 transaction attributes, 222
 when to use, 397–398
EL (Expression Language)
 for indexed properties, 463
 Spring tags supporting, 503–504
email
 attachments, 360
 COS, sending mail using, 356
 creating messages, 358, 359
 example of, 354–355
 existing mail session, reusing, 355–356
 features for, 354
 mail manager for, 356–360
 recipients, handling, 358–359
 sending messages, 360
enterprise applications
 crosscutting concerns of, 113–114
 security requirements for, 367–368
 separate target objects for, 119
 services support for, 335
 stored procedures and, 196
Enterprise Information System (EIS) tier, 23
entity beans
 checked exceptions and, 16
 disadvantages of, 2
 persistent domain objects replacing, 585
 POJO persistence replacing, 30
environments
 insulating application from, 6
 supported, 37
escape syntax, JDBC, 175
Evans, Eric (Domain Driven Design), 597
events, 89–91
examples. See account manager example; sample application; weather data service example
Excel-based views, 515–519
exception handling
 AOP used for, 13
 exception hierarchy for, 15–16
 exception translations, 201–203
 JDBC framework, 175–176, 180–182
 notification of exceptions, example using, 115–118
exception resolvers, MVC, 438, 445, 456–458
 ExceptionResolver **class**, 445
Expert One-on-One J2EE Design and Development (Johnson), 32, 427, 458, 597
exporters, for lightweight remoting
 for Burlap, 312–313
 configuring, 308
 definition of, 306
 for Hessian, 312–313
 for HTTP invoker, 315–316
 for RMI, 322–323
 for RMI-IIOP 324
 for Web Services with JAX-RPC, 328–330
 ExposeInvocationInterceptor **class**, 157
 exposeProxy **property**, ProxyConfig **class**, 134, 156
expression builder, TopLink, 257
Expression Language (EL)
 for indexed properties, 463
 Spring tags supporting, 503–504
Extreme Programming (XP), 18

F

factory beans, 69–71, 99–102. See also specific factory beans

FactoryBean interface, 69–71

factory-method attribute, bean element, 54–55

FieldError class, 87

FieldRetrievingFactoryBean class, 101, 157

file: prefix for resource location string, 85, 92

FileEditor class, 102

files

application components, handling, 97–99

multiple, for container definitions, 95–96

uploading, 478–481

FilesystemResource class, 85, 92

filesystems, using as resources, 84

FilesystemXmlApplicationContext interface

constructors taking resource location paths, 92

resource access and, 85

FilterSecurityInterceptor class, Acegi, 369, 385

finder methods, DAO, 24

formCheckboxes macro, 510

formInput macro, 510

formMultiSelect macro, 510

formRadioButton macro, 510

formSingleSelect macro, 510

formTextArea macro, 510

formView property, SimpleFormController class, 468

Fowler, Martin

coined Dependency Injection, 8

Patterns of Enterprise Application Architecture, 343

framework. See application framework; Spring Framework; web frameworks

FreeMarker template engine

configuring view resolver for, 506–507

constructing view using, 489–490

definition of, 505

form simplification macros, 507–510

integration with, 21

Front Controller (Dispatcher Servlet), 424, 430

frozen property, ProxyConfig class, 134

G

Gamma, Erich (*Design Patterns*), 597

Gateway pattern, JMS using, 343–345

generated keys, 194–196

GeneratedKeyHolder interface, JDBC, 196

getArguments() method, MethodInvocation interface, 161

getBean() method, BeanFactory interface, 47, 52, 70

getInputStream() method, InputStreamSource interface, 84

getLastModified() method, HandlerAdapter interface, 478

getMessage() method, MessageSource interface, 87

getObject() method, FactoryBean interface, 70

getResource() method, ResourceLoader interface, 84–85, 92

getTargetSource() method, Advised interface, 146

GETTERS constant, Pointcuts class, 127

global (distributed) transactions, 221, 260

globally unique identifiers (GUIDs), 194

GrantedAuthority interface, Acegi, 377, 378

granularity, 588

Groovy scripts

definition of, 362–363

when to use, 582–583

GUIDs (globally unique identifiers), 194

H

Hammant, Paul (coined Dependency Injection), 8

handle() method, HandlerAdapter interface, 478

handleInvalidSubmit() method, SimpleForm Controller class, 470

handler adapters, MVC, 449

handler exception resolvers, MVC, 456–458

handler interceptors, MVC, 446–449

handler mappings, MVC, 441–446

HandlerAdapter interface, 446–447, 478

HandlerExceptionResolver class, 435, 441, 454, 456–458

HandlerExecutionChain class, 446–449

HandlerInterceptor class, 435, 436–437, 446–449, 517–519

HandlerMapping class, 435, 441–446, 454

handlers, MVC, 436–439, 478

Hellesoy, Aslak (coined Dependency Injection), 8

Helm, Richard (*Design Patterns*), 597

Hessian, remoting using

accessing a service, 310–312

definition of, 306, 309

exception handling, 308

exporting a service, 312–313

support classes for, 309–310

when to use, 310, 333

HessianProxyFactoryBean class, 309, 310–312

HessianServiceExporter class, 309

Hibernate

BLOB handling, 285–287

CLOB handling, 285–287

compared to Spring Framework, 1, 3

configuration file, 274

DAO implementation for, 270–273

DAO support class for, 173

definition of, 255, 268–269, 287–288

integration with, 19

JCA connector setup, 275–276

mapping file for, 269–270

object query language used by, 257

Open Session in View pattern, 273, 282–285

reattachment and duplicate object problem, 280–281

resource management for, 16, 17–18

session lifecycle, 280

Spring setup for, 273–276

transaction management, 276–282

when to use, 287–288

Hibernate in Action (King, Bauer), 269

Hibernate Query Language (HQL), 268

- hibernate.cfg.xml **file**, 274
- HibernateDaoSupport **class**, 173
- HibernateInterceptor **class**, 281–282
- HibernateTemplate **class**, 245, 270, 272, 278
- HibernateTransactionManager **class**
definition of, 225, 226, 227, 245–246
when to use, 276
- HierarchicalBeanFactory **interface**, 47–48, 50
- Hoeller, Juergen**
J2EE Development without EJB, 169, 398, 491, 597
Spring Framework developer, 32–33
- Hollywood Principle**, 8
- Home object lookups, encapsulating**, 400–402
- hot swapping**, 13, 148–149
- HotSwappableTargetSource **class**, 148–149, 162
- HQL (Hibernate Query Language)**, 268
- HSQL**
isolation level support, 220
key generation using, 195
- HsqlMaxValueIncrementer **class**, JDBC, 195
- HTTP invoker, remoting using**
accessing a service, 314–315
additional invocation attributes for, 316
customizing, 316–317
definition of, 306, 313
exception handling, 308
exporting a service, 315–316
secure communication with, 316
support classes for, 313–314
when to use, 333
- http:// **prefix for resource location string**, 85, 92
- HttpInvokerProxyFactoryBean **class**, 313
- HttpInvokerRequestExecutor **class**, 317
- HttpInvokerServiceExporter **class**, 313
- HttpRequestIntegrationFilter **module**, Acegi, 382–383
- HttpServletRequest **class**, 479–480
- HttpServletRequestWrapper **class**, Acegi, 381
- HttpSessionIntegrationFilter **module**, Acegi, 382
- HttpUnit tool**, 485
- I**
- iBatis SQL Maps**
configuration file, 265–266
DAO implementation for, 263–265
DAO support class for, 173
definition of, 255, 257, 261, 267–268
integration with, 19
mapped statements (mapping file) for, 261–263
Spring setup for, 265–266
transaction management using, 266–267
when to use, 267, 584
- iBatis SQL Maps Developer Guide (Begin)**, 261
- id **attribute**, bean **element**, 54
- idref **element**, 59
- IIOB protocol**, 323–324
- ImageDB sample application**, 598
- Impedence Mismatch**, 256
- import **element**, 96
- initBinder() **method**, SimpleFormController **class**, 469
- InitializingBean **interface**, 65, 67
- init-method **attribute**, bean **element**, 65, 66–67
- InputStreamEditor **class**, 102
- InputStreamSource **interface**, 87
- integration testing**
AOP framework used for, 160
in application server, 421–422
container used for, 106–108
guidelines for, 593–594
support for, 18
tools for, 485
- interceptor chain, AOP**
definition of, 118
terminating early, 153
- InterceptorNames **property**, AbstractAutoProxyCreator **class**, 143
- interceptorNames **property**, AdvisedSupport **class**, 137, 138, 141–142
- interceptors**
chaining, 118, 153
JSF features for, 531
Spring MVC features for, 530–531
Struts features for, 530
Tapestry features for, 531
transaction interceptors
definition of, 233–235
Hibernate, 281–282
JDO, 294–295
in Web MVC application, 436–439, 446–449
WebWork features for, 530
- Interface21 framework**, 33
- interfaces. See also specific interfaces**
dependencies between services as, 8
programming to, instead of classes, 8
- interfaces **property**, AdvisedSupport **class**, 136
- InternalResourceView **class**, 497
- InterruptedException **interface**, 351
- intra-server remoting**, 305
- introduction, AOP**
alternatives to, 156
definition of, 12
making, 154–156
when to use, 153
- IntroductionAdvisor **interface**, 154
- IntroductionInfo **interface**, 154
- InvalidDataAccessResourceUsageException **class**, JDBC, 182
- Inversion of Control (IoC) container. See also application context; bean factory; Dependency Injection**
bean factory as, 47
definition of, 5, 8–11, 33, 40, 578
example of, 41–43
templates and, 17
- invocableClone() **method**, ReflectiveMethodInvocation **interface**, 163
- invoke() **method**, MethodInterceptor **interface**, 120
- IoC (Inversion of Control) container. See also application context; bean factory; Dependency Injection**
bean factory as, 47
definition of, 5, 8–11, 33, 40, 578
example of, 41–43
templates and, 17

`isAopProxy()` **method**, `AopUtils` **class**, **145**
`isCglibProxy()` **method**, `AopUtils` **class**, **145**
`isFormChangeRequest()` **method**, `SimpleFormController` **class**, **470**
`isFormSubmission()` **method**, `SimpleFormController` **class**, **470**
`isJdkDynamicProxy()` **method**, `AopUtils` **class**, **145**
isolation of transactions, **219–220**, **228–229**
ISOLATION_DEFAULT option for transactions, **229**
ISOLATION_READ_COMMITTED option for transactions, **229**
ISOLATION_READ_UNCOMMITTED option for transactions, **229**
ISOLATION_REPEATABLE_READ option for transactions, **229**
ISOLATION_SERIALIZABLE option for transactions, **229**
`isRuntime()` **method**, `MethodMatcher` **interface**, **127**
`ItemView.html` **file**, **545–547**
`ItemView.page` **file**, **544–545**
iText library, constructing view using, **490–491**

J

JAAS (Java Authentication and Authorization Service)

advantages of, 373–375
definition of, 372–373
integration with Acegi Security, 373
when to use, 375

Jakarta Apache Cactus tool, **106**, **421**

Jakarta Commons Attributes, **238–241**

Jakarta Commons FileUpload package, **478**

Jakarta POI, **516**

JARs, list of, **36**

Jasper Reports, integration with, **21**

Java

evolution of, Spring Framework and, 31
when to use, 582–583

Java 5.0 annotations, **241–242**

Java Authentication and Authorization Service (JAAS)

advantages of, 373–375
definition of, 372–373
integration with Acegi Security, 373
when to use, 375

Java Data Objects (JDO)

DAO (Data Access Object) interface, 289–291
DAO support class for, 173
definition of, 255, 256, 288–289, 298–299
exception translations, 297
integration with, 19
isolation level, applying, 297
JDBC connection, access to, 297
object query language used by, 257
Open PersistenceManager in View pattern, 295–297
persistent object lifecycle, 289, 294–295
resource management for, 16
Spring setup for, 291–293
transaction management, 293–294
transaction timeout, applying, 297
transactions, flushing, 297
unchecked exceptions and, 16
when to use, 298–299

Java Message Service (JMS)

accessing with a template, 338–340
`ConnectionFactory` interface for, 340
definition of, 6, 335–337, 580
destination management for, 342–343
exception handling, 340
future support for, 345
Gateway pattern and, 343–345
message converters for, 340–342
message publication, 35
QOS (quality of service) parameters, 339
standard usage compared to Spring support, 337
synchronous receiving with, 339
transaction management, 343

Java Objects (POJOs)

AOP and, 11, 14, 578
application objects as, 7
business service objects as, 27, 585
persistence, 30, 302, 599
persistence specification, 599
pooling, 150
remoting and, 35
Spring Framework support for, 3, 4, 5, 37, 577
testing, 381, 420
transaction management and, 12, 34, 219, 225

Java Properties format, for container definitions, **108–109**

Java Standard Tag Library (JSTL), **489**, **494**, **497**

Java Transaction API (JTA)

definition of, 220
resource management for, 16, 17
simplified use of, 5
synchronization using, 278–280
transaction manager for, 249–251
when to use, 224–225

Java Transaction Processing: Design and Implementation (Little, Maron, Pavlik), **217**

Java Transaction Service (JTS) specification, **220–222**

JavaDoc, **597**

JavaMail API, **354**

JavaServer Faces (JSF) web framework

controllers, 531
data binding, 528
definition of, 525
features of, 528, 529, 531, 533
integration with, 20, 547–548
interceptors, 531
presentation layer, 529

JavaServer Pages (JSPs)

configuring application for, 497
constructing view using, 489
creating forms using custom tags, 497–505
definition of, 496–497

JAX-RPC, Web Services using

accessing a service, 326–328
custom application objects, 330–332
definition of, 324–325
exporting a service, 328–330
support classes for, 325
when to use, 325, 333

`JaxRpcPortProxyFactoryBean` **class**, **325**, **327**

`JaxRpcServicePostProcessor` **interface**, **330**

JBoss 3.x server, transaction manager for, 249**JbossIntegrationFilter module, Acegi, 383****JCA connector, Hibernate, 275–276****JDBC API**

advantages of, 175
 connection management, 174
 cross database escape syntax, 175
 DAO support class for, 172
 direct use of, 173–175
 example of, 67–69
 exception handling, 174
 resource lookup, 175
 resource management for, 16, 17
 SQL and, 175
 vendor implementations of, 175
 when to use, 215

JDBC framework

advantages of, 175–176
 batch updates, 208–209
 data access layer using, 177–182
 database querying with `JdbcTemplate` class,
 183–189, 204–206, 214
 database querying with RDBMS operation classes,
 189–194, 206–209
 definition of, 5, 171, 579
 example of, 176–177
 exception handling, 175–176, 180–182
 exception translations, 201–203
 key generation, 194–196
 performance of, 214
 running in application server, 199–201
 stored procedures, 196–199, 210–214
 versions of, 215
 when to use, 215, 584

JDBC RowSet Implementations tutorial, 189**JdbcDaoSupport class, 172****JdbcExtractor class, JDBC, 200, 208****JdbcTemplate class, JDBC**

batch updates, 209
 callback methods, 183–184
 convenience methods, 184–185
 example of, 176–177
 LOB access using, 204–206
 performance of, 214
 prepared statements using, 187–188
 queries using, 185–186
 updates using, 186

JDK proxy, determining whether proxy is, 145**JdkDynamicAopProxy interface, 158****JDO (Java Data Objects)**

DAO (Data Access Object) interface, 289–291
 DAO support class for, 173
 definition of, 255, 256, 288–289, 298–299
 exception translations, 297
 integration with, 19
 isolation level, applying, 297
 JDBC connection, access to, 297
 object query language used by, 257
 Open PersistenceManager in View pattern, 295–297
 persistent object lifecycle, 289, 294–295
 resource management for, 16
 Spring setup for, 291–293

transaction management, 293–294

transaction timeout, applying, 297

transactions, flushing, 297

unchecked exceptions and, 16

when to use, 298–299

JdoCallback class, 291**JdoDaoSupport class, 173****JdoDialect interface, 297–298****JdoInterceptor class, 294****JDOQL (JDO Query Language), 288****JdoTemplate class, 290–291****JdoTransactionManager class**

definition of, 225, 226, 227, 246–247

when to use, 293–294

J2EE

evolution of, Spring Framework and, 29–31

integration with Spring, 580

Spring Framework and, 38

traditional applications of, disadvantages of, 1–3

versions of, 215

J2EE Design and Development (Johnson), 428**J2EE Development without EJB (Johnson, Hoeller), 169,****398, 491, 597****jExcelApi, 516****jMock library, 106****JMS (Java Message Service)**

accessing with a template, 338–340

`ConnectionFactory` interface for, 340

definition of, 6, 335–337, 580

destination management for, 342–343

exception handling, 340

future support for, 345

Gateway pattern and, 343–345

message converters for, 340–342

message publication, 35

QOS (quality of service) parameters, 339

standard usage compared to Spring support, 337

synchronous receiving with, 339

transaction management, 343

JMSException class, JMS, 340**JmsException class, Spring, 340****JmsGatewaySupport class, 343–345****JmsTemplate class, 338–340****JmsTransactionManager class, 225, 226, 227, 343****JMX support, 6, 35, 580****JNDI API**

data source declaration for transactions, 253

resource management for, 16, 17

simplified use of, 5, 34–35

JndiDestinationResolver class, 343**JndiObjectFactoryBean class, 70, 342, 356,****400–402****JndiRmiProxyFactoryBean class, 323–324****JndiRmiServiceExporter class, 324****Job interface, 349****job, Quartz scheduler, 349–351****JobDataMap class, 351****JobDetail class, 351****JobDetailBean class, 351****Johnson, Ralph (Design Patterns), 597****Johnson, Rod**

AOP alliance founder, 120

coined Dependency Injection, 8

Johnson, Rod (continued)

Expert One-on-One J2EE Design and Development, 597
J2EE Design and Development, 428
J2EE Development without EJB, 169, 398, 491, 597
Spring Framework developer, 32–33

join point, AOP

definition of, 118
set of (pointcuts), 125–131

JOnAS/JOTM server, transaction manager for, 249

jPetStore sample application

definition of, 598
iBATIS SQL Maps and, 261
lightweight remoting and, 307
Tapestry and, 543
transactions and, 592

J2SE dynamic proxies, 12, 158

JSF web framework

controllers, 531
data binding, 528
definition of, 525
features of, 528, 529, 531, 533
integration with, 20, 547–548
interceptors, 531
presentation layer, 529

JSF-Spring, 595

JSPs (JavaServer Pages)

configuring application for, 497
constructing view using, 489
creating forms using custom tags, 497–505
definition of, 496–497

JSR-220 persistence, 30–31, 302

JSTL (Java Standard Tag Library), 489, 494, 497

JstlView class, 451, 497

JTA (Java Transaction API)

definition of, 220
resource management for, 16, 17
simplified use of, 5
synchronization using, 278–280
transaction manager for, 249–251
when to use, 224–225

JtaTransactionManager class

definition of, 226–227, 249–251
Hibernate and, 278
JMS and, 343
when to use, 224, 230, 266, 276, 277, 293, 294

JTS (Java Transaction Service) specification, 220–222

JUnit Recipes (Rainsberger), 110

JUnit tests

DAO and, 24
guidelines for, 593–594
integration testing and, 421–422
mock objects for, 36
security and, 381, 391–395
using, 103–108
web applications and, 485

jWebUnit tool, 485

K

KeyHolder interface, JDBC, 196

King, Gavin (*Hibernate in Action*), 269

Kopylenko, Dmitriy (Spring developer), 115

L

Laddad, Ramnivas (*AspectJ in Action*), 170, 597

large object (LOB), 203–208

layers

of architecture, 21–23, 583–588
business services layer
definition of, 23, 27, 583
guidelines for, 585–587
security for, 27, 368, 370–371
DAO (Data Access Object) interface layer, 23, 24–26, 583
data access layer
definition of, 583–585
for sample application, 177–182
database layer (EIS tier), 23, 583
presentation layer
definition of, 23, 27–29, 583
guidelines for, 587–588
JSF features for, 529
Spring MVC features for, 529
Struts features for, 528
Tapestry features for, 529
WebWork features for, 528
replaceability of, 7

lazy loading

Hibernate, 273
iBATIS SQL Maps, 266
JDO, 295

lazy-init attribute, bean element, 57

Lee, Bob (AOP Alliance founder), 120

lightweight container architecture, 21. See also architecture

lightweight frameworks, 3–4

lightweight remoting

accessors for, 306, 307–308
Burlap for
accessing a service, 310–312
definition of, 306, 309
exception handling, 308
exporting a service, 312–313
support classes for, 309–310
when to use, 310, 333
configuration for, 307–308
definition of, 5, 35, 305–307, 580
exporters for, 306, 308
Hessian for
accessing a service, 310–312
definition of, 306, 309
exception handling, 308
exporting a service, 312–313
support classes for, 309–310
when to use, 310, 333
HTTP invoker for
accessing a service, 314–315
additional invocation attributes for, 316

- customizing, 316–317
 - definition of, 306, 313
 - exception handling, 308
 - exporting a service, 315–316
 - secure communication with, 316
 - support classes for, 313–314
 - when to use, 333
 - RMI for
 - accessing a service, 318–320
 - customizing, 323
 - definition of, 306, 317
 - exception handling, 308
 - exporting a service, 322–323
 - RMI-IOP support, 323–324
 - stub lookups for, 317, 320–322
 - support classes for, 317–318
 - when to use, 318, 333
 - stateless remoting, 307
 - strategies for, choosing, 323–333
 - third-party remoting solutions supported, 306–307
 - Web Services with JAX-RPC for
 - accessing a service, 326–328
 - custom application objects, 330–332
 - definition of, 324–325
 - exporting a service, 328–330
 - support classes for, 325
 - when to use, 325, 333
 - limited privilege execution, 368**
 - list element, 59–60**
 - ListableBeanFactory interface, 48, 50**
 - listener, for servlet context, 93**
 - listeners property, AdvisedSupport class, 136**
 - lists, data binding and, 463**
 - Little, Mark (Java Transaction Processing: Design and Implementation), 217**
 - LOB (large object), 203–208**
 - LobHandler interface, 203–208, 285–286**
 - local EJBs, 402, 403–404**
 - local objects, when to use, 582**
 - local transactions, 221, 260**
 - LocalDataSourceConnectionFactory class, 300**
 - locale resolution, MVC, 454–455**
 - LocaleChangeInterceptor class, 449, 454–455**
 - LocaleEditor class, 102**
 - LocaleResolver class, 435, 441, 454–455**
 - LocalJaxRpcServiceFactoryBean class, 325**
 - LocalPersistenceManagerFactoryBean class, 291–293**
 - LocalSessionFactoryBean class, 71**
 - LocalSessionFactoryBean interface, 273–274, 276**
 - LocalStatelessSessionProxyFactoryBean class, 71**
 - LocalStatelessSessionProxyFactoryBean interface, 140, 403–404**
 - location paths for resources**
 - ant-style paths, 93
 - in application context constructors, 92–93
 - prefixes for, 85, 92
 - locking**
 - concurrency control and, 220
 - isolation level and, 219–220
 - LoginModule class, JAAS, 372, 373**
 - Lookup Method Injection, 44**
 - loosely typed operations, DAO, 260**
- ## M
- mail, electronic (email)**
 - attachments, 360
 - COS, sending mail using, 356
 - creating messages, 358, 359
 - example of, 354–355
 - existing mail session, reusing, 355–356
 - features for, 354
 - mail manager for, 356–360
 - recipients, handling, 358–359
 - sending messages, 360
 - MailManager interface, 356–357**
 - MailMessage class, COS, 356**
 - MailSender interface, 354**
 - Malks, Dan (Core J2EE Patterns), 172, 424**
 - MANDATORY transaction attribute, 222**
 - manual dependencies, 61**
 - map element, 59–60**
 - mapped statements, iBATIS SQL Maps, 257, 261–263**
 - mappers, 97**
 - mapping database tables to Java objects. See O/R (Object-Relational) mapping**
 - mapping file**
 - Hibernate, 269–270
 - iBATIS SQL Maps, 261–263
 - MappingSqlQuery class, JDBC, 189, 190–192, 207–208**
 - maps, data binding and, 463**
 - Maron, Jon (Java Transaction Processing: Design and Implementation), 217**
 - matches() method, MethodMatcher interface, 127**
 - Message Driven Beans, implementing, 411–412**
 - message publication using JMS, 35**
 - MessageConsumer class, 339**
 - MessageConverter interface, 341**
 - MessageCreator interface, 338–339**
 - MessageListener interface, 411–412**
 - Message-Oriented Middleware (MOM), 336**
 - messages. See also JMS (Java Message Service)**
 - application context as source of, 87–89
 - email
 - attachments, 360
 - COS, sending mail using, 356
 - creating messages, 358, 359
 - example of, 354–355
 - existing mail session, reusing, 355–356
 - features for, 354
 - mail manager for, 356–360
 - recipients, handling, 358–359
 - sending messages, 360
 - MessageSource interface, 49, 50, 87–89**
 - MessageSourceAccessor class, 89**
 - MessageSourceAware interface, 65, 88**
 - MessageSourceResolvable interface, 87**
 - messaging. See JMS (Java Message Service)**
 - Method Injection**
 - definition of, 9, 589
 - example of, 44–45

Method Injection (continued)

- testing and, 45
- when to use, 44, 590
- method interceptors, 120–121**
- `MethodBeforeAdvice` **interface, 122–123**
- `MethodInterceptor` **interface, 120–121, 153, 161**
- `MethodInvocation` **interface, 157, 161**
- `MethodInvokingFactoryBean` **class, 71, 101–102**
- `MethodInvokingTimerTaskFactoryBean` **class, 347**
- `MethodMatcher` **interface, 126, 127**
- `MethodSecurityInterceptor` **class, Acegi, 385**
- migration**
 - to future versions of Spring Framework, 6
 - to other lightweight containers, 6
- mixin, 153, 154–156**
- mock objects, for unit testing, 104–106, 481–483**
- MockEJB framework, 420**
- model objects. See also ModelAndView class**
 - definition of, 28, 426
 - provided by Spring, 429–430
 - rendered by views, 491–492
- Model View Controller. See MVC web frameworks**
- `ModelAndView` **class, 429–430, 450**
- modules. See also specific modules**
 - handling, 97–99
 - list of, 5–6, 33–36
- MOM (Message-Oriented Middleware), 336**
- Mule framework, 596**
- `MultiActionController` **class, 460–461**
- multipart resolver, 479–481**
- `MultipartHttpServletRequest` **class, 479, 480**
- `MultipartResolver` **class, 435, 441**
- `MutablePropertyValues` **class, 110**
- MVC web frameworks. See also controller objects; model objects; view objects**
 - choosing, 523–524, 587–588
 - comparing available frameworks
 - controllers and interception handling, 530–532
 - data binding, 525–528
 - presentation layer, 528–529
 - requirements for, 524–525, 532–533
 - integration with Spring, 534–536
 - JSF web framework
 - controllers, 531
 - data binding, 528
 - definition of, 525
 - features of, 528, 529, 531, 533
 - integration with, 20, 547–548
 - interceptors, 531
 - presentation layer, 529
 - list of, 20
 - requirements for, 524
 - Spring MVC web framework
 - application design using, 427, 430–431, 440–441
 - architecture of, 425–430
 - components of, 430–436, 440–441
 - controllers, 530–531
 - data binding, 461–463, 526–527
 - definition of, 5, 35, 580
 - example of, 423–425, 464–474
 - exception handling, 438, 456–458
 - features of, 526–527, 529, 530–531, 532–533

- handler adapters, 449
- handler exception resolvers, 456–458
- handler interceptors, 446–449
- handler mappings, 441–446
- handlers, extending, 478
- handling requests, 436–439
- Hibernate, Open Session in View, 282–285
- interceptors, 530–531
- locale resolution, 454–455
- presentation layer, 529
- pull model and, 426
- push model and, 427
- requirements for, 428
- testing controllers, 481–485
- testing web applications, 485
- uploading files, 478–481
- view resolvers, 450–454
- Struts web framework
 - compared to Spring Framework, 1, 3
 - controllers, 530
 - converting to Spring MVC, 437
 - data binding, 526
 - definition of, 525
 - features of, 526, 528, 530, 532
 - integration with, 20, 537–543
 - interceptors, 530
 - presentation layer, 528
- Tapestry web framework
 - controllers, 531
 - data binding, 527
 - definition of, 525
 - features of, 527, 529, 531, 533
 - integration with, 20, 543–547
 - interceptors, 531
 - presentation layer, 529
- WebWork web framework
 - controllers, 530
 - definition of, 596
 - features of, 530, 532
 - integration with, 536–537
 - interceptors, 530

MySQL database server, 552

N

- name **attribute, bean element, 54**
- named services, application objects as, 7**
- `NameMatchMethodPointcut` **class, 128**
- namespace **parameter, DispatcherServlet class, 432**
- `NativeJdbcExtractorAdapter` **class, JDBC, 200**
- .NET, The Spring Framework.NET, 595**
- NEVER transaction attribute, 222**
- NONE isolation level, 219**
- non-invasive framework, 6**
- non-repeatable reads, 229**
- non-singleton (prototype) beans**
 - compared to singleton beans, 55–56
 - destruction process for, 67
 - in lifecycle of AOP proxies, 140–141
- NOT SUPPORTED transaction attribute, 222**

O**Oberg, Rickard (typed advice), 125****Object Graph Navigation Language (OGNL), 463****Object Oriented (OO) design**

books about, 597

crosscutting concerns not addressed by, 114

guidelines for, 589

Spring Framework support for, 7

object query languages, 257**Object relational Bridge (Apache OJB)**

DAO support class for, 173

definition of, 255

integration with, 19

object query language used by, 257

O/R mapping and, 299–300

transaction manager for, 247–248

Object Transaction Service (OTS), 221**Object-Relational (O/R) mapping**

Apache OJB and, 299–300

automatic change detection, 258

books about, 597

Cayenne and, 302

DAO design and implementation for, 259–260

definition of, 256–257, 302–303

Hibernate and

BLOB/CLOB handling, 285–287

DAO implementation for, 270–273

features for, 268–269, 287–288

mapping file, 269–270

Open Session in View pattern, 282–285

Spring setup for, 273–276

transaction management, 276–282

iBATIS SQL Maps and, 261–268

JDO and

DAO implementation for, 289–291

features for, 288–289, 298–299

JDO dialect, 297–298

Open PersistenceManager in View pattern, 295–297

PersistenceManager lifecycle, 294–295

persistent object lifecycle, 289

Spring setup for, 291–293

transaction management, 293–294

JSR-220 persistence and, 302

JTA transactions and, 251

object query languages used in, 257

third-party tools, Spring integrating with, 255–256

TopLink and, 300–302

transaction management for, 260

transparent persistence, 258

when to use, 258–259, 583–585

objects. See also DAO (Data Access Object) interface;**POJOs (Plain Old Java Objects)**

BLOB (Binary Large Object), 203–206, 285–287

CLOB (Character Large Object), 203–204, 285–287

dirty checking for, 113–114

distributed compared to local, 582

dynamic, 13

LOB (large object), 203–208

providing services to, with AOP, 11–12

target object, AOP, 118–119

Observer design pattern, 12**OGNL (Object Graph Navigation Language), 463****OJB (Apache)**

DAO support class for, 173

definition of, 255

integration with, 19

object query language used by, 257

O/R mapping and, 299–300

transaction manager for, 247–248

OjbFactoryUtils class, 248**onBind() method, SimpleFormController****class, 469****onBindAndValidate() method, AbstractWizard****FormController class, 475****onBindAndValidate() method, SimpleForm****Controller class, 469****ONJava articles (Bergsten), 496****onSubmit() method, SimpleFormController****class, 469–470****OO (Object Oriented) design**

books about, 597

crosscutting concerns not addressed by, 114

guidelines for, 589

Spring Framework support for, 7

opaque property, ProxyConfig class, 135**Open PersistenceManager in View pattern, JDO API,****295–297****Open Session in View pattern, 273, 282–285****open source, Spring Framework and, 32****Open Symphony WebWork. See WebWork web framework****OpenPersistenceManagerInViewFilter class,****295–297****OpenPersistenceManagerInViewInterceptor****class, 295–297, 449****OpenSessionInViewFilter class, 283–285****OpenSessionInViewInterceptor class, 283–285,****449****optimistic locking, 220****OptimisticLockingFailureException class, 259****optimize property, ProxyConfig class, 134****O/R (Object-Relational) mapping**

Apache OJB and, 299–300

automatic change detection, 258

books about, 597

Cayenne and, 302

DAO design and implementation for, 259–260

definition of, 256–257, 302–303

Hibernate and

BLOB/CLOB handling, 285–287

DAO implementation for, 270–273

features for, 268–269, 287–288

mapping file, 269–270

Open Session in View pattern, 282–285

Spring setup for, 273–276

transaction management, 276–282

iBATIS SQL Maps and, 261–268

JDO and

DAO implementation for, 289–291

features for, 288–289, 298–299

JDO dialect, 297–298

Open PersistenceManager in View pattern, 295–297

O/R (Object-Relational) mapping (continued)

- PersistenceManager lifecycle, 294–295
- persistent object lifecycle, 289
- Spring setup for, 291–293
- transaction management, 293–294
- JSR-220 persistence and, 302
- JTA transactions and, 251
- object query languages used in, 257
- third-party tools, Spring integrating with, 255–256
- TopLink and, 300–302
- transaction management for, 260
- transparent persistence, 258
- when to use, 258–259, 583–585

Oracle, key generation using, 195

Oracle OC4J server, transaction manager for, 249

Oracle TopLink, 255

OracleSequenceMaxValueIncrementer class,
JDBC, 195

Order property, AbstractAutoProxyCreator class,
143

O'Reilly Servlet (COS) package, 478–479

Orion server, transaction manager for, 249

orthogonal (crosscutting) concerns, 113–114

OTS (Object Transaction Service), 221

P

P2P (Point-to-Point) messaging, 336

Page class, **544**

pageAttribute property, AbstractWizardForm
Controller class, **475, 476**

pages property, AbstractWizardFormController
class, **475**

ParameterizableViewController class, **460**

parent bean definitions, 72–73, 74

**Patterns of Enterprise Application Architecture (Fowler),
343**

**Pavlik, Greg (Java Transaction Processing: Design and
Implementation), 217**

PDFs

- generating with iText library, 490–491
- view classes for, 517

persist (save) methods, DAO, 24

persistence frameworks, integration with, 19–20

persistence services, 23–26

**persistence solutions. See O/R (Object-Relational)
mapping**

PersistenceBroker API, Apache OJB, 300

PersistenceBrokerDaoSupport class, **173, 300**

PersistenceBrokerTemplate class, **300**

PersistenceBrokerTransactionManager class

- definition of, 226, 227, 247–248

- when to use, 225, 300

PersistenceManager class, **289, 294, 295**

PersistenceManagerFactory class, **290–293, 294**

PersistenceManagerFactoryUtils class, **246**

persistent domain objects, 23, 26, 583, 585

persistent object lifecycle, JDO API, 289, 294–295

pessimistic locking, 220

PetClinic sample application, 598

phantom reads, 219, 229

Plain Old Java Objects (POJOs)

- AOP and, 11, 14, 578
- application objects as, 7
- business service objects as, 27, 585
- persistence, 30, 302, 599
- persistence specification, 599
- pooling, 150
- remoting and, 35
- Spring Framework support for, 3, 4, 5, 37, 577
- testing, 381, 420
- transaction management and, 12, 34, 219, 225

PlatformTransactionManager interface, **225, 227,
230, 278**

pluggability

- Acegi Security support for, 369

- Spring Framework support for, 7

pluggable cache strategies, Hibernate, 269

plumbing code, 2

Pointcut interface, **126–127**

PointcutAdvisor class, **130**

pointcuts, AOP

- control flow pointcuts, 129

- definition of, 12, 114, 125–126

- dynamic, 127, 129–130

- implementing, 126–127

- operating on, 130

- provided by Spring, 127–130

- regular expression pointcuts, 128–129, 131

- static, 127, 129

Pointcuts class, **127, 130**

Point-to-Point (P2P) messaging, 336

POJO-based remoting. See lightweight remoting

POJOs (Plain Old Java Objects)

- AOP and, 11, 14, 578

- application objects as, 7

- business service objects as, 27, 585

- persistence, 30, 302, 599

- persistence specification, 599

- pooling, 150

- remoting and, 35

- Spring Framework support for, 3, 4, 5, 37, 577

- testing, 381, 420

- transaction management and, 12, 34, 219, 225

pooling target objects, 149–152

PoolingConfig interface, **152**

prefixes for resource location strings, 85, 92

prepared statements, JDBC, 187–188

PreparedStatement interface, **JDBC, 174**

PreparedStatementCreator interface, **JDBC, 187**

PreparedStatementSetter interface, **JDBC, 188**

presentation layer

- definition of, 23, 27–29, 583

- guidelines for, 587–588

- JSF features for, 529

- Spring MVC features for, 529

- Struts features for, 528

- Tapestry features for, 529

- WebWork features for, 528

primary keys, generating, 194–196

principal, 367

Principal class, **JAAS, 372**

Pro JSP Third Edition (Brown et al.), 496

`proceed()` **method**
 MethodInterceptor interface, 121
 MethodInvocation interface, 159–160, 161, 163
`processCancel()` **method**, AbstractWizardForm
 Controller **class**, 475
`processFinish()` **method**, AbstractWizardForm
 Controller **class**, 475
programmatically transaction demarcation, 227, 230–232
programmatically transaction management, 221
programming model, consistent, 6
propagation behavior of transactions, 227–228
PROPAGATION_MANDATORY option for transactions, 228
PROPAGATION_NESTED option for transactions, 228
PROPAGATION_NEVER option for transactions, 228
PROPAGATION_NOT_SUPPORTED option for transactions, 228
PROPAGATION_REQUIRED option for transactions, 228
PROPAGATION_REQUIRES_NEW option for transactions, 228
PROPAGATION_SUPPORTS option for transactions, 228
Properties format, for container definitions, 108–109
 PropertiesBeanDefinitionReader **class**, 49
 PropertiesEditor **class**, 102
property editors
 custom, creating and using, 79–80
 provided by Spring, list of, 102–103
 type conversion using, 57
 property **element**, 58–61
 PropertyEditor **class**, 57, 79–80, 102–103, 462
 PropertyOverrideConfigurer **interface**, 78
 PropertyPathFactoryBean **class**, 99–100
 PropertyPlaceholderConfigurer **interface**,
 77–78, 274
props element, 59–60
prototype (non-singleton) beans
 compared to singleton beans, 55–56
 destruction process for, 67
 in lifecycle of AOP proxies, 140–141
 ProviderManager **class**, Acegi, 377, 378
proxy-based AOP framework, 12, 34, 114, 118
 ProxyConfig **class**, 132–135
 ProxyFactory **class**, 132–133, 147
 ProxyFactoryBean **interface**
 creating AOP proxies using, 137–140, 170
 definition of, 70, 132–133
 transaction interceptors and, 233–235
 when to use, 145
 proxyInterfaces **property**, ProxyFactoryBean
class, 138
 proxyTargetClass **property**, ProxyConfig **class**, 134
publications
AspectJ in Action (Laddad), 170, 597
Core J2EE Patterns (Alur, Crupi, Malks), 172, 424
Design Patterns (Gamma, Helm, Johnson, Vlissides), 597
Domain Driven Design (Evans), 597
*Eclipse AspectJ: Aspect-Oriented Programming with
 AspectJ and the Eclipse AspectJ Development Tools*
 (Colyer), 170, 597
Expert One-on-One J2EE Design and Development
 (Johnson), 32, 427, 458, 597
Hibernate in Action (King, Bauer), 269

iBATIS SQL Maps Developer Guide (Begin), 261
J2EE Design and Development (Johnson), 428
J2EE Development without EJB (Johnson, Hoeller), 169,
 398, 491, 597
Java Transaction Processing: Design and Implementation
 (Little, Maron, Pavlik), 217
 JDBC RowSet Implementations tutorial, 189
 JDBC tutorial, 171
JUnit Recipes (Rainsberger), 110
 list of, 597
 ONJava articles (Bergsten), 496
Patterns of Enterprise Application Architecture (Fowler),
 343
Pro JSP Third Edition (Brown et al.), 496
 “Spring and AspectWerkz—A Happy Marriage” (Boner),
 169
 publishContext **parameter**, DispatcherServlet
class, 432
 publishEvent() **method**, ApplicationEvent
 Publisher **interface**, 90
Pub/Sub (Publish/Subscribe) messaging, 336
pull configuration, 9
pull model, in MVC, 426
push configuration, 9. *See also* Dependency Injection
push model, in MVC, 427
 Push2Test TestMaker tool, 485

Q

QoS (quality of service) parameters, JMS, 339
Quartz scheduler
 definition of, 345
 features for, 346
 integration with, 21
 job for, 349–351
 scheduler for, 352–354
 trigger for, 348–349, 351–352
 QuartzJobBean **class**, 349–350
query statements, in iBATIS SQL mapping file, 263

R

radio buttons, data binding and, 463
Rainsberger, J. B. (*JUnit Recipes*), 110
RCP (Rich Client Project), 29, 596
RDBMS
 batch updates, 209
 definition of, 189
 LOB support, 206–208
 queries using, 190–192
 updates using, 192–194
 RDBMSOperation **class**, JDBC, 189
READ COMMITTED isolation level, 219, 220
READ UNCOMMITTED isolation level, 219, 220
read-only transactions, 229
 RedirectView **class**, 494–496
 ref **element**, 58
 referenceData() **method**
 AbstractWizardFormController **class**, 475
 SimpleFormController **class**, 469
 ReflectiveMethodInvocation **interface**, 163
regular expression pointcuts, 128–129, 131

release schedule, 29

ReloadableResourceBundleMessageSource
interface, 89

remote EJBs, 402–403, 404–406

RemoteAccessException class, 307–308

RemoteAuthenticationManager interface, Acegi, 380

RemoteInvocation class, 323

RemoteInvocationExecutor class, 316, 323

RemoteInvocationFactory class, 316, 323

RemoteInvocationResult class, 323

remoting. See also lightweight remoting

client-server remoting, 305

features for, 28

guidelines for, 588

intra-server remoting, 305

removeAdvice property, AdvisedSupport class, 136

removeAdvisor property, AdvisedSupport class, 136

render() method

AbstractView class, 520

View interface, 488, 492

REPEATABLE READ isolation level, 219

requestContextAttribute property,

AbstractView class, 520

RequestHandledEvent event, 90

RequestUtils class, 465, 518

REQUIRED transaction attribute, 222

REQUIRES NEW transaction attribute, 222

requireSession property, WebContentInterceptor
class, 448

Resin server, transaction manager for, 249

resolvers, MVC, 435, 438, 440–441

Resource interface, 49, 50, 84–89

resource management

ant-style paths for location strings, 93

features for, 16–18

JDBC API, 175

location paths for resources, in application context

constructors, 92–93

low-level resources, accessing, 84–89

prefixes for location strings, 85, 92

transactions and, 220

ResourceArrayPropertyEditor class, 102

ResourceBundleMessageSource interface, 88

ResourceBundleViewResolver class, 451–452, 493

ResourceEditor class, 57, 102

ResourceLoader interface, 49, 50, 84–87

ResourceLoaderAware interface, 65, 85–86

resources. See publications; websites

result maps, in iBATIS SQL mapping file, 263

ResultSet interface, JDBC, 174, 187, 193–194,
211–214

retrying (transparent recovery), 13

Rich Client Project (RCP), 29, 596

RMI, remoting using

accessing a service, 318–320

customizing, 323

definition of, 306, 317

exception handling, 308

exporting a service, 322–323

RMI-IIOP support, 323–324

stub lookups for, 317, 320–322

support classes for, 317–318

when to use, 318, 333

RMI-IIOP, remoting using, 323–324

RmiProxyFactoryBean class, 70, 317–318, 320–321

RmiServiceExporter class, 318, 323

RoleVoter class, Acegi, 383–384

rollback rules, 234–235

RootBeanDefinition class, 110

RowCallbackHandler interface, JDBC, 187

RowMapper interface, 256

RowSet interface, JDBC, 188–189

RuntimeException class, 48

S

sample application (ImageDB), 598

sample application (jPetStore)

definition of, 598

iBATIS SQL Maps and, 261

lightweight remoting and, 307

Tapestry and, 543

transactions and, 592

sample application (PetClinic), 598

sample application (ticket reservation system)

application context, 565–566

application layers, 552–553

application server, 552, 574, 575

build and deployment, 574–575

business services layer, 563–566

compared to sample application in *J2EE Design and
Development*, 572–573

configuration with web.xml file, 568–569

controllers, 569–570

data model for, 177–178

database, creating and loading, 574–575

database server, 573–574

DataSource for, 178–180

downloading source code for, 551

error handling, 606

persistence layer

DAO implementation, 561–562

data access context, 563

data model, 554–555

domain object model, 556–557

O/R mapping, 558–561

requirements for

assumptions, 603–604

delivery schedule, 604

hardware and software, 619–620

overview, 601–602

performance, 618–619

scope limitations, 604

user populations, 602–603

use cases

making a reservation, 466–474

viewing list of performances, 464–466

user interface

application screens, 606–608

book seats screen, 611–612

box office interface, 618

confirm reservation screen, 616–617

- display show screen, 609–611
- payment details screen, 614–616
- show reservation screen, 612–614
- welcome screen, 608–609
- workflow for, 605–606
- view technology, 571–572
- web layer, 566–572
- save (persist) methods, DAO, 24**
- SAXHandlers **class, 519**
- SchedulerFactoryBean **class, 349, 352–353**
- scheduling**
 - definition of, 345
 - with Quartz, 345, 346, 348–354
 - with Timers, 345, 346–348
- Script **interface, 365**
- scripting**
 - Beanshell scripts, 363–365, 582–583
 - Groovy scripts, 362–363, 582–583
 - support for, 31, 35, 361–365, 580
- security. See also Acegi Security**
 - in business services layer, 27, 368, 370–371
 - for domain objects, 368
 - JAAS (Java Authentication and Authorization Service), 372–375
 - limited privilege execution, 368
 - requirements for, 367–368
 - Servlet Specification, 375–377
 - for web requests, 368
- SecurityEnforcementFilter **class, 369**
- SecurityInterceptionEvent **class, 372**
- SecurityManager **class, JAAS, 373**
- serializability, 588**
- serializable AOP proxies, 162–163**
- SERIALIZABLE isolation level, 219**
- server**
 - running JDBC framework in, 199–201
 - testing applications deployed in, 108
 - transaction management and, 200–201, 220, 249–251
 - when to use, 581
- server-config.wssdd **file, 330**
- service abstraction, 14–18**
- services**
 - abstraction for, 14–18, 67–71
 - named, application objects as, 7
 - providing to objects with AOP, 11–12
 - Web Services with JAX-RPC
 - accessing a service, 326–328
 - custom application objects, 330–332
 - definition of, 324–325
 - exporting a service, 328–330
 - support classes for, 325
 - when to use, 325, 333
- services layer. See business services layer**
- servlet context listener, 93**
- servlet, load-on-startup, 93**
- Servlet Specification**
 - Acegi Security and, 380–381
 - definition of, 375–377
- ServletContext **class, 534–536**
- ServletContextResource **class, 85**
- ServletEndpointSupport **class, 325, 328**
- Session API, Hibernate, 270, 280**
- SessionFactory **class, Hibernate, 270, 272, 275**
- SessionFactoryBean **class, 301**
- SessionFactoryUtils **class, 245, 272**
- sessionForm **property, SimpleFormController class, 468**
- SessionLocaleResolver **class, 455**
- set **element, 59–60**
- setApplicationContext() **method, Application ContextAware interface, 65, 67**
- setApplicationEventPublisher() **method, ApplicationEventPublisherAware interface, 65**
- setBeanFactory() **method, BeanFactoryAware interface, 65, 67**
- setBeanName() **method, BeanNameAware interface, 65**
- setMessageSource() **method, MessageSource Aware interface, 65**
- setResourceLoader() **method, ResourceLoader Aware interface, 65**
- Setter Injection, 9**
 - definition of, 589
 - example of, 10, 41–43
 - specifying dependencies and, 56
 - when to use, 45–47, 590
- SETTERS **constant, Pointcuts class, 127**
- SimpleFormController **class, 467–470**
- SimpleMailMessage **class, 354**
- SimpleMappingExceptionResolver **class, 458**
- SimpleMessageConverter **interface, 341**
- SimpleNativeJdbcExtractor **class, JDBC, 200**
- SimpleRemoteStatelessSessionProxyFactoryBean **class, 71, 404–405**
- SimpleService **interface, 408–410**
- SimpleTriggerBean **class, 348–349, 351–352**
- SimpleUrlHandlerMapping **class, 443–444**
- single session mode, Hibernate Open Session in View, 283**
- SingleConnectionDataSource **interface, JDBC, 179**
- single-resource transaction managers, 243–248**
- singleton aspects, AspectJ, 165–166**
- singleton attribute, bean element, 55–56**
- singleton beans**
 - accessing container using, 99
 - compared to non-singleton beans, 55–56
 - in lifecycle of AOP proxies, 140
- singleton container access for EJBs, 413–420**
- singleton property, AdvisedSupport class, 137**
- SingletonBeanFactoryLocator **class, 99, 414–416**
- SingletonTargetSource **interface, 144–145**
- SiteMesh, 513**
- snapshot comparisons, Hibernate, 268**
- SOAP, remoting using, 307**
- SourceForge c3PO, 252**
- source-level metadata for transactions, 238–242**
- “Spring and AspectWerkz – A Happy Marriage” (Boner), 169**
- Spring Framework**
 - advantages of, 3, 36, 577
 - architecture of, 21–29
 - definition of, 1
 - environments supported by, 37

Spring Framework (continued)

- future of, 29–32, 598–599
- goals of, 6–8
- guidelines for development using
 - application structure, 589–592
 - layering best practices, 583–588
 - technology choices, 581–583
 - testing, 592–594
- history of, 32–33
- integration with other frameworks, 19–21
- JARs, list of, 36
- modules in, list of, 5–6, 33–36, 578–580
- related projects for, 594–596
- release schedule of, 29
- size of, 36
- techniques used by, 18
- technologies used by, 8–18

Spring Framework community, 33

The Spring Framework.NET, 595

Spring IDE for Eclipse, 595

The Spring Modules project, 595

Spring MVC web framework. See also controller objects; model objects; view objects

- choosing, 523–524, 587–588
- comparing available frameworks
 - controllers and interception handling, 530–532
 - data binding, 525–528
 - presentation layer, 528–529
 - requirements for, 524–525, 532–533
- integration with Spring, 534–536

JSF web framework

- controllers, 531
- data binding, 528
- definition of, 525
- features of, 528, 529, 531, 533
- integration with, 20, 547–548
- interceptors, 531
- presentation layer, 529

list of, 20

requirements for, 524

Spring MVC web framework

- application design using, 427, 430–431, 440–441
- architecture of, 425–430
- components of, 430–436, 440–441
- controllers, 530–531
- data binding, 461–463, 526–527
- definition of, 5, 35, 580
- example of, 423–425, 464–474
- exception handling, 438, 456–458
- features of, 526–527, 529, 530–531, 532–533
- handler adapters, 449
- handler exception resolvers, 456–458
- handler interceptors, 446–449
- handler mappings, 441–446
- handlers, extending, 478
- handling requests, 436–439
- Hibernate, Open Session in View, 282–285
- interceptors, 530–531
- locale resolution, 454–455
- presentation layer, 529
- pull model and, 426
- push model and, 427

- requirements for, 428
- testing controllers, 481–485
- testing web applications, 485
- uploading files, 478–481
- view resolvers, 450–454

Struts web framework

- compared to Spring Framework, 1, 3
- controllers, 530
- converting to Spring MVC, 437
- data binding, 526
- definition of, 525
- features of, 526, 528, 530, 532
- integration with, 20, 537–543
- interceptors, 530
- presentation layer, 528

Tapestry web framework

- controllers, 531
- data binding, 527
- definition of, 525
- features of, 527, 529, 531, 533
- integration with, 20, 543–547
- interceptors, 531
- presentation layer, 529

WebWork web framework

- controllers, 530
- definition of, 596
- features of, 530, 532
- integration with, 536–537
- interceptors, 530

Spring RCP (Rich Client Project), 29, 596

Spring Rich Client (RCP) project, 29, 596

Spring XMLDB, 595

spring-beans.dtd file, 53

spring:bind tag, 498, 499–501

spring:hasBindErrors tag, 498

spring:htmlEscape tag, 498

spring:message tag, 498, 502–503

spring-mock.jar file, 481

spring:nestedPath tag, 501

SpringObjectFactory class, 536–537

spring:transform tag, 498, 501–502

SQL Maps (iBATIS)

- configuration file, 265–266

- DAO implementation for, 263–265

- DAO support class for, 173

- definition of, 255, 257, 261, 267–268

- integration with, 19

- mapped statements (mapping file) for, 261–263

- Spring setup for, 265–266

- transaction management using, 266–267

- when to use, 267, 584

SQL queries, JDBC

- features for, 175

- with `JdbcTemplate` class, 183–189, 204–206

- with RDBMS operation classes, 189–194, 206–209

SQL state code, JDBC, 174

SQLExceptionCodeSQLExceptionTranslator class, JDBC, 201, 202–203

SQLException class, JDBC, 174, 180–182

SqlMapClient API, iBATIS SQL Maps, 263–265

SqlMapClientDaoSupport class, 173, 263

SqlMapClientFactoryBean class, 265

- SqlMapClientTemplate **class**, 263–265
 sql-map-config.xml **file**, 265–266
 SqlQuery **class**, JDBC, 189, 190–192
 SqlUpdate **class**, JDBC, 189, 192–193, 206–208
standards, Spring Framework and, 31–32
Stateful Session Beans, implementing, 410–411
 StatefulJob **interface**, 350–351
 StatefulService **interface**, 410
stateless remoting, 307
Stateless Session Beans
 implementing, 407–410
 lookups, encapsulating, 402–406
 Statement **interface**, JDBC, 174
statement wrappers, with application servers, 200–201
static pointcuts, 127, 129
 staticAttributesMap **property**, AbstractView **class**, 520
 StaticMethodMatcherPointcut **class**, 129
status of transaction, 229
 status.displayValue() **method**, BindStatus **class**, 500
status.error() **method**, BindStatus **class**, 500
 status.errorCode() **method**, BindStatus **class**, 500
status.errorCodes() **method**, BindStatus **class**, 500
 status.errorMessage() **method**, BindStatus **class**, 500
 status.errorMessages() **method**, BindStatus **class**, 500–501
 status.expression() **method**, BindStatus **class**, 500
 status.value() **method**, BindStatus **class**, 500
stored procedures, JDBC, 196–199, 210–214
 StoredProcedure **class**, JDBC, 189, 196–199, 211–214
 StringArrayPropertyEditor **class**, 103
 StringTrimmedEditor **class**, 103
strongly typed operations, DAO, 260
Struts Tiles, 511–513
Struts web framework
 compared to Spring Framework, 1, 3
 controllers, 530
 converting to Spring MVC, 437
 data binding, 526
 definition of, 525
 features of, 526, 528, 530, 532
 integration with, 20, 537–543
 interceptors, 530
 presentation layer, 528
 struts-config.xml **file**, 539, 541
stub lookups, RMI remoting, 317, 320–322
 Subject **class**, JAAS, 372
 successView **property**, SimpleFormController **class**, 468
 supportedMethods **property**, WebContentInterceptor **class**, 448
 supports() **method**, HandlerAdapter **interface**, 478
SUPPORTS transaction attribute, 222
 suppressValidation() **method**, SimpleFormController **class**, 470
 swap() **method**, HotSwappableTargetSource **class**, 148
Swing-based applications, using Spring with, 596
synchronous messaging, 339
- T**
tag files, 504–505
tags, Spring, 497–504
Tapestry web framework
 controllers, 531
 data binding, 527
 definition of, 525
 features of, 527, 529, 531, 533
 integration with, 20, 543–547
 interceptors, 531
 presentation layer, 529
target object, AOP
 custom target sources, 152
 definition of, 118–119, 147–148
 hot swapping, 13, 148–149
 pooling, 149–152
 programmatically access to, 152
 when not to use, 152–153
 target **property**, AdvisedSupport **class**, 135, 138
 TargetSource **interface**
 for autoproxy creation, 144–145
 custom, 152
 implementations of, 147–153
 programmatically access to, 152
 targetSource **property**, AdvisedSupport **class**, 135
 TargetSourceCreator **interface**, 144–145
TDD (Test Driven Development), 18
template (abstract parent bean definition), 73
templates for resource management, 17
Test Driven Development (TDD), 18
testing
 AOP used for, 159–161
 container used in, 106–108
 ease of, Spring Framework support for, 7
 EJBs, 420–422
 guidelines for, 592–594
 integration testing
 AOP framework used for, 160
 in application server, 421–422
 container used for, 106–108
 guidelines for, 593–594
 support for, 18
 tools for, 485
 splitting container definitions into multiple files for, 95–96
 unit testing
 difficulty of, 2
 example of, 103–106
 features for, 6, 18
 guidelines for, 160, 161, 593
 mock objects used for, 104–106
TestMaker tool (Push2Test), 485
 ThemeResolver **class**, 435
 ThemeSource **interface**, 50
thread safety
 advice and, 123
 AOP framework and, 146

thread safety (continued)

- guidelines for, 586–587
- hot swapping and, 148
- of `JdbcTemplate` class, 183
- of RDBMS operation classes, 189
- of `SingleConnectionDataSource` interface, 179
- of `StoredProcedure` class, 199
- `ThrowAwayController` class, 449, 465
- throws advice, 124–125**
- `ThrowsAdvice` interface, 122, 124–125
- tiles, 511–513**
- `TilesConfigurer` class, 513
- timeout for transaction, 229**
- `Timer` class, 345, 346–348
- `TimerFactoryBean` class, 348
- Timers, scheduling using, 345, 346–348**
- Tirsen, Jon (AOP Alliance founder), 120**
- Tomcat server (Apache), 552, 574, 575**
- TopLink API**
 - integration with, 19
 - object query language used by, 257
 - O/R mapping and, 300–302
 - resource management for, 16
- `TopLinkDaoSupport` class, 301
- `TopLinkTemplate` class, 301
- `TopLinkTransactionManager` class, 247, 301
- `toProxyConfigString()` method, `Advised` interface, 160
- TP monitor, 220**
- transaction attributes, EJB, 222**
- transaction demarcation**
 - declarative
 - `BeanNameAutoProxyCreator` interface, 237–238
 - definition of, 225, 227, 233
 - guidelines for, 242
 - performance of, 242
 - `ProxyFactoryBean` interface and transaction interceptor, 233–235
 - source-level metadata, 238–242
 - `TransactionProxyFactoryBean` interface, 235–236
 - guidelines for, 242
 - performance of, 242
 - programmatically, 225, 227, 230–232
 - strategies for, 221–222, 225, 229–230
 - where to demarcate, 230
- transaction interceptors**
 - definition of, 233–235
 - Hibernate, 281–282
 - JDO, 294–295
- transaction management**
 - ACID requirements of, 218–220
 - with application servers, 200–201, 220, 249–251
 - atomicity, 218–219
 - in business services layer, 27
 - CMT (container-managed transactions), 219, 224
 - concurrency control, 220
 - configuration for, 226–229
 - as crosscutting concern, 113–114
 - data source declaration, 251–254
 - declarative transaction management, 13, 219, 222, 224

- definition of, 5, 34, 217–218, 579
- example of, 222–224
- global (distributed) transactions, 221, 260
- guidelines for, 585–586
- isolation, 219–220, 228–229
- for JMS (Java Message Service), 343
- JTA (Java Transaction API), 224–225
- JTA synchronization, 278–280
- JTS specification regarding, 220–222
- local transactions, 221
- for O/R mapping, 260
- for O/R mapping with Hibernate, 276–282
- for O/R mapping with iBATIS SQL Maps, 266–267
- for O/R mapping with JDO, 293–294
- programmatically, 221
- propagation behavior of transactions, 227–228
- read-only transactions, 229
- rollback rules, 234–235
- single-resource transaction managers, 243–248
- source-level metadata for, 238–242
- Spring features for, list of, 224–226
- status of transaction, 229
- strategies for
 - list of, 225–226
 - multiple resource, 249–251
 - single resource, 243–248
- timeout for transaction, 229
- transaction context, 221
- transaction propagation, 221
- where to apply transactions, 230

transaction manager, 220

- `TransactionAttribute` interface, 226
- `TransactionAwareDataSourceProxy` class, 244
- `TransactionDefinition` interface, 226
- `TransactionManager` interface, 250–251
- `TransactionProxyFactoryBean` class, 70, 72
- `TransactionProxyFactoryBean` interface, 140, 235–236, 266, 276
- `TransactionStatus` interface, 229
- `TransactionTemplate` class, 225, 232, 266, 276

Transfer Objects

- disadvantages of, 2
- distribution and, 582
- remoting and, 28, 588
- as unnecessary, 23

transparent persistence, 258

transparent recovery (retrying), 13

trigger, Quartz scheduler, 348–349, 351–352

typed advice, 125

U

- `union()` method, `Pointcuts` class, 130
- unit testing**
 - difficulty of, 2
 - example of, 103–106
 - features for, 6, 18
 - guidelines for, 160, 161, 593
 - mock objects used for, 104–106
- `UpdateableSqlQuery` class, `JDBC`, 193–194
- update methods, 24**
- update statements, in iBATIS SQL mapping file, 263**

uploading files, 478–481

UrlBasedViewResolver **class, 450–451**

URLEditor **class, 103**

UrlFilenameViewController **class, 459–460**

UrlResource **class, 85, 92**

URLs, as resources, 84

useCacheControlHeader **property, WebContent
Interceptor class, 448**

useExpiresHeader **property, WebContent
Interceptor class, 448**

UserRoleAuthorizationInterceptor **class,
448–449**

V

validateOnBinding **property, SimpleForm
Controller class, 469**

validatePage() **method, AbstractWizard
FormController class, 475**

validator **property, SimpleFormController class,
468**

value **element, 58–59**

Velocity template engine

configuring view resolver for, 505–506

definition of, 505

form simplification macros, 507–510

integration with, 21

for mail manager, 356

Velocity Template Language (VTL), 505–506

VelocityConfigurer **class, 513**

version release schedule, 29

View **interface, 430, 492, 520**

view objects

definition of, 28, 426

prefixes for, 495–496

provided by Spring, 430

rendering models, 491–492

view resolvers

configuring for FreeMarker, 506–507

configuring for Velocity, 505–506

MVC, 430, 445, 450–454

view technologies

AbstractView **class, 493–496**

caching views, 522

choosing, 491

configuration for, 488–489

custom views, 519–522

document-based, 515–519

example of, 488–491

Excel-based, 515–519

FreeMarker template engine, 489–490, 505–510

iText library, generating PDFs using, 490–491

JavaServer Pages, 489, 496–505

tiles, 511–513

Velocity template engine, 505–510

View **interface, 488**

ViewResolver **interface, 488**

XML-based, 514–515

XSLT-based, 514–515

ViewResolver **class**

definition of, 435, 440

example of, 488

JSP and, 497

multiple instances of, 445

types of, 450–454

Vlissides, John (*Design Patterns*), 597

VTL (Velocity Template Language), 505–506

W

Wanghy Cache, 595

weather data service example

IoC version

Constructor Injection, 43–44

Method Injection, 44–45

Setter Injection, 41–43

JDBC used by, 67–69

non-IoC version, 40–41

web frameworks. See also MVC web frameworks

application dependency on, 6

books about, 597

integration with, 20

web requests, security for, 368, 369–370**Web Service Definition Language (WSDL), 307****Web Services with JAX-RPC**

accessing a service, 326–328

custom application objects, 330–332

definition of, 324–325

exporting a service, 328–330

support classes for, 325

when to use, 325, 333

web tier, 27–28

WebApplicationContext **interface, 50, 430–431,
432, 434–436**

WebApplicationObjectSupport **class, 435–436**

WebContentGenerator **interface, 458**

WebContentInterceptor **class, 447–448**

WebLogic 7.0 and 8.1 server, transaction manager for,
249, 250

WebLogicJtaTransactionManager **class, 250**

WebLogicServerTransactionManager
FactoryBean **class, 250**

websites

Acegi Security, 368

Apache Commons Pool, 150

Apache OJB (Object relational Bridge), 256

AspectWerkz AOP framework, 169

Aurora MVC, 595

Canoo WebTest tool, 106, 485

Cayenne O/R mapping project, 302, 596

EasyMock library, 105, 483

Hibernate, 269

iBATIS SQL Maps, 261

iText library, 490

Jakarta Apache Cactus tool, 106, 421

Jakarta Commons FileUpload package, 478

Jakarta POI, 516

JavaDoc, 597

JDBC RowSet Implementations tutorial, 189

JDBC tutorial, 171

jExcelApi, 516

jMock library, 106

JSF web framework, 525

JSF-Spring, 595

websites (continued)

- MockEJB framework, 420
- Mule framework, 596
- Oracle TopLink, 256
- sample application source code, 551
- SiteMesh, 513
- Spring documentation, 597
- The Spring Framework.NET, 595
- Spring IDE for Eclipse, 595
- The Spring Modules project, 595
- Spring XMLDB, 595
- Struts web framework, 525
- Tapestry web framework, 525
- TestMaker tool (Push2Test), 485
- TopLink, 301
- Velocity macros, 510
- Velocity template engine, 356
- Wanghy Cache, 595
- WebWork web framework, 525, 596
- WebSphere 4.0 and 5.x server, transaction manager for, 249**
- WebSphereTransactionManagerFactoryBean **class, 250**
- WebWork web framework**
 - controllers, 530
 - data binding, 525–526

- definition of, 525, 596
- features of, 525–526, 528, 530, 532
- integration with, 20, 536–537
- interceptors, 530
- presentation layer, 528

wizard-style forms, 474–477

WSDL/SOAP, remoting using, 307. See also Web Services with JAX-RPC

X

XDoclet, 412

XML-based views, 514–515

XmlBeanFactory **class, 52**

XML-configured containers

- alternatives to, 108–110

- format of, 53

- initialization of beans and, 57

- loading, 52

XMLDB, Spring, 595

XmlViewResolver **class, 451, 493**

XmlWebApplicationContext **interface, 85, 435, 484**

XP (Extreme Programming), 18

XSLT-based views, 514–515

XWork SpringObjectFactory, 536–537