

## Symbols

---

- ! operator 65
  - example of 253, 300, 369
- .. wildcard 66
  - example of 31, 138
  - in method signatures 70
  - use in type signature pattern 259–261
- @Access annotation 422
- @AdviceName annotation 90, 179
- @After annotation 170, 179, 185
  - example of 274
- @AfterReturning
  - annotation 170, 179, 185
  - returning attribute 186
  - throwing attribute 186
- @AfterThrowing
  - annotation 170, 179, 185
  - example of 195
- @Aggregate annotation 454
- @AggregateRoot
  - annotation 453
- @annotation() pointcut 84, 101
  - example of 262
  - Spring AOP 231
- @args() pointcut 84, 101
  - context collection 99
  - Spring AOP 231
- @Around annotation 170, 179, 187
  - example of 284, 380, 396
  - Spring AOP, example of 233
- @Aspect annotation 38, 170, 172, 273

- example of 37, 396
- Spring, example of 221
- @AspectJ 184–186
  - abstract aspect 172
  - abstract pointcut 174–175
  - after advice 184–187
  - aspect association 173
  - aspect inheritance 172
  - aspect instance, accessing 173
  - aspect mapping 172
    - constructor restriction 172
    - general 172
    - generic parameter restriction 173
    - inheritance restriction 172
  - aspect precedence 173
    - deviation from traditional syntax 174
  - aspect, nested class restriction 173
- Aspects class 173
- before advice 180–184
- compilation,
  - recommendation 177
- concrete pointcut 175
- empty method body 175
- driving principles 170
- early error detection 171
  - Java compatibility 171
- early error detection example of 171
  - hypothetical syntax, comparison 171
- ease of learning 170
- IDE, use of 171
- integration into Spring Framework 221, 229–235

- Java compatibility
  - binary weaving 171
  - load-time weaving 171
  - tooling compatibility 171
- mapping illustration 169
- natural mapping, example of 170
- pointcut expression, similarity with traditional syntax 170
- pointcut mapping 174–179
  - abstract pointcut 174
  - concrete pointcut 175
- ProceedingJoinPoint 187
- similarity with Java elements 170
- Spring, example of 221
- subaspect 172
- supported elements 170
- tooling 171
- type signature pattern, difference 66
- @AspectJ advice
  - anonymous pointcut, with 180
  - join point 181
  - join point context 182
  - name 180
  - named pointcut, with 181
  - reflective join point context 181
- @AspectJ syntax 16, 31
  - adoption 37
  - ease 197
  - role in 460
- advice syntax 179
- ajc vs. javac 38

- `@AspectJ` syntax (*continued*)
  - around advice 187–189
  - associating annotation 194
  - authentication aspect,
    - example of 410
  - basics 170
  - binary weaving 204
  - choosing, vs. traditional 215
  - concrete compilation
    - requirements 176
  - concrete pointcut, example of 176
  - dual purpose
    - example of 410
    - transaction
      - management 379
  - ease of learning 196
  - exception softening 195
  - expression power 196
  - fully qualified type 176
  - g flag 177
  - `if()` pointcut 177
    - method body 177
    - method declaration 178
    - reason for deviation 179
    - special forms 178
  - import statements 176
  - inter-type declarations 190
  - member introduction 194
  - overview 169–171
  - pointcut expression 176
  - privileged aspect 196
    - alternative 196
  - recommendation 196
  - Spring AOP 218
    - subset 229
  - Spring, example of 45
  - static crosscutting 189–194
  - toolability 39
  - tooling 196
  - traditional syntax,
    - comparison 196
  - unimplemented
    - features 194–196
      - associating annotation 194
      - exception softening 195
      - member introduction 194
      - privileged aspects 196
    - verbosity 196
- `@Async` annotation 360
- `@Audit` annotation 262
- `@Autowired` annotation
  - 304, 437, 477
  - example of 442, 478
  - simplifying code 437
- `@Before` annotation 38, 170,
  - 179–180, 183
  - example of 37, 183, 274, 410
  - Spring, example of 221, 235
- `@Cacheable` annotation 110, 158
  - use of 342
- `@Configurable` annotation 435
  - alternative 438
  - example of 435, 442
  - performance 438
  - static crosscutting, use of 132
  - use of 342
  - use of `initialization()`
    - pointcut 60
- `@Configurable` object, unit-testing of 437
- `@ContextConfiguration` annotation 436
- `@Controller` annotation
  - 262, 477
  - example of 483
- `@DeclareError` annotation
  - 170, 189
  - example of 190
  - restrictions 190
- `@DeclareParents`
  - annotation 170, 191
  - `defaultImpl` attribute 192
  - example of 191
  - marker interface, with 191
  - non-marker interface,
    - with 191
  - Spring AOP 235
  - Spring, example of 235
  - typecast, need for 193
  - using parent interface 193
- `@DeclarePrecedence` annotation
  - example of 418
  - Spring AOP, and 233
- `@DeclareWarning` annotation 170, 189
  - example of 190
  - restrictions 190
- `@Entity` annotation 261
  - example of 342, 470
- `@GeneratedValue` annotation,
  - example of 470
- `@HIPAAData` annotation 262,
  - 422
- `@Id` annotation, example of 470
- `@Idempotent` annotation 97
- `@Inherited` annotation,
  - selection in pointcut 84
- `@Interceptors` annotation 21
- `@ManyToOne` annotation,
  - example of 471
- `@MappedSuperclass` annotation,
  - example of 470
- `@Mock` annotation 158
- `@OneToMany` annotation,
  - example of 473
- `@OneToOne` annotation,
  - example of 474
- `@Order` annotation 234
  - use of 418
- `@Override` annotation 38
- `@PersistenceContext` annotation 306
  - example of 476
- `@Pointcut` annotation
  - 38, 170, 174
  - `argNames` 177
  - example of 37, 274, 380,
    - 396, 410
    - Spring, example of 221, 235
- `@PostConstruct` annotation 438
- `@PreAuthorize` annotation 428
- `@PreDestroy` annotation 438
- `@ReadOnly` annotation 369
  - use of 454
- `@ReadWriteLockManaged` annotation 369
- `@Repository` annotation
  - 303, 477
  - example of 476
- `@RequestMapping` annotation,
  - example of 483
- `@Required` annotation 438
- `@Retention` annotation,
  - example of 454
- `@RunWith` annotation 436
- `@Secured` annotation 420
- `@Sensitive` annotation, use of 342
- `@Service` annotation 477
  - example of 478
- `@Slow` annotation 337
  - use of 364
- `@Stateless` annotation 369
- `@Static` annotation 369
- `@SuppressWarnings` annotation 38
- `@Sync` annotation 360
- `@Table` annotation, example of 470
- `@target()` pointcut 84, 101
  - context collection 99
  - reflection, and 113
  - Spring AOP 231

@this() pointcut 84, 101  
 context collection 99  
 reflection, and 113  
 Spring AOP, lack of 231  
 @ThreadSafe annotation 360  
 @Timing annotation 340  
 use of 364  
 @TraceMe annotation 262  
 @Transactional annotation 264  
 annotation-driven participant  
 pattern, example of 337  
 @WebService annotation 262  
 @within() pointcut 84, 101  
 context collection 99  
 Spring AOP 231  
 @withincode() pointcut 84, 101  
 context collection 99  
 \* wildcard 66  
 bean pointcut 231  
 example of 31  
 use in type signature  
 pattern 259–261  
 && operator 65  
 %C log4j pattern 256  
 %F log4j pattern 256  
 %L log4j pattern 256  
 %l log4j pattern 256  
 %M log4j pattern 256  
 %m log4j pattern 266  
 %n log4j pattern 266  
 %X log4j pattern 280  
 %x log4j pattern 265  
 + wildcard 66  
 example of 35, 300  
 <aop:advisor>, example of 482  
 || operator 65  
 example of 141, 143, 146,  
 260, 300

## A

abc. *See* AspectBench compiler  
 abstract aspect 138  
 example of 138, 267, 380, 414  
 generic parameter, and 140  
 inheritance limitation 139  
 use of 351  
 abstract bean 436  
 abstract method  
 abstract aspect, and 138  
 aspect in, example of 381  
 abstract pointcut 138  
 example of 143, 145, 267,  
 351, 367, 396, 410, 414

AbstractInterface-  
 DrivenDependency-  
 InjectionAspect 439  
 abstraction, AOP costs 24  
 access control  
 factory pattern, and 301  
 plain Java, limitations of 299  
 using AspectJ 298–302  
 access control list 428  
 access specification  
 member introduction,  
 and 120  
 pointcut, and 32  
 AccessDecisionManager  
 408, 415  
 AccessDecisionVoter 408  
 AccessibleObject.setAccessible()  
 196  
 accidental use, enforcing 312  
 Acegi. *See* Spring Security  
 ACID, properties of transaction  
 management 373  
 ACL. *See* access control list  
 adjective, crosscutting concern,  
 and 464  
 adoption  
 @AspectJ syntax 37  
 complexity, Spring AOP vs.  
 AspectJ 245  
 path 23  
 Spring AOP 218  
 ADPP. *See* annotation-driven par-  
 ticipant pattern  
 adverb, crosscutting concern,  
 and 464  
 advice 92–99  
 after (finally). *See* after  
 (finally) advice  
 after returning. *See* after  
 returning advice  
 after throwing. *See* after throw-  
 ing advice  
 after. *See* after advice  
 anatomy of 88  
 around. *See* around advice  
 before. *See* before advice  
 body 90  
 categories 88  
 classification 88  
 collecting context 99–105  
 concept 12  
 CSS, analogy 13  
 declaration 89–90  
 defining 53  
 definition of 32  
 depiction using sequence  
 diagram 89  
 exception, throwing 91  
 exposed join point,  
 restriction 88  
 methods 90–91  
 multiple, ordering of 159  
 name of 90  
 name, @AspectJ 180  
 ordering 161, 164  
 overview 88–90  
 passing context to 99–105  
 precedence 159, 161  
 triggers, analogy 15  
 weaver mapping 42  
 advice execution join point 61  
 adviceDidNotMatch,  
 warning 488  
 advice-execution join point  
 pointcut syntax 76  
 adviceexecution pointcut 76  
 advisor, Spring AOP 226  
 AffirmativeBased 409  
 after (finally) advice 92  
 after advice 92  
 @AspectJ 184–187  
 example of 62, 120, 122  
 exceptional return, on 93  
 successful return, on 93  
 variation 92  
 after returning advice 92  
 example of 120, 122  
 after throwing advice 93  
 exception processing 94  
 use in exception logging 275  
 After. *See* @After annotation  
 AfterInvocationManager 429  
 AfterReturning. *See* @After-  
 Returning annotation  
 AfterThrowing. *See* @After-  
 Throwing annotation  
 aggregate boundary  
 concept 453  
 illustration 453  
 aggregate root  
 concept 453  
 .aj extension 31  
 ajbrowser 48  
 ajc 29, 201  
 @AspectJ syntax with 38  
 -aspectpath option 204  
 -inpath option 203  
 javac, differences 39, 202  
 -outjar option 204  
 -outxml option 213

- ajc (*continued*)
  - outxmlfile option 213
  - showWeaveInfo option 202
  - source code weaving,
    - example of 202
  - sourceroots option 202
  - Xlint option 202
  - Xlint:warning 163
- ajdoc 48
  - sample output 49
- AJDT 16, 221
  - Equinox Aspects, use of 208
  - installation 46
  - internal use of AspectJ 47
  - policy enforcement, and 293
- AMS. *See* SpringSource Application Monitoring Suite
- AND operator 210
- anemic domain model, example of 471
- annotating method, example of 338
- annotating type, example of 338
- annotation
  - @Inherited, selection with 84
  - AOP adoption, and 463
  - bridging 131
  - clutter, static crosscutting,
    - avoiding through 130
  - constructs 131
  - custom
    - example of 453
    - transaction
      - management 390
  - definition of 67
  - functionality-specific 339
  - implementing core
    - concern 465
  - macro-like facility, with 339
  - monitoring, join point
    - selection 261
  - objective characteristic
    - based 340
  - retention 67
  - subjective characteristic
    - based 339
  - supplying
    - avoiding annotation
      - clutter 130
    - example of 131
    - static crosscutting 130–132
  - type-level 387
  - using in pointcut 52
  - XML vs. 465
- annotation attribute, transaction management 387
- Annotation Processing Tool (APT) 18
- annotation type
  - abstraction 338–341
- annotation type pattern, use of 129
- annotation type signature
  - examples 68
  - inheritance 68
- annotation-based method
  - signature 72
  - examples 72
- annotation-based pointcut 84
  - example of 98, 337, 369, 456
  - retention policy 84
  - Swing thread safety, and 360
- annotation-driven authorization
  - subaspect, without
    - annotation 419–421
- annotation-driven participant
  - pattern 336–342
- asynchronous routing,
  - and 364
  - current solution 337
  - example of 369, 397
  - illustration 336
  - library aspect, use of 342
  - retry aspect, and 397
  - template 337–338
- anonymous pointcut
  - using in a pointcut 65
  - using in an advice 65
- Ant
  - aspect library, creating 487
  - weaving sources, using 486
- ant
  - weaving jars, using 489
- AOP
  - adoption
    - @AspectJ syntax 460
    - applications 460
    - AspectJ 460
    - crosscutting domain
      - aspects 462
    - debugging helper 461
    - design phase 464
    - development aspects 460
    - enforcing policies 461
    - implementation phase 464
    - infrastructure aspects 462
    - legacy projects 467
    - maintenance phase 466
  - monitoring
    - performance 461
    - process 462
    - production aspects 462
    - refactoring aspects 462
    - Spring AOP 460
    - syntax choices 460
    - technologies 459
    - testing helper 461
    - weaving models 460
  - alternatives 17–23
    - differentiating from 11
  - analogy, by 13–15
  - anatomy of language 9–11
  - benefits 23–25
    - cleaner implementation 25
    - code reuse 25
  - costs 23–25
    - complex program flow 24
    - need for greater skills 24
  - CSS, analogy 13
  - database, analogy 14
  - event-oriented programming,
    - analogy 15
  - fundamental concepts 11–12
  - generic model 13
  - implementations 15–17
  - inventors 15
  - language
    - base programming
      - language 9
    - implementation 9–10
    - implementation of
      - concerns 9
    - weaving rules
      - specification 9
  - language specification 9
  - managing concerns
    - without 4–7
  - modularization using 7
  - motivation 4
  - relation to XP 25
  - separation of responsibility 25
  - SQL, analogy 14
  - system, choosing 245–247
  - triggers, analogy 14
- AOP alternatives
  - code generation 18
  - design patterns 19–22
    - chain of responsibility 19
    - decorator 20
    - interceptor 21
    - observer 19
    - proxy 20

- AOP alternatives (*continued*)
  - dynamic languages 22
  - frameworks 17
- AOP design pattern
  - annotation-driven participant pattern. *See* annotation-driven participant pattern
  - participant pattern. *See* participant pattern
  - worker object pattern. *See* worker object pattern
  - worker object pattern
  - wormhole pattern. *See* wormhole pattern
- AOP model
  - generic, AspectJ mapping 31
- <aop:advisor> 237
  - example of 391
- <aop:after> 238
- <aop:after-returning> 238
- <aop:after-throwing> 238
- <aop:aspect> 237
- <aop:aspectj-autoproxy> 227, 229
  - example of 45, 274, 389, 413
- <aop:aspectj-autoproxy/> 221
- <aop:before> 238
- <aop:config> 237
  - example of 391, 482
- <aop:declare-parents> 240
- <aop:include> 229
- <aop:pointcut> 237
  - example of 391, 482
- aop.xml 41, 206
  - aspect precedence 210
  - directory placement 206
  - example of 41, 209, 272
  - fully qualified type names 210
  - weaving options 211
- AopContext.currentProxy()
  - method 228
- API, reflection. *See* reflection API
- application framework, policy enforcement, and 293
- ApplicationContext 45, 220
- APT. *See* Annotation Processing Tool
- Aquarium 22
  - AOP implementation 17
- argNames 177
  - @Pointcut 177
- arg-names attribute, Spring AOP 238
- args pointcut 101
  - @AspectJ syntax, example of 182
  - example of 120
  - Spring AOP 230
- args() pointcut 83
  - context collection 99
  - examples of 83
- argument pointcut 83
  - examples of 83
- Around. *See* @Around annotation
- around advice 94–99
  - @AspectJ 187–189
  - accommodating different return types 98
  - example of 33, 89–90, 95–97, 103, 120, 122, 154
  - exception, throwing from 96
  - field-read join point 96
  - primitive return value 96
  - proceed(), and 94
  - return value 95
  - usage of 94
  - wrapping and unwrapping of 96
- Art of the Metaobject Protocol, The 23
- <aspect> 209
- aspect
  - abstract declaration. *See* abstract aspect
  - abstract aspect
  - abstract, use of 351
  - access specification 138
  - access to private data 166
  - association. *See* aspect association
  - code-analysis tools, comparison 297
  - compiled 43
  - concept 4
  - configuring Spring, using 218
  - configuring with factory-method 240
  - configuring with Spring DI 240
  - constructor, limitations 137
  - creating, process 52
  - crosscutting domain 462
  - data members 137
  - deciding to use 462
  - definition of 32, 137
  - dependency injection, and 139
  - designing 464
  - differences from class 139–140
  - emergency path, with 467
  - enabling, control of 154
  - example of 31
  - externalizing configuration, example of 153–159
  - general form 137
  - generic parameter, and 140
  - implementing 464
    - AOP adoption 467
    - process 463
  - infrastructure 462
  - infrastructure vs. domain-specific 431
  - inheritance 138
    - limitation 139
  - instance, accessing 151, 173
  - instantiation 139
  - instantiation models 234
  - inviting, participant pattern, and 333
  - mapping to Java class 43
  - method members 137
  - monitoring
    - deployment options 271–275
  - narrowly scoped 451
  - necessary for correctness 466
  - nested
    - access specification 138
    - example of 127, 449
    - See also* nested aspect
  - ordering
    - Spring AOP 233
  - prebuilt
    - Spring AOP 460
    - Spring Security 425–428
  - precedence 173
    - explicit 162–164
    - XML syntax 210
  - privileged 140, 166
    - avoiding overuse 167
    - example of 167
  - production 462
  - refactoring 260, 468
  - referring to instance using this 91
  - reusable 468
    - aspect association, and 153
  - similarities with class 137–139
  - single, ordering advice in 164
  - testing 105, 348, 422
    - example of 157
    - using mock object 153, 157
  - third-party 200
  - tightly bound 451

- aspect (*continued*)
  - using as debugging helper 461
  - using during development 460
  - using for performance monitoring 461
  - using for policy enforcement 466
    - AOP adoption 461
  - using for refactoring 462
  - using to help with testing 461
  - using to implement new features 467
  - weaver mapping 42
  - working with 137–140
- aspect association 140–153, 173
  - accessing aspect instances 151
  - default association. *See* default association
  - implicit limiting of join points and, 150
  - kinds 140
  - per-control-flow association. *See* per-control-flow association
  - per-object association. *See* per-object association
  - per-type association. *See* per-type association
  - syntax 140
- aspect library
  - binary weaving 204
  - creating
    - Ant, example of 487
    - Maven, example of 492
    - Swing 316
- aspect precedence 159
  - aop.xml 211
  - example of 418
  - Swing thread safety example 354
- aspect precedence rule, retry aspect, and 397
- aspect state, aspect association, and 140
- aspect weaver, concept 4
- Aspect, as a controller 113
- Aspect. *See* @Aspect annotation
- Aspect# 17
- AspectBench compiler 16
- AspectJ
  - adoption complexity, vs. Spring AOP 245
  - adoption, policy enforcement, and 289
  - alternative syntax 37–39
  - AOP, adoption, role in 460
  - binary weaving 39
  - compared to Spring AOP 245
  - compiling sources 29
  - configurability, vs. Spring AOP 246
  - crosscutting constructs 31–37
  - downloading and installing 28
  - elements, mapping of 42
  - integration in Spring 44–46, 221
  - internal details vs. language semantics 42
  - internals 42–44
  - introduction 15
  - join point model, vs. Spring AOP 245
  - load-time weaving 41
  - logistics, overview 46–48
  - Maven plugin 491
  - mojo 491
  - overview 27
  - performance, vs. Spring AOP 246
  - pseudo keywords 34
  - relation to Java 27
  - running programs 29, 40
  - Scala, with 16
  - simple program 28–31
  - source weaving 39
  - Spring, simplifying using 218
  - syntax
    - @AspectJ 31
    - choices 31
    - guidelines 215
    - traditional 31
    - weaving mechanism, orthogonality 39
  - tracing, graphical illustration 258
  - vs. Spring AOP, decision process 246
  - weaver 39–42
    - and Spring 240–245
  - weaving model, guidelines 215
  - weaving, vs. Spring AOP, guidelines 245–247
  - woven code performance 44
  - XP, and 450
- AspectJ ant task, defining 486
- AspectJ design pattern
  - annotation-driven participant pattern. *See* annotation-driven participant pattern
  - participant pattern. *See* participant pattern
  - worker object pattern. *See* worker object pattern
  - wormhole pattern. *See* wormhole pattern
- AspectJ Development Tools (AJDT). *See* AJDT
- aspectj-maven-plugin
  - example of 492
- aspectjrt.jar 201
  - @AspectJ syntax annotations 38
- aspectjweaver.jar 241
  - use of 41
- aspectLibraries, Maven, and 495
- aspectOf() method
  - @AspectJ syntax 173
  - accessing aspect instance 151
  - aspect association, and 151
  - configuring aspect 240
  - example of 152, 154, 158, 285, 413, 418, 440
  - limitation, with Spring 241
  - Spring, use in application context 154
- Aspect-Oriented C, AOP implementation 17
- aspect-oriented programming. *See* AOP
- aspect-oriented tracing 253–254
- aspectpath flag 489
  - ajc option 204
  - example of 40
- Aspects 173
- <aspects> 42, 209
- Aspects.aspectOf() method, example of 383–384
- AspectWerkz 16, 37, 169
- asynchronous call, example of 346
- asynchronous routing, worker object pattern, and 364
- atomicity, property of transaction management 373
- auditing
  - security, and 429
  - Spring AOP, using 221
  - using logging 296

augmenting existing build system 205  
 Authentication 408  
 authentication 29  
   just in time 415  
   Spring Security 407  
   illustration 408  
 authentication aspect  
   implementing 410–413  
 AuthenticationException 411  
 Author Online 468  
 authorization  
   base aspect 414–416  
   field-level,  
     implementing 421–425  
   Spring Security 408  
   illustration 409  
   subaspect  
     with annotation 419–421  
     without annotation 416–419  
 authorization aspect  
   illustration of 414  
   implementing 413–421  
 autoproxy mechanism, Spring AOP 227  
 autowiring 435  
 AWT thread 313  
 AWT, EJB programming restrictions, and 308

**B**

---

base aspect, example of 395  
 bean pointcut  
   \* wildcard 231  
   combining 232  
   examples of 232  
   illustration 232  
   negating 232  
   selecting beans based on architectural role 231  
   selecting beans based on business functionality 231  
 BeanFactoryPostProcessor 233  
 BeanPostProcessor 233  
 Because 425  
 Before. *See* @Before annotation  
 before advice 92  
   @AspectJ 180–184  
   example of 30–31, 62, 90, 130, 142  
   exception throwing and join point execution 92  
   usage 92

behavior  
   augmenting, participant pattern 335  
   rich 432  
     implementing 433–434  
 best practice  
   aop.xml 213  
   aspect as a controller 113  
   policies, and 294  
   public members 298  
 binary operator 65  
   type signature 69  
 binary weaver, linker,  
   comparison 40  
 binary weaving 39, 200  
   @AspectJ syntax 204  
   @AspectJ, and 171  
   aspect library 204  
   augmenting existing build system 205  
   build-time 202–205  
   example of 40  
   IDE, use of 204  
   illustration 203  
   input  
     compilation of 203  
     -g flag 203  
     -g:vars flag 203  
 bridged participation  
   pattern 341–342  
   illustration 342  
   use of 455  
 bridged participation, example of 371  
 build system, policy enforcement, and 295  
 build-time weaving 200–205  
   binary 202–205  
   inter-type declaration, and 201  
   source code 201  
   illustration 201  
   suitability while learning 201  
   XML syntax 205  
 bulk update, JPA, and 305  
 business functionality, mainline, design of 464  
 business logic,  
   implementing 447–452  
 byte-code weaving  
   transaction management, and 379  
   *See also* binary weaving

**C**


---

C/C++ preprocessor, declare errors and warning, comparison 132  
 C++ 295  
   #error 132  
   #warning 132  
   friend 301  
   mixin, use of 121  
 cache, read-write lock pattern, and 365  
 caching  
   example of 153–159  
   expression language, use of 109  
   scripts, use of 109  
   using aspect 102, 109  
   using reflection API 109  
 call depth  
   definition 62  
   monitoring 264  
 call() pointcut 76  
   example of 36, 134  
 caller context passing  
   current solution 327  
   wormhole pattern 327  
 caller context, MDC, establishing using 280  
 Cascading Style Sheets (CSS). *See* CSS  
 catch block, empty, code-analysis tools and 297  
 cflow() pointcut 78  
   avoiding recursion, example of 449  
   comparison with cflowbelow() 78  
   depiction using sequence diagram 78  
   example of 79, 263, 309, 329, 353, 356  
   security auditing 429  
   wormhole pattern, example of 329  
 cflowbelow() pointcut 78  
   comparison with cflow() 78  
   depiction using sequence diagram 78  
   example of 79, 263, 398, 456  
   selecting top-level join point 264  
   usage of 79

- CGLIB
  - dynamic proxy 224
  - exclusion using pointcut 272
  - Spring AOP, limitations of 226
- chain of responsibility design pattern, AOP
  - alternatives 19
- chain of responsibility, security, and 406
- characteristics-based crosscutting 331
- Checkstyle 297
- circular dependency
  - package structure 443
  - precedence control 164
- class
  - annotating, annotation-driven participation pattern 370
  - woven 43
- class attribute, CSS 13
- class diagram
  - reflection API 106–107
- ClassCastException 225
- class-initialization join point 59
  - comparison with object initialization 60
  - pointcut syntax 76
  - static blocks 59
- ClassPathXmlApplication-Context 45, 155, 220
- Clover 171
- code
  - refactoring 467
  - reuse, AOP benefits 25
  - well-factored, writing 465
- code fragment, duplicated,
  - code-analysis tools and 297
- code generation, AOP
  - alternatives 18
- code review, policies, and 294
- code scattering 7, 30, 463
- code tangling 5–6, 30, 463
  - illustration 6
- code-analysis tool
  - aspect, comparison 297
  - policy enforcement, and 297
- collaboration, limiting 299
- compile-time
  - errors and warnings 36, 132
  - example of 133
- compile-time enforcement 293, 295
  - limitations 295
- complexity
  - managing with abstraction 24
  - software system 3
- concern
  - composition of 4
  - concept 3
  - core. *See* core concern
  - crosscutting. *See* crosscutting concern
  - managing without AOP 4–7
  - tangling 5
- concern space,
  - multidimensional 6
- concrete aspect
  - example of 439
  - using XML, example of 272
- <concrete-aspect> 209
- concurrency
  - characteristics, monitoring 281
  - code-analysis tools 297
- ConcurrencyFailureException 399
- conditional check pointcut 85
- configurability, Spring AOP vs. AspectJ 246
- configuration, dependency injection 219
- ConsensusBased 409
- consistency, property of transaction management 373
- constructor join point 57
  - call 58
  - execution 57
  - execution vs. call 58
  - pointcut syntax 76
- constructor signature
  - similarity to method signature 73
- constructor signature pattern 69
- constructor, aspects, and 137
- context
  - dynamic 105
  - static 105
- context collection 99–105
  - after returning advice 93
  - after throwing advice 93
  - collecting exception object 93
  - collecting field set value 83
  - collecting return value 93
  - example, caching 102–105
  - implicit restriction on selection 100
- observer design pattern 19
- pointcut vs. reflection 113
  - using @args() pointcut 99
  - using @target() pointcut 99
  - using @this() pointcut 99
  - using @within() pointcut 99
  - using @withincode() pointcut 99
  - using anonymous pointcut 100
  - using args() pointcut 83, 99
  - using named pointcut 100
  - using target() pointcut 81, 99
  - using this() pointcut 81, 99
- context passing, method parameters, through 327
- context:classpath-scanning,
  - example of 477
- <context:component-scan>,
  - example of 283
- <context:load-time-weaver> 241
- <context:property-placeholder>,
  - example of 482
- <context:spring-configured>,
  - dependency injection 436
- context.xml, Spring-driven LTW, and 242
- continuation, worker object, and 324
- contract, enforcement 296
- Contract4J 296
- control flow, concepts 77
- control-flow based pointcut 77
  - examples of 79
- Controller Spring MVC
  - class 261
- COR. *See* chain of responsibility design pattern
- core concern
  - concept 3
  - implementing with AOP 464
- correctness, of software system 466
- Cross References view 47
- Crosscut Programming Interface 291
- crosscutting
  - characteristics-based 331
  - dynamic 32
    - concept 12
    - database 14
  - dynamic, construct, mapping 179–189
  - fine-grained 247
  - noninvasive 416

- crosscutting (*continued*)
    - static, concept 12
    - viewing in an IDE 47
  - crosscutting concern
    - choosing underlying technology 465
    - client-side modularization 7
    - concept 4
    - definition 4
    - implementing with AOP 465
    - modularizing, using Spring AOP 222
    - pseudocode example 4
    - refactoring 467
    - separating from core modules 465
    - server-side modularization 7
    - weaving 8
  - crosscutting construct 31–37
    - common 31
    - dynamic 32
    - static 34
  - crosscutting domain aspect 462
  - crosscutting functionality, design of 464
  - CRUD application 442
  - CSS 13
    - class attribute 13
    - scattering 14
    - selector 13
    - tangling 14
- D**
- 
- DAO. *See* Data Access Object
  - Data Access Object 303, 475
    - Spring AOP 247
  - data carrier 432
  - data introduction. *See* member introduction
  - Data Transfer Object 452
  - DataAccessResourceFailureException 399
  - database 14
    - SQL 14
    - triggers 14
  - data-driven authorization, EJB, limitations of 406
  - DataSourceTransactionManager 392
  - Dcom.sun.management.jmxremote 285
  - DDD. *See* Domain Driven Design
  - deadlock, detection, load-time weaving, example 206
  - debugger, concurrency, and 461
  - debugging helper aspect 461
  - debugging, AJDT 47
  - declaration
    - inter-type. *See* inter-type declaration
    - weave-time, definition of 36
  - declare @constructor 131
  - declare @field 131
    - bridged participation pattern, use of 341
  - declare @method 130–131
    - bridged participation pattern, use of 341
    - example of 455
  - declare @type 131
    - annotation bridging 131
    - bridged participation pattern, use of 341
    - example of 131, 310, 341, 371
  - declare annotation
    - @Target, and 131
    - availability to other tools 132
  - declare error 132
    - example of 133, 290, 309
    - policy enforcement example 300
    - statically determinable point-cut, and 132
    - syntax 132
  - declare parent, aspect association, comparison with 153
  - declare parents 128
    - @AspectJ syntax 190
    - annotation type pattern 129
    - example of 34–35, 128, 130
    - with annotation type pattern 130
  - declare precedence
    - @AspectJ syntax 173
    - example of 162, 399
    - Swing thread safety, example of 354
    - syntax 162
  - declare soft
    - example of 134
    - syntax 133
  - declare warning 132
    - example of 36, 133
    - statically determinable point-cut, and 132
    - syntax 132
  - DeclareError. *See* @DeclareError annotation
  - DeclareParents. *See* @DeclareParents annotation
  - DeclarePrecedence 173
  - DeclareWarning. *See* @DeclareWarning annotation
  - decorator design pattern 319
    - AOP alternatives 20
  - default association 141–143
    - example of 141
    - NoAspectBoundException, behavior 151
  - dependency
    - circular, precedence control 164
    - hard-coded 433
    - injecting with Spring and AspectJ 434–441
  - dependency injection 431, 434
    - annotation-based 435–438
    - aspect, example of 153–159, 397
    - authentication aspect 410
    - configuration 219
    - container, Spring, and 434
    - domain interfaces-based 438–441
    - domain object 434, 442–447
      - abstract bean 436
      - AOP adoption, and 462
      - API, simplifying 446
      - illustration 435
      - package arrangement 443
      - prototype scope 436
      - service layer, and 445
    - explained 219
    - proxy creation 225
    - rich domain model, and 433
    - transaction management, and 381
  - Dependency Inversion Principle 258
  - dependency lookup 219
  - deployment phase
    - monitoring, use of 251
  - design pattern
    - AOP alternatives 19–22
    - definition of 319
    - policies, and 294
  - design, simplified, AOP
    - benefits 24
  - development aspect 460
    - policy enforcement 297
  - development phase, monitoring, use of 251

- developmental aspect, policy enforcement 289
  - DI. *See* dependency injection
  - documentation, policy enforcement, using 294
  - Domain Driven Design 432
  - domain model 432
    - anemic 432
    - CRUD application, and 442
    - example of 432
    - exposed 452
  - domain object
    - API, simplifying 446
    - behavior, enriching 442–445
    - dependency injection 434
    - illustration 435
    - security 428
  - domain object dependency injection
    - AOP adoption, and 462
  - DomainEntity class, source code 470
  - domain-object dependency injection
    - static crosscutting, use of 132
  - domain-specific language (DSL) 17
  - Dorg.aspectj.weaver.load-time.configuration. *See* aop.xml
  - Drools, use with aspects 117
  - DSL. *See* domain-specific language
  - DTO. *See* Data Transfer Object
  - <dump> 212
  - duplicated code fragment, code-analysis tools 297
  - durability, property of transaction management 373
  - dynamic context 105
    - monitoring, join point selection 263
  - dynamic crosscutting 88
    - concept 12
    - construct, mapping 179–189
    - database 14
    - member introduction, use of 120
    - runtime policy enforcement, and 296
    - Spring AOP 230–235
    - use of pointcut 33
    - vs. static crosscutting, evaluation order 128
  - dynamic language, AOP
    - alternatives 22
  - dynamic proxy 223–224
    - creation of 223
    - difficulties 224
    - example of 223
    - explicit proxy creation 224
    - join point context 224
    - lack of pointcut language 224
    - typecast, need for 224
- E**
- 
- EasyMock 158
  - Eclipse, AspectJ integration 46
  - EclipseLink 306
  - e-commerce
    - anemic domain model, example of 432
    - application configuration, example of 479–482
    - building 484
    - DAO, example of 475–477
    - domain classes, example of 469–475
    - logging configuration, example of 484
    - repository layer, example of 475–477
    - running 485
    - service layer, example of 477–479
    - web layer, example of 482–484
  - e-commerce application
    - inventory management 447
    - updating inventory 442
  - EJB
    - 2.0 specification 308
    - 3 309
      - annotation-based, policies 310
      - XML-based, policies 312
    - enforcement aspect 309
    - interceptor design, problems 22
    - policy enforcement 308–313
    - programming restrictions
      - AWT, and 308
      - native library loading 312
      - socket creation 312
      - static fields 310
      - System.in access 312
      - threading 308
    - security, implementing 21
    - transaction management, implementing 21
  - EJB framework
    - AOP alternatives 18
    - security, and 406
  - EJB3, interceptor design
    - pattern 21
  - EJBPointcuts, library aspect 310
  - Emacs 47
  - emergency patch, aspects, with 467
  - Emma 197
  - enclosing context 106
  - enterprise services
    - abstraction 217
  - entity manager, proxy 388
  - EntityManager, tracing 386
  - equals() method, excluding using pointcut 269
  - Equinox Aspects 208
  - Erlang 344
  - error
    - detection, early 171
    - weave-time, weaver mapping 42
  - #error, weave-time error and warning, comparison with 132
  - evaluation order, static vs. dynamic crosscutting 128
  - event-dispatching thread 313, 346
  - event-oriented programming 15
  - EventQueue.invokeAndWait() method 345, 349, 356
  - EventQueue.invokeLater() method 345, 349, 356
  - EventQueue.isDispatchThread() 315
  - every 464
  - exception handling, transaction management 382
  - exception logging aspect 275
  - exception monitoring 275–277
  - exception softening 133–134
  - exception, checked
    - runtime exception, converting into 133
    - softening 133–134
  - exception-handler execution
    - join point 59
  - exception-handler join point
    - pointcut syntax 76
  - <exclude> 212

execution() pointcut 76  
   example of 119, 141, 146,  
   259, 356  
   Spring AOP 230  
 execution-object pointcut 80  
   examples of 82  
 Executor, use in asynchronous  
   routing 364  
 explicit aspect precedence  
   162–164  
 explicit proxy creation, dynamic  
   proxy 224  
 exposure, restricting, via policy  
   enforcement 298  
 expression, aop, pointcut  
   attribute 237  
 extension, for aspect source  
   code 31  
 extract method call 451  
 Extract Method refactoring,  
   tracing, and 267  
 Extreme Programming. *See* XP

## F

---

façade, service layer as 445  
 factory design pattern 319  
 factory pattern, enforcing 301  
 factory-method attribute 240  
   creating aspect bean 154  
   example of 285, 413, 418, 440  
 fault injection  
   aspect, using 400  
   Jexin, library aspect 401  
 fault tolerance  
   base aspect 395  
   nontransactional  
   resources 401  
   testing 399  
   transaction management,  
   and 395–402  
 FFDC. *See* first failure data  
   capture  
 field signature 73  
   annotation-based 74  
   examples of 74  
   basic 74  
   examples 74  
   use in type signature 73  
 field signature pattern. *See* field  
   signature  
 field, content, filtering 429  
 field-access join point 58  
   pointcut syntax 76  
   read access 58  
 field-get join point, getArgs()  
   and, 108  
 field-level authorization  
   aspect 424  
 field-read join point  
   around advice, return  
   value 96  
   pointcut syntax 76  
   return value, around  
   advice 96  
 field-set join point, getArgs()  
   and, 108  
 field-write join point, pointcut  
   syntax 76  
 filter, servlet, chain of responsi-  
   bility design pattern 20  
 filtering, field content 429  
 FindBugs 171, 289, 297  
 fine-grained crosscutting 247  
 first-failure data capture 275  
 flexible access control  
   e-commerce control, use  
   of 299  
   e-commerce example 299  
   factory pattern, and 301  
   friend (C++) in Java 301  
   policy enforcement 298–302  
 forums 468  
 fragile pointcut 448  
 framework, AOP alternatives 17  
 friend (C++) 301  
 fully qualified type  
   @AspectJ syntax 176  
   @DeclarePrecedence 174  
 fully qualified type name, XML  
   syntax 210  
 functionality-specific  
   annotation 339

## G

---

-g flag 238  
   binary weaving 203  
 -g:vars flag 238  
   binary weaving 203  
 generic model, AOP 13  
 generic repository 475  
 generic type signature, examples  
   of 68  
 generic-based type signature  
   pattern 68  
 get() pointcut 76  
 getArgs() 108  
   example of 268  
   field-get join point 108

field-set join point 108  
 handler-execution join  
   point 108  
   primitive argument 108  
 getKind() 108  
 getLine() method 254  
 getSignature() 108–109  
   example of 315, 347  
   use in logging 268  
 getSourceLocation() 108–109  
   example of 315  
 getSourceLocation()  
   method 254  
 getStaticPart() 106–108  
 getTarget() 108  
   returning null 108  
   static methods 108  
 getThis() 108  
   example of 268  
   returning null 108  
   static methods 108  
 getWithinTypeName()  
   example of 148, 271  
   perTypeWithin(), and 148  
 Glassbox 281  
 GlassFish  
   Spring-driven LTW 242  
   support for Spring-driven  
   LTW 244  
 Grails 22  
 Groovy 17, 22  
 GUI application. *See* UI applica-  
   tion  
 GUI, blocking, asynchronous  
   routing, avoiding using 363  
 Guice 17

## H

---

handler() pointcut 76  
 handler-execution join point,  
   getArgs() and, 108  
 hasAspect() 152, 173  
   @AspectJ syntax 173  
   example of 152  
 hasfield() 129  
 hashCode() method, excluding  
   using pointcut 269  
 Haskell 344  
 hasmethod() 129  
   domain-object dependency  
   injection, use in 132  
 health care, field-level  
   authorization 421

Health Insurance Portability and Accountability Act 422  
 Hibernate 469  
   mixing with JDBC,  
   policies 305  
   transaction management 377  
   transactional resource 376  
 Hilsdale, Erik 44  
 HIPAA 422  
 Hugunin, Jim 44  
 humane interface 446

## I

iajc 486  
   defining 486  
 id, aop, pointcut attribute 237  
 IDE  
   binary weaving, and 204  
   integration 46  
   cross references view 47  
   functionality 46  
   Spring AOP support 222  
 idempotent  
   retry aspect, and 97  
   transaction management,  
   and 395  
 idiom  
   !within, use of 80  
   avoiding infinite  
   recursion 254, 269  
   avoiding the advising of aspect  
   elements 80  
   default implementation  
   124–128  
   partial 127  
   providing choice 127  
   default interface implementa-  
   tion  
   example of 440  
   matching subclass join  
   points 83  
   policies, and 294  
   precedence control  
   aspect 164  
   return-value restriction 303  
   selecting top-level join  
   point 79  
 IdTransferringMergeEvent-  
 Listener 481  
 if() pointcut 85  
   @AspectJ syntax 176–177  
   example of 85, 315, 351  
 implementation space,  
   one-dimensional 6

implementation, cleaner, AOP  
   benefits 25  
 import statement, @AspectJ  
   syntax 176  
 <include> 211  
 incremental adoption 460  
   policy enforcement, and 289  
 indentation, in tracing 264–266  
 infinite recursion  
   avoiding by excluding Object  
   methods 269  
   avoiding by using  
   !within() 254  
 infrastructure aspect 462  
   domain-specific aspects,  
   vs. 431  
 inheritance, monitoring, join  
   point selection 261  
 initialization() pointcut 76  
   use of 60  
 initMocks() 158  
 injars 489  
 inner transaction, retry aspect,  
   and 397  
 -inpath flag  
   ajc option 203  
   example of 40  
 installing, AspectJ 28  
 IntelliJ IDEA 48, 196  
 interception, proxy, with 222  
 interceptor design pattern  
   AOP alternatives 21  
   EJB3, use of 21  
 interface  
   default implementation, pro-  
   viding, idiom 124–128  
   humane 446  
   minimal 446  
 InterruptedException 352  
 inter-type declaration  
   @AspectJ syntax 190  
   concept 12  
   definition of 34  
   member introduction 34, 117  
   modifying the type  
   hierarchy 128  
   reason for the name 34  
   weaver mapping 42  
 introduction. *See* inter-type  
 declaration  
 inventory, management, using  
   aspects 447  
 inventory, real-time  
   monitoring 447

InventoryItem class, source  
   code 474  
 InventoryServiceImpl, source  
   code 479  
 inviting aspect, participant pat-  
 tern, and 333  
 InvocationContext 22  
 InvocationHandler 223  
 InvocationTargetException 352  
 invoke() method 223  
   InvocationHandler 223  
 IO operation, pointcut 330  
 isolation, property of transaction  
   management 373  
 issingleton() 141  
 ITD. *See* inter-type declaration

## J

JAAS. *See* Java Authentication  
 and Authorization Service  
 Jakarta Commons Logging 255  
 Jamon 282  
 jar, weaving, ant, example  
   of 489  
 Java  
   annotation 37  
   plain, @AspectJ, and 169  
 Java 5 218  
   Spring AOP, options 218  
 Java Authentication and Autho-  
 rization Service 405, 408  
 Java byte-code specification,  
   woven code 39  
 .java extension 31  
 Java Management Extension  
   aspects, exposing to 153  
   exposing aspect bean 154  
 Java Persistence API 261, 469  
   load-time weaving, and 206  
 Java virtual machine 27  
 Java Virtual Machine Tools  
   Interface 41, 206  
 -javaagent 41  
   Spring-driven LTW 241  
   use of 41  
   VM option 207  
 javac, ajc, differences 202  
 Javadoc 48  
   annotation, AOP with 16  
 javax.ejb.TransactionAttribute  
   393  
 javax.swing.text.Document 316  
 JBoss, AOP implementation 17  
 JBuilder 47

- JComponent 316
  - JConsole 156, 285
  - JDBC
    - transaction management 377
    - transactional resource 376
  - JDBCPointcuts, implementation of 291
  - JDK, dynamic proxy 224
  - Jess, use with aspects 117
  - Jexin 401
  - JMock 158
  - JMS
    - transaction management 377
    - transactional resource 376
  - JMX 285
    - runtime control of
      - aspects 284
    - See also* Java Management Extension
  - join point
    - absence of, policy enforcement, and 295
    - advice execution. *See* advice-execution join point
    - capturing using kinded pointcut 76
    - categories 55–61
    - class initialization. *See* class-initialization join point
    - collecting context 52
    - concept 11, 53
    - constructor. *See* constructor join point
    - context collection 54
    - definition of 32
    - demonstration example 61–64
    - enclosing context 106
    - exception handler. *See* exception-handler join point
    - exposed, advice, on 88
    - field access. *See* field-access join point
    - identifying 52, 465
    - implicit limiting of, and aspect association 150
    - information, accessing 105–113
    - kind 106
    - method. *See* method join point
    - multiple, policy enforcement, and 295
    - object initialization. *See* object-initialization join point
    - object pre-initialization. *See* object pre-initialization join point
    - selecting for monitoring 258–264
      - annotations 261
      - based on dynamic
        - context 263
      - based on static
        - structure 259–263
      - method signature 262
      - package structure 259–261
      - type structure 261
    - sequence diagram
      - illustration 53
    - source location 106
    - structural context 106
    - subclass, matching 83
    - subject 84
    - top level
      - selecting 264
      - selecting using
        - flowbelow() 79
  - join point context
    - collection 99–105
    - dynamic proxy 224
    - Spring AOP 225
  - join point model 51
    - concept 12
    - introduction to 52–55
    - Spring AOP vs. AspectJ 245
  - join point selection, inheritance 261
  - JoinPoint 106
    - @AspectJ syntax, use in 182
    - getArgs(). *See* getArgs()
    - getKind(). *See* getKind()
    - getSignature(). *See* getSignature()
    - getSourceLocation(). *See* getSourceLocation()
    - getStaticPart() method. *See* getStaticPart()
    - getTarget(). *See* getTarget()
    - getThis(). *See* getThis()
    - toLongString() 109
    - toShortString() 109
    - toString() 109
    - use in logging 268
  - JoinPoint.EnclosingStaticPart, @AspectJ syntax, use in 182
  - JoinPoint.StaticPart 106
    - @AspectJ syntax, use in 182
  - JoinPointContextUtil 111
  - JPA 450
    - annotations, examples 470
    - domain object dependency, and 433
    - mixing with JDBC, policies 305
    - policies 304
    - tracing 386
    - transaction management 377
    - transactional resource 376
    - See also* Java Persistence API
  - JPAPointcuts, implementation of 292
  - JpaTransactionManager 391–392
  - JTA, use of 449
  - JVMTI 206
    - load-time weaving, with 206
- ## K
- 
- keyword
    - aspects, treatment of 34
    - pseudo 34
  - Kiczales, Gregor 15, 23
  - kinded join point, getKind(), using 108
  - kinded pointcut 75–77
    - combining with &&, warning 76
    - definition of 75
    - syntax 76
- ## L
- 
- language
    - base programming language 9
    - dynamic, AOP alternatives 22
    - implementation 9–10
    - implementation of
      - concerns 9
      - specification 9
      - weaving rules specification 9
  - Law of Demeter 441
  - layered architecture 289
    - transaction management, and 376
  - layering enforcement, using aspects 290
  - LDAP. *See* Lightweight Directory Access Protocol
  - legacy project, AOP in 467

- Level.INFO 255
  - lexical scope
    - concept 79
    - pertypewithin(), and 150
  - lexical-structure-based
    - pointcut 79
    - examples of 80
  - library aspect
    - annotation-driven participant pattern, use of 342
    - EJBPointcuts 310
    - pointcut 260
    - use of 386
  - library, policy enforcement 293
  - Lightweight Directory Access Protocol 407–408, 411, 413
  - LineItem class, source code 471
  - linker, binary weaver
    - comparison 40
  - load-time weaving 38, 41, 200, 206–213
    - @AspectJ, and 171
    - AOP adoption 467
    - <aspect> elements 209
    - <aspects> elements 209
    - classloader-based 206
    - concrete aspect, defining 209
    - <concrete-aspect> elements 209
    - configuration 208–213
    - <dump> elements 212
    - Equinox Aspects 208
    - example of 213
    - <exclude> elements 212
    - illustration 207
    - <include> elements 211
    - Java Persistence API 206
    - javaagent VM option 207
    - JVM TI, and 206
    - monitoring, example 206
    - multiple aop.xml files 213
    - OSGI 208
    - outxml option, and 213
    - outxmlfile option, and 213
    - overview 206–208
    - specifying aspects to weave in 209
    - Spring-driven 240–245
    - steps 207
    - Tomcat 214
    - tracing, use of 271–273
    - use cases 206
    - <weaver> elements 211–212
  - locator pattern 433
  - log4j 255
    - configuration 265, 270
    - layout patterns 256
    - obtaining caller information 256
  - log4j.xml, example of 484
  - logger, type-specific 270–271
  - logging
    - conventional 255–256
    - improving using AOP 277–281
    - problems 256
    - Swing thread safety, example of 347, 353, 356
    - tracing, vs. 257
  - login, up-front, authentication, and 410
  - LOOM.NET 17
  - Lopes, Cristina 15
  - LTW. *See* load-time weaving
- M**
- 
- macro-like facility, annotations, with 339
  - mailing lists 468
  - mainline business functionality, design of 464
  - Mapped Diagnostic Context 255
    - See also* MDC
  - mapping 169
    - @AspectJ 169
  - marker interface, member introduction, and 121
  - Maven
    - aspect library, using 492
    - AspectJ plugin 491
    - build script, example of 469
    - weaving sources, using 491
    - web application
      - building 484
      - running 485
  - MBeanExporter 285
  - MBeanInfoAssembler 156
  - MDA. *See* Model Driven Architecture
  - MDC 265
    - adding context information 280
    - caller context, establishing using 280
    - modularizing, conventional logging, for 279–281
  - MDC.put 280
  - MDC.remove 280
  - member introduction
    - 34, 117–128
    - access specifier 120
    - relative to aspect 120
    - accessing introduced member, example of 35
    - access-specification rules 123
    - conceptual framework 121
    - example of 35, 118–121, 124, 152
    - holistic approach 121
    - interface
      - into 124
      - with implementation 124
    - mangling 123
    - marker interface, use of 121
    - multiple aspects, from 124
    - multiple inheritance 126
    - multiple target type 129
    - overriding 124
      - limitation on super() 124
      - the default
        - implementation 127
    - per-object association, relation to 152
    - responding to messages 121
    - rules 123
    - single target type 124
    - using from other code 130
  - META-INF/aop.xml. *See* aop.xml
  - META-INF/aop-ajc.xml 213
    - See also* aop.xml
  - META-INF/context.xml 242
    - Spring-driven LTW, and 242
  - meta-object protocol 17
  - meta-programming 22
  - method
    - annotating, example of 338
    - parameters, logging 268–270
    - signature, basic 70
  - method attribute, Spring AOP 238
  - method body, advice, and 32
  - method execution join point, Spring AOP 227
  - method join point 56
    - call 56
    - execution 56
    - execution vs. call 57
    - pointcut syntax 76
  - method signature 72
    - annotation-based, examples of 72
    - examples of 70
    - matching of modifiers 70

method signature (*continued*)  
 monitoring, join point  
 selection 262  
 similarity to constructor  
 signature 73  
 use of type signature 69–70  
 method signature pattern 69  
 MethodSignature, example  
 of 382  
 minimal interface 446  
 mixin  
 aspects, using 121–123  
 definition 121  
 Spring AOP, using 235  
 mock object, testing aspects 153  
 Mockito 158  
 example of 457  
 initMocks(), example of 158  
 use of 383  
 Model-Driven Architecture 390  
 Model-View-Controller 259  
 join point selection 259  
 modularity, managing complex-  
 ity with 24  
 modularization 3  
 authorization example 8  
 security example 7  
 using AOP 7  
 mojo, AspectJ 491  
 monitoring  
 application phases, use in 251  
 crosscutting nature 251  
 deployment options 271–275  
 runtime control 284–285  
 selecting join points 258–264  
 based on dynamic  
 context 263  
 based on static  
 structure 259–263  
 by annotations 261  
 by method signature 262  
 by package structure  
 259–261  
 by type structure 261  
 MOP. *See* meta-object protocol  
 multicore processors 344  
 MVC. *See* Model-View-Controller

## N

named pointcut. *See* pointcut  
 naming convention, consistent  
 fragile pointcut 416

importance of, example 316  
 using 465–466  
 native library loading, EJB pro-  
 gramming restrictions 312  
 NDC 265  
 modularizing, conventional  
 logging, for 277–279  
*See also* Nested Diagnostic  
 Context  
 NDC.pop 265, 278  
 NDC.push 265, 278  
 nested aspect  
 example of 125, 127, 129,  
 301, 449, 451  
 policy enforcement, and 302  
 Nested Diagnostic Context 255  
*See also* NDC  
 .NET, AOP implementation  
 16–17  
 NetBeans 47  
 network failure  
 handling using aspect 97  
 simulation of 97  
 networking, EJB programming  
 restrictions 312  
 no-argument constructor,  
 aspects, and 138  
 NoAspectBoundException 151  
 non-kindred pointcut 77–85  
 definition of 75  
 Spring, and 77  
 non-modularization, performing  
 code reviews 25  
 nontransactional resource  
 notifications 448  
 no-rollback-for, example of 392  
 null return, policy  
 enforcement 303

## O

object  
 empowering 432–447  
 immutable 452  
 managing access to 452–457  
 object pre-initialization join  
 point 60  
 pointcut syntax 76  
 object-initialization join  
 point 60  
 comparison with class  
 initialization 60  
 pointcut syntax 76  
 objective characteristic-based  
 annotation 340  
 object-level constraint  
 using an aspect 449  
 object-relational mapping 472  
 bulk update, and 305  
 dependency injection,  
 and 434  
 domain object dependency,  
 and 433  
 mixing with JDBC,  
 policies 305  
 observer design pattern 15, 19  
 AOP alternatives 19  
 context collection 19  
 observer 19  
 subject 19  
 with aspect 117  
 OC4J, support for Spring-driven  
 LTW 244  
 OneWay annotation 401  
 OOP, program flow  
 abstraction 24  
 Open/Close principle 5  
 operator  
 ! 65  
 && 65  
 || 65  
 binary 65  
 unary 65  
 operator precedence 65  
 OR operator 210  
 Order annotation, use of 402  
 Order class, source code 472  
 Ordered interface 233  
 use of 402  
 Ordered, use of 418  
 OrderService, source code 477  
 org.aspectj.lang package 106  
 org.aspectj.lang.Aspects. *See*  
 Aspects  
 org.aspectj.lang.JoinPoint  
 interface 108  
 org.aspectj.lang.JoinPoint. *See*  
 JoinPoint  
 org.aspectj.lang.JoinPoint.Static-  
 Part interface 108  
 org.aspectj.lang.JoinPoint.Static-  
 Part. *See* JoinPoint.StaticPart  
 org.aspectj.lang.reflect  
 package 107  
 org/aspectj/aop.xml  
*See also* aop.xml  
 ORM  
 validation 450  
*See also* object-relational map-  
 ping

- OSGi 208
  - outjar 489
    - ajc option 204
    - ant, example of 487
  - outxml 489
  - outxml option 213
  - outxmlfile option 213
  - overdesign, avoiding with
    - aspects 464
- P**
- 
- package arrangement, domain
    - object dependency injection 443
  - package structure, monitoring,
    - join point selection 259–261
  - Palo Alto Research Center 15
  - parameter
    - generic, and aspects 140
    - logging 268–270
  - PARC. *See* Palo Alto Research Center
  - participant pattern 330–336
    - augmenting behavior 335
    - consequences 336
    - current solutions 331–333
    - overview 333
    - scoping 335
    - Swing thread safety, and 353
    - template 333–336
    - UML class diagram 335
  - patching, aspects, and 463
  - perflow aspect association,
    - example of 400–401
  - perflow() 146
    - example of 145, 147
    - implicit limiting of join points 151
    - sequence diagram 146
    - Spring AOP, lack of 235
  - perflowbelow()
    - implicit limiting of join points 151
    - Spring AOP, lack of 235
    - See also* perflow
  - per-control-flow
    - association 145–148
    - See also* perflow()
    - See also* perflowbelow()
  - performance
    - Spring AOP vs. AspectJ 246
    - woven code 44
  - performance guarantee,
    - monitoring 284
  - performance monitoring 281–284
    - deployed system adoption 461
  - performance-monitoring aspect 281, 461
  - per-object aspect association,
    - read-write lock pattern, example of 367
  - per-object association 143–145
    - example of 143
    - member introduction, relation to 152
    - sequence diagram 144
    - static-crosscutting, relation to 152
    - See also* pertarget()
    - See also* perthis()
  - personal productivity, improving by using policy enforcement 318
  - pertarget()
    - implicit limiting of join points 151
    - Spring AOP 234
    - See also* perthis
  - perthis() 143
    - example of 144, 152
    - implicit limiting of join points 151
    - Spring AOP 234
  - perthis() aspect association
    - read-write lock pattern, example of 367
    - static methods, and 369
  - per-type association 148–150
    - See also* pertypewithin()
  - pertypewithin() 270
    - example of 148–149, 270
    - getWithinTypeName, and 148
    - implicit limiting of join points 151
    - Spring AOP, lack of 235
    - staticinitialization pointcut, and 149
  - plain old Java objects 217
  - PlatformTransactionManager 377
    - use of 392
  - plug-and-play, EJB programming restrictions 309
  - PMD 171, 289, 297
  - pointcut
    - @annotation 84
    - @args 84
    - @target 84
    - @this 84
    - @within 84
    - @withincode 84
    - abstract 174
      - example of 351
    - access specification 64
      - example of 32
    - adviceexecution() 76
    - annotation-based 84
      - example of 456
    - annotation-based, CSS, analogy 14
    - anonymous 64
    - args() 83
    - argument 83
    - basics 64–65
    - call() 76
    - capturing based on return type, example 355
    - capturing no join point 64
    - cflow() 78
    - cflowbelow() 78
    - characteristics 54–55
    - class initialization 76
    - collecting context 99, 101
    - concept 12
    - concrete 175
    - conditional check 85
    - constructor call 76
    - constructor execution 76
    - context collection 54
    - control-flow based 77
    - CSS selector, analogy 13
    - defined in a class, example of 332
    - definition of 32
    - enumeration 331
    - example of 30–31, 143
    - exception handler 76
    - execution object 80
    - execution() 76
    - expression
      - @AspectJ syntax 175–176
      - @AspectJ, and 170
    - field-read access 76
    - field-write access 76
    - fragile definition, example of 332
    - get() 76
    - handler() 76
    - if() 85

- pointcut (*continued*)
    - implementing 75–85
    - initialization() 76
    - kinded. *See* kinded pointcut
    - lexical-structure-based 79
    - mapping 174–179
      - abstract pointcut 174
      - concrete pointcut 175
    - method call 76
    - method execution 76
    - named 64
      - general form 64
    - naming 32
    - non-kinded. *See* non-kinded pointcut
    - object initialization 76
    - object pre-initialization 76
    - operators 65
    - preinitialization() 76
    - runtime information, use of 54
    - selecting subclass methods, example of 34
    - selection criterion 54
    - selection, using annotation 84
    - signature 54
    - signature syntax 65–75
    - Spring AOP 230–232
    - static initialization 76
    - statically determinable
      - declare error and warning, use in 132
      - policy enforcement, and 295
    - statically determining 54
    - staticinitialization() 76
    - target() 81–82
    - this() 81–82
    - use in advice 90
    - using from another aspect, example of 353
    - vs. reflective API 113
    - weaver mapping 42
    - within() 80
    - withincode() 80
    - write() 76
    - writing 52
  - pointcut attribute, Spring AOP 238
  - pointcut expression, Spring AOP 44
  - pointcut language
    - dynamic proxy, lack of 224
    - Spring AOP 225
  - pointcut library aspect, example of 291–292
  - Pointcut. *See* @Pointcut
  - pointcut-ref attribute, Spring AOP 238
  - POJO 217
  - policy
    - destinations 294
    - origins 294
  - policy enforcement
    - access control 452
    - application frameworks, and 293
    - build system, and 295
    - code-analysis tools, and 297
    - compile-time 295
    - core program behavior 297
    - Data Access Object, and 303
    - definition of 289
    - deployment consideration 297
    - developmental aspect 289
    - documentation, as 294
    - EJB 308–313
    - EJB programming restrictions 308
    - JPA policies 304, 308
    - layered architecture, and 289
    - layering enforcement example 290
    - library development 293
    - no EJB policy 312
    - overcoming adaptation resistance 318
    - overview 289–294
    - patterns 299
    - restricting exposure 298
    - return-value restriction 303
    - runtime enforcement 296
    - schematic 293
    - Swing 313–317
    - tests, implementation 348
    - using AOP 294–297
  - policy enforcement pattern, flexible access control 298–302
  - policy violation, consequences 294
  - policy-enforcement aspect, AOP adoption 461
  - post-compilation step @AspectJ syntax, and 48 IDE, with 48
  - PostSharp 17
  - pre-built aspect, Spring AOP 460
  - precedence
    - advice 159
    - aspect 159
    - control 162–164
      - aspect inheritance, and 164
      - wildcards, use of 163
    - control patterns 163–164
    - rules for advice 161
  - precedence control, circular dependency 164
  - precedence, declaring XML syntax, example of 399
  - preinitialization() pointcut 76
  - prescreening for security 427
  - principal, security concept 408
  - Principle of Least Knowledge 441
  - privileged aspect 166
    - avoiding overuse 167
    - example of 167
  - proceed() method 187–188
    - @AspectJ syntax 187
    - altered context 188
    - weaknesses 189
  - altered context rules 188
  - arguments to 94
  - bypassing join point 94
  - example of 33, 120, 122, 154
  - InvocationHandler, comparison with 223
  - return value of 94
  - rules 188
  - use of 90
  - worker object and 321
    - example of 356
- ProceedingJoinPoint 187
  - example of 282, 284, 380, 396
  - Spring AOP, and 233
- ProceedingJoinPoint.proceed 188
  - exception processing 188
- Product class, source code 470
- production aspect 462
- production, monitoring, use of 251
- profiling, example of 33
- program, running 29
- programming practices, policy enforcement 289
- propagation attribute 375
- property, change notifications 117

prototype scope 436  
 proxy  
   creation, Spring AOP 225  
   dynamic 20  
 Proxy class, Spring AOP, use  
   in 223  
 proxy design pattern  
   AOP alternatives 20  
   Spring AOP, use of 21  
 proxy-based  
   transaction management 378  
   illustration 378  
   limitations 379  
 proxy-based AOP 37, 226–227  
 proxy-target-class  
   attribute 229

## Q

---

QA, runtime policy enforcement, and 296

## R

---

Rails 22  
 read-write lock pattern 143,  
   365–371  
   aspect association, need  
   for 143  
 AspectJ implementation  
   367–371  
   conventional  
   implementation 365–367  
 ReadWriteLock 366  
 ReentrantReadWriteLock 366  
 refactoring  
   aspects 260  
   process 260  
   using aspects 450  
 refactoring aspect 462  
 reflection API 105–113  
   class diagram 106  
   dynamic information 105  
   example of 109  
   Signature interface 107  
   SourceLocation interface 107  
   static information 105  
   static vs. dynamic  
   information 105  
   structural relationship 107  
   UML class diagram 106–107  
 reflection API, Signature  
   interface 109

ReflectionTestUtils, use in  
   testing 437  
 reflective API  
   using 109  
   vs. pointcuts 113  
   *See also* reflection API 106  
 remote client, service layer,  
   and 445  
 Remote Method Invocation 263  
 repository  
   generic, implementation  
   of 475  
   generic, interface of 475  
   injecting 445  
 repository layer, transaction  
   management 376  
 resource management, Thread-  
   Local, using aspect 327  
 responsiveness, improving  
   362–365  
 retention policy 84  
   context collection 101  
 retry aspect, base 395  
 retry logic, use of 449  
 retry transactional operation,  
   illustration of 397  
 RetryCallback, example of 396  
 RetryTemplate 396  
 return value, restricting, policy  
   enforcement 303  
 rich behavior 432  
   implementing 433–434  
 RMI. *See* Remote Method Invoca-  
   tion  
 role mapping, externalizing 417  
 rollback-for, example of 392  
 routing, asynchronous, worker  
   object pattern, and 364  
 Ruby 17, 22  
   AOP implementation 17  
   mixin, use of 121  
 rule engine, use with  
   aspects 117  
 rule of thumb, crosscutting  
   concern 464  
 Runnable, worker object,  
   and 320  
 RunnableWithReturn 324  
   use of 325, 355  
 runtime policy  
   enforcement 296  
 runtime policy violation,  
   logging 296

## S

---

Scala 16  
   mixin, use of 121  
   plug-in, Equinox Aspects, use  
   of 208  
 scattering  
   CSS 14  
   static crosscutting, and 117  
   structural 117  
 schema-style Spring AOP  
   218, 236–240  
   mapping advice 238  
   mapping aspects 236  
   mapping pointcuts 237  
   mapping static  
   crosscutting 239  
   singleton instantiation  
   model 237  
 scoping, participant pattern 335  
 Seasar 17  
 security  
   auditing, and 429  
   conventional  
   implementation 405  
   domain object, and 428  
   EJB framework 406  
   EJB, implementation of 21  
   filtering field-content 429  
   framework approach 406  
   modularizing, using AOP 406  
   prescreening for 427  
   principal, concept of 408  
   simple example 29–31  
   single sign-on 407  
   solution, implementing  
   409–421  
   using aspects, overview 406  
   using OOP 7  
 security aspect, example of 30  
 security attribute,  
   externalizing 417  
 security namespace 425  
 <security:global-method-  
   security> 426  
 <security:http> 425  
 <security:intercept-url> 425  
 <security:protect-pointcut> 426  
 SecurityAttribute 405  
 SecurityContext 408, 412  
 SecurityContextHolder 408  
 self-call limitation, example  
   of 390

- separation of concerns
  - CSS 13
  - design/implementation mismatch 6
  - multidimensional space 6
- sequence diagram, tracing, and 264
- service layer
  - and co-located web layer 445
  - as façade 445
  - domain object dependency injection, and 445
  - transaction management 376, 378
- service locator 433
- service object, Spring AOP 247
- service-level agreement, monitoring 284
- service-oriented architecture 445
- servlet filter
  - chain of responsibility design pattern 20
  - security, limitations of 406
- servlet, security, and 406
- servlets framework, AOP alternatives 18
- set pointcut, example of 309
- setter method
  - pointcut 330
  - selecting with 262
- showMessageDialog() method, Swing thread safety, example of 346
- showWeaveInfo 202
- signature
  - signature. *See* pointcut
  - syntax 65–75
- signature pattern
  - constructor. *See* constructor signature pattern
  - field. *See* field signature method. *See* method signature pattern
  - type. *See* type signature
  - use in static crosscutting 131
- Simon 282
- Single Responsibility Principle 5
- single sign-on 407, 411, 413
- slf4j 255
  - support for NDC and MDC 255
- SOA. *See* service-oriented architecture
- socket creation, EJB programming restrictions 312
- SoftException 195
- source 5 flag 33
- source code weaving
  - 39, 200–201
  - illustration 201
  - internal mechanism 202
- source, compiling 29
- SourceLocation 109
- sourceroots 202
- SpelExpressionEvaluator 111
- Spring
  - @AspectJ integration 221
  - @AspectJ syntax, use of 37
  - @AspectJ, example of 221
  - adoption 218
  - complexity, vs. AspectJ 245
  - role in 460
  - and AspectJ weaver 240–245
  - and dependency injection 434
  - and non-kinded pointcuts 77
  - application context, example of 220
  - aspect, configuring with 154
  - AspectJ integration 44–46
  - aspectOf(), using 154
  - aspects
    - configuring with 240
    - configuring with DI 240
  - autowiring 435
  - bean, selecting 231
  - ClassPathXmlApplicationContext 155
  - configuration, example of 45
  - configuring aspects 154, 218
  - container services, domain objects, in 438
  - exposing aspect bean 154
  - expression language 109
  - injecting dependencies into aspect 154
  - load-time weaving 213
  - method-execution join point 56
  - native load-time weaving 206
  - native LTW 42
  - tracing aspect, example of 273
  - transaction management abstraction 377
- Spring AOP 16
  - @annotation pointcut 231
  - @args pointcut 231
  - @AspectJ integration, illustration 229
  - @DeclareParents 235
  - @target pointcut 231
  - @this pointcut, lack of 231
  - @within pointcut 231
  - and @DeclarePrecedence 233
  - and ProceedingJoinPoint 233
  - <aop:advisor> 237
  - <aop:after-returning> 238
  - <aop:after-throwing> 238
  - <aop:aspect> 237
  - <aop:before> 238
  - <aop:after> 238
  - <aop:config> 237
  - <aop:pointcut> 237
  - arg-names attribute 238
  - args pointcut 230
  - aspect ordering 233
  - AspectJ weaving, vs., guidelines 245–247
  - autoproxy mechanism 227
  - bean-only limitation 227
  - CGLIB proxy 226
  - compared to AspectJ 245
  - configurability, vs. AspectJ 246
  - dynamic crosscutting 230–235
  - execution pointcut 230
  - external call-only limitation 228
  - illustration 228
  - fundamentals 218–222
  - illustration 225–226
  - internals 225
  - join point context 225
  - join point model, vs. AspectJ 245
  - lack of compile-time errors and warnings 229
  - lack of percfLOW 235
  - lack of percfLOWbelow 235
  - lack of perctypewithin 235
  - limitations 227–228, 460
  - method attribute 238
  - method execution-only limitation 227
  - workaround 228
  - performance monitoring 283
  - performance, vs. AspectJ 246
  - pertarget 234
  - perthis 234
  - pointcut 230–232
  - pointcut attribute 238

- Spring AOP (*continued*)
  - pointcut language 225
  - pointcut-ref attribute 238
  - policy enforcement, and 296
  - pre-built aspects 460
  - proxies 224
    - use of 223
  - proxy-based 226–227
  - schema-style 236–240
    - illustration 236
    - mapping advice 238
    - mapping aspects 236
    - mapping pointcuts 237
    - mapping static
      - crosscutting 239
    - singleton instantiation
      - model 237
  - selecting a horizontal slice of
    - beans 231
  - selecting a vertical slice of
    - beans 231
  - static crosscutting 235
  - subset of `@AspectJ` syntax 229
  - syntax comparison 246
  - this pointcut 230
  - tracing 273–275
  - under the hood 222–228
  - vs. `AspectJ`, decision
    - process 246
- Spring Batch 396
- Spring bean 44
- Spring container 44
- Spring Framework
  - `@AspectJ` syntax, and 169, 197
  - `@AspectJ`, integration 229–235
  - core ideas 217
- Spring IDE 222
  - AJDT, and 222
  - illustration 222
- Spring MVC 469
- Spring OP, proxy creation 225
- Spring Security
  - authentication 407
  - authorization 408
  - introduction 405
  - overview 407–409
  - prebuilt solutions 425–428
  - service-level security 425
    - annotation option 426
    - XML option 426
  - URL level security 425
  - web security 425
- Spring.NET
  - AOP 16
- Spring-driven load-time
  - weaving 240–245, 272
  - configuration 242
  - GlassFish 242
  - illustration 244
  - Tomcat 242
  - tracing, and 272
  - Under the hood 243
- SpringJUnit4ClassRunner 436
  - use of 436
- SpringPointcuts, implementa-
  - tion of 292
- SpringSource Application Moni-
  - toring Suite 281
- spring-tomcat-weaver.jar 242
- SQL 14
- SRP. *See* Single Responsibility
  - Principle
- Standard Widget Toolkit, policy
  - enforcement 314
- state, aspect association 140
- static context 105
- static crosscutting
  - compile-time errors and
    - warnings 132
  - concept 12
  - example of 132
  - humane interface, and 446
  - member introduction
    - 117–128
  - mix-in 121–123
  - modifying the type
    - hierarchy 128–130
  - per-object association, relation
    - to 152
  - simplified API, and 446
  - Spring AOP 235
  - supplying annotations
    - 130–132
  - vs. dynamic crosscutting,
    - evaluation order 128
- static field, EJB programming
  - restrictions 309–310
- static initializer, `pertypewithin()`,
  - and 149
- static method
  - `getTarget()` 108
  - `getThis()` 108
- staticinitialization pointcut
  - example of 148, 313
  - `pertypewithin()`, and 149
- stereotype, UML 390
- strongly typed languages 295
- stylistic violation, policy enforce-
  - ment, and 295
- subaspects 138
  - annotation-driven participant
    - pattern, and 337
  - example of 138, 143, 267,
    - 353, 369, 412, 416
  - inheritance limitation 139
- subject, observer design
  - pattern 19
- subjective characteristic-based
  - annotation 339
- Swing
  - aspect library 316
  - library aspect
    - use of 347, 351
  - policy enforcement 313–317
  - single-thread rule,
    - modularizing 345–362
  - thread safety
    - `AspectJ` solution 351–360
    - conventional solution
      - 348–350
    - test problem 346–348
  - `SwingThreadSafetyAspect` 351
  - `SwingWorker` 350
  - SWT. *See* Standard Widget
    - Toolkit
  - synchronous call, example
    - of 346
  - syntax
    - `@AspectJ` 31
    - traditional 31
  - syntax choices, adoption, role
    - in 460
  - syntax tree, abstract 19
- System.in access, EJB program-
  - ming restrictions 312
- SystemArchitecture
  - implementation of 290
  - use of 290
- SystemArchitecture aspect 260

---

**T**

- tangling
  - CSS 14
  - static crosscutting, and 117
  - structural 117
- `target()` pointcut 81, 101
  - `@AspectJ` syntax, example
    - of 183
  - context collection 99
  - example of 82
  - reflection, and 113

- target() pointcut (*continued*)
  - static methods 82
  - use of wildcard 81
- taskdef, iajc task, defining 486
- team, policies, and 294
- testing
  - aspect 348, 457
    - example of 118–121, 383, 399
  - transaction-management aspect 382–384
- testing-helper aspect 461
- this 82
- this pointcut
  - @AspectJ syntax, example of 193
  - example of 35, 119, 141, 329
  - Spring AOP 230
- this() pointcut 81, 101
  - context collection 99
  - difference from call() pointcut 82
  - difference from within() 82
  - example of 82
  - reflection, and 113
  - static methods 82
  - use of wildcard 81
- thisEnclosingJoinPointStaticPart 105–106
  - @AspectJ syntax, in 182
  - example of 106, 280
- thisJoinPoint 105–106, 108
  - @AspectJ syntax, in 182
  - example of 62, 94
  - string representation 63
  - tracing, use in 268
  - usage of 106
  - use of 90
- thisJoinPointStaticPart 105–106, 108, 254
  - @AspectJ syntax, in 182
  - example of 33, 120, 142, 275, 347
  - usage of 106
- thread
  - AWT 313
  - event-dispatching 313, 346
- thread pool, size, monitoring, and 281
- thread safety
  - AspectJ solution 351–360
  - avoiding the overhead 362
  - conventional solution 348–350
  - dealing with exceptions 361
  - read-write lock pattern. *See* read-write lock pattern
  - test problem 346–348
- Thread.currentThread() 347
- threading, EJB programming restrictions 308
- ThreadLocal
  - clearing, using aspect 327
  - exception logging, and 275
  - percfow, and 146
  - wormhole pattern, and 327
- thread-local
  - authentication, use of 408
  - call-depth monitoring 264
  - transactional resources 379
- throttling, monitoring, and 281
- tipping point, AspectJ usage 460
- toLongString() method 254
- Tomcat
  - load-time weaving 214
  - Spring-driven LTW 242
  - support for Spring-driven LTW 244
- TomcatInstrumentableClassLoader 242
- TopLink, transaction management 378
- toShortString() method 254
- toString() method, excluding using pointcut 269
- trace aspect
  - indentation, with 265
  - shared functionality 266
- tracing 252–256, 264–271
  - AOP vs. conventional 257–258
  - AspectJ, graphical illustration 258
  - aspect-oriented 253–254
  - consistency requirement 257
  - conventional 255–256
  - illustration 257
  - problems 256
  - dynamic proxy, using 223
  - indentation effect 62, 264–266
  - intra-method activities 267
  - load-time weaving, use of 271–273
  - logging, vs. 257
  - method parameters 268–270
  - transaction activities 385
  - type-specific logger, use of 270–271
  - variations 252
- traditional syntax 31
  - choosing, vs. @AspectJ 215
- transaction attribute, concept 375
- transaction management
  - abstract aspect 380
  - abstraction, Spring 377
  - annotation driven 393
  - AOP implementation 375
  - AspectJ weaver based 394
  - AspectJ-based, Spring, and 391
  - conventional implementation 374
  - dependency injection, and 381
  - distributed 377
  - domain objects, and 444
  - EJB, implementation of 21
  - entity manager, proxy 388
  - exception handling 381
  - fault tolerance, and 395–402
  - Hibernate 377
  - idempotent, and 395
  - implementation 377
  - implementation choices 378–379
    - byte-code weaving 379
    - proxy-based 378
  - implementations 374–376
  - implementing 52
  - JDBC 377
  - JMS 377
  - JPA 377
  - layered architecture, and 376
  - need for 373
  - players 376–378
  - properties 373
  - proxy-based illustration 378
  - Spring, and 391
  - resource management, and 375
  - Spring support 390–395
  - TopLink 378
  - XML driven 391–393
- transaction management aspect, annotation-based 381
- transaction manager 375
- transaction retry, considerations 397
- transaction, activities, tracing 385
- TransactionAttribute 393
- TransactionDefinition 377

transaction-management aspect  
 advice 380  
 attribute, obtaining 380  
 constituents 380–382  
 pointcut definition 380  
 testing 382–384  
 XML vs. annotations 380  
 TransactionPointcuts 386  
 TransactionStatus 377  
 transient, domain object dependency, and 433  
 trigger 14  
 try/finally 266  
   nested diagnostic context, with 266  
   read-write lock pattern, and 367  
 two-phase commit 402  
 tx:advice, example of 391, 482  
 tx:annotation-driven  
   AspectJ weaving 394  
   example of 393  
 tx:attributes 393  
   example of 391  
 tx:method, example of 391  
 type hierarchy, modifying 128  
 type signature  
   annotation, examples 68  
   example use in method signature 69–70  
   examples 66  
   package declaration 66  
   patterns 66–69  
   subtype specification 66  
   using binary operator 69  
   using unary operator 69  
 type signature pattern  
   annotation-based 67  
   basic 66  
   combining 69  
   generic-based 68  
 type structure, monitoring, join point selection 261  
 type, definition of 66  
 typecast, dynamic proxy, need for 224  
 type-level annotation 387

## U

UI application, improving responsiveness. *See* responsiveness  
 UI controller, Spring AOP 247

UML 390  
   reflection API 106–107  
 UnanimousBased 409  
   use of 418  
 unary operator 65  
   type signature 69  
 unit test, dependency injection, and 219  
 untangling, with aspects 118–121  
 up-front login, authentication, and 410  
 URL level security, Spring Security 425

## V

validation  
   ORM frameworks 450  
   using an aspect 449  
 visitor design pattern 319  
 VM. *See* Java virtual machine

## W

wait cursor, management, annotation, with 339  
 Wampler, Dean 22  
 #warning, weave-time error and warning, comparison with 132  
 warning, weave-time, weaver mapping 42  
 weaveDependencies  
   Maven, and 495  
 <weaver> 211–212  
 weaver  
   AspectJ 39–42  
   definition of 10  
   mapping 42–44  
   source-code input 201  
   using byte-code transformation 10  
   using source-to-source translation 10  
 weave-time declaration  
   concept 12  
   definition of 36  
 weave-time error, weaver mapping 42  
 weave-time warning, weaver mapping 42

weaving  
   binary 200  
     build-time 202–205  
     input  
       compilation of 203  
       -g flag 203  
       -gvars flag 203  
   build-time 200–205  
   source code,  
     illustration 201  
   byte-code. *See* binary weaving  
   comparison with event-based programming 15  
   concept 4  
   dynamic 10  
   load-time 200  
   source code 200–201  
     internal mechanism 202  
   weaving mechanism 39–42  
     syntax, orthogonality 39  
 weaving model  
   adoption, role in 460  
   classification 200  
   combining 200  
 weaving rule  
   annotations, use of 10  
   definition of 9  
   economical expression 9  
   expression language 10  
   specificity 10  
 weaving source, Ant, example of 486  
 weaving sources, Maven, example of 491  
 web application,  
   monitoring 281  
 web layer  
   co-located 452  
   and service layer 445  
 web security, Spring Security 425  
 WebLogic, support for Spring-driven LTW 244  
 wildcard  
   .. 66  
   ..., in method signatures 70  
   \* 66  
   + 66  
   bean pointcut 231  
   need for 66  
 within() pointcut 80  
   @AspectJ syntax, example of 181  
   avoiding infinite recursion 62

- within() pointcut (*continued*)
    - difference from this() 82
    - example of 61, 80, 260–261
    - restricting aspect
      - application 463
    - restricting scope, example of 351
    - usage of 80
    - use in access control 300
    - use to avoid infinite recursion 275
  - withincode() pointcut 80
    - example of 80, 302
    - use of 267
  - worker method 320
  - worker object
    - continuation, and 324
    - current solution 320
    - definition of 320
    - retry aspect, use of 396
  - worker object pattern 320–327
    - example of 352, 364
    - getting the return value 324–326
  - overview 321
  - passing worker to thread 323
  - routing methods with return value 324–326
  - Swing thread safety, use in 351
  - template 321–324
  - wormhole pattern 327–330
    - calleeSpace() pointcut 328
    - callerSpace() pointcut 328
    - current solutions 327
    - example of 329
    - illustration 328
    - overview 327
    - pattern template 328–330
    - use of 456
    - wormhole() pointcut 328
  - woven code, Java byte-code specification 39
  - write() pointcut 76
  - write-behind strategy, ORM frameworks, and 305
- X**
- 
- Xerox Corporation 15
  - Xlint 202
  - Xlint:adviceDidNotMatch warning 488
  - Xlint:warning 163
  - xlint.properties 488
  - xlintfile 488
  - XML syntax
    - aspect precedence 210
    - fully qualified type names 210
    - limitations 210
    - weaving options 211
  - XP 450
    - AspectJ, using 450
    - relation to AOP 25
  - XPI. *See* Crosscut Programming Interface
- Y**
- 
- YAGNI 25