



# AspectJ

---

# IN ACTION

Practical Aspect-Oriented Programming

Ramnivas Laddad

 MANNING

# contents

---

*preface* xvii  
*how real is AspectJ?* xix  
*into the future!* xxi  
*acknowledgments* xxiii  
*about this book* xxv

---

## PART 1 UNDERSTANDING AOP AND ASPECTJ ..... 1

---

### **1** *Introduction to AOP* 3

- 1.1 The architect's dilemma 5
- 1.2 Evolution of programming methodologies 6
- 1.3 Managing system concerns 7
  - Identifying system concerns* 8
  - *A one-dimensional solution* 10
  - *It's all about modularizing* 11
- 1.4 Implementing crosscutting concerns in nonmodularized systems 14
  - Symptoms of nonmodularization* 15
  - *Implications of nonmodularization* 18
  - *Introducing AOP* 19
  - A bit of history* 20
  - *The AOP methodology* 21

- 1.5 Anatomy of an AOP language 22
  - The AOP language specification* 23
  - *The AOP language implementation* 24
  - *A weaving example* 26
- 1.6 Benefits of AOP 27
- 1.7 Myths and realities of AOP 29
- 1.8 Summary 30

## 2 **Introducing AspectJ** 32

- 2.1 AspectJ: a bird's eye view 33
  - Crosscutting in AspectJ* 33
  - *Crosscutting elements* 34
- 2.2 AspectJ Hello World 37
- 2.3 AspectJ: under the hood 40
- 2.4 The join point model 43
  - Exposed join point categories* 44
  - *Join point demonstration example* 50
- 2.5 Aspects 55
- 2.6 AspectJ logistics overview 59
  - The AspectJ compiler* 59
  - *AspectJ browser* 60
  - IDE integration* 61
- 2.7 Summary 62

## 3 **AspectJ: syntax basics** 64

- 3.1 Pointcuts 65
  - Wildcards and pointcut operators* 67
  - *Signature syntax* 68
  - *Implementing pointcuts* 73
- 3.2 Advice 81
  - Anatomy of advice* 82
  - *The before advice* 83
  - *The after advice* 83
  - *The around advice* 85
  - *Comparing advice with methods* 86
  - *Passing context from a join point to advice* 87
  - Returning a value from around advice* 89
  - *An example using around advice: failure handling* 90
  - *Context collection example: caching* 92
- 3.3 Static crosscutting 95
  - Member introduction* 95
  - *Modifying the class hierarchy* 96
  - Introducing compile-time errors and warning* 97
- 3.4 Tips and tricks 98
- 3.5 Summary 99

- 4 *Advanced AspectJ* 100**
- 4.1 Accessing join point information via reflection 101
    - The reflective API* 103 ▪ *Using reflective APIs* 106
  - 4.2 Aspect precedence 111
    - Ordering of advice* 114 ▪ *Explicit aspect precedence* 115
    - Aspect inheritance and precedence* 117 ▪ *Ordering of advice in a single aspect* 119 ▪ *Aspect precedence and member introduction* 120
  - 4.3 Aspect association 122
    - Default association* 123 ▪ *Per-object association* 125
    - Per-control-flow association* 128 ▪ *Implicit limiting of join points* 132 ▪ *Comparing object association with member introduction* 134 ▪ *Accessing aspect instances* 135
  - 4.4 Exception softening 136
  - 4.5 Privileged aspects 139
  - 4.6 Summary 141

---

**PART 2 BASIC APPLICATIONS OF ASPECTJ ..... 143**

- 5 *Monitoring techniques: logging, tracing, and profiling* 145**
- 5.1 Why use AspectJ for logging? 146
    - A simple case in point* 147 ▪ *Logging the conventional way* 149 ▪ *Logging the aspect-oriented way* 153
  - 5.2 What's wrong with conventional logging 154
  - 5.3 The beauty of AspectJ-based logging 156
  - 5.4 Developing logging and tracing aspects 156
    - Method call tracing* 157 ▪ *Exceptions logging* 163
  - 5.5 Common logging idioms 167
    - Logging the method parameters* 168 ▪ *Indenting the log statements* 170 ▪ *Aspect precedence* 172 ▪ *Changing the underlying logging mechanism* 173 ▪ *Using logging in a multithreaded environment* 173
  - 5.6 Extending logging for other usage 174
    - Testing* 174 ▪ *Profiling* 175
  - 5.7 Summary 176

- 6 Policy enforcement: system wide contracts 178**
- 6.1 AspectJ-based policy enforcement overview 179
  - 6.2 The current solution and its challenges 181
  - 6.3 Enforcement using AspectJ 182
    - Policy enforcement implementation choices 183* ▪ *The role of policy enforcement during the product lifecycle 184*
  - 6.4 Policy enforcement patterns 185
    - Detecting the violation of a specific call pattern 185*
    - Implementing flexible access control 187* ▪ *Enforcing the best-practices principles 189*
  - 6.5 Example: implementing EJB programming restrictions 191
    - Implementing “no AWT” 193* ▪ *Implementing “no nonfinal static field access” 194*
  - 6.6 Example: implementing Swing policies 195
    - Understanding the problem 196* ▪ *Detecting the violation 198*
  - 6.7 Summary 200
- 7 Optimization: pooling and caching 202**
- 7.1 The typical case 203
    - Return, reuse, recycle: The role of resource pooling 205* ▪ *Resource pooling issues 206*
  - 7.2 Diving into the pool using AspectJ 208
    - Designing a template aspect 208* ▪ *Implementing the template aspect 209*
  - 7.3 Example 1: database connection pooling 211
    - Understanding the database connection pool interface 212*
    - AspectJ-based database connection pooling 213*
    - Implementing the connection pool 216* ▪ *Testing our solution 218* ▪ *Tweaking the solution 222*
  - 7.4 Example 2: thread pooling 223
    - The echo server 224* ▪ *Understanding the thread pool interface 226* ▪ *AspectJ-based thread pooling 226*
    - Implementing the thread pool 230* ▪ *Testing our solution 231* ▪ *Tweaking the solution 234*

- 7.5 Extending pooling concepts to caching 235
  - AspectJ-based caching: the first version* 237
  - *AspectJ-based caching: the second version* 239
  - *Ideas for further improvements* 240
- 7.6 Summary 241

---

## PART 3 ADVANCED APPLICATIONS OF ASPECTJ ..... 243

### 8 *Design patterns and idioms* 245

- 8.1 The worker object creation pattern 247
  - The current solution* 248
  - *An overview of the worker object creation pattern* 249
  - *The pattern template* 249
  - A summary of the worker object creation pattern* 256
- 8.2 The wormhole pattern 256
  - The current solution* 257
  - *An overview of the wormhole pattern* 257
  - *The pattern template* 258
  - *A summary of the wormhole pattern* 260
- 8.3 The exception introduction pattern 260
  - The current solution* 261
  - *An overview of the exception introduction pattern* 265
  - *The pattern template* 265
  - A summary of the exception introduction pattern* 269
- 8.4 The participant pattern 270
  - Current solutions* 271
  - *An overview of the participant pattern* 273
  - *The pattern template* 274
  - *A summary of the participant pattern* 276
- 8.5 Idioms 277
  - Avoiding infinite recursion* 277
  - *Nullifying advice* 279
  - Providing empty pointcut definitions* 280
  - *Providing a default interface implementation* 281
- 8.6 Summary 285

### 9 *Implementing thread safety* 286

- 9.1 Swing's single-thread rule 287
  - The rule* 288
  - *The problem* 288
  - *The solution* 289
- 9.2 A test problem 290
- 9.3 Solution: the conventional way 293

- 9.4 Solution: the AspectJ way 297
  - The first version* 298 ▪ *The second version* 303
  - The third version* 307
- 9.5 Improving the solution 311
  - Dealing with exceptions* 311 ▪ *Avoiding the overhead* 312
- 9.6 Improving the responsiveness of UI applications 313
- 9.7 Modularizing the read-write lock pattern 316
  - Implementation: the conventional way* 316
  - Implementation: the AspectJ way* 318
- 9.8 Summary 321

## 10 **Authentication and authorization 323**

- 10.1 Problem overview 324
- 10.2 A simple banking example 325
- 10.3 Authentication: the conventional way 329
  - Implementing the solution* 329 ▪ *Testing the solution* 331
- 10.4 Authentication: the AspectJ way 333
  - Developing the solution* 333 ▪ *Testing the solution* 336
- 10.5 Authorization: the conventional way 336
  - Understanding JAAS-based authorization* 337 ▪ *Developing the solution* 338 ▪ *Testing the solution* 342 ▪ *Issues with the conventional solution* 345
- 10.6 Authorization: the AspectJ way 346
  - Developing the solution* 346 ▪ *Testing the solution* 350
- 10.7 Fine-tuning the solution 353
  - Using multiple subaspects* 353 ▪ *Separating authentication and authorization* 354
- 10.8 Summary 354

## 11 **Transaction management 356**

- 11.1 Example: a banking system with persistence 358
  - Implementing the core concern* 358
  - Setting up the test scenario* 362
- 11.2 The conventional solution 364
  - Using the same connection object* 365 ▪ *Committing at the top level only* 367

- 11.3 Developing a simple AspectJ-based solution 368
  - Implementing the JDBC transaction aspect* 368
  - *Handling legacy system issues* 373
  - *Enabling transaction management for the banking system* 374
  - *Testing the solution* 375
- 11.4 Improving the solution 378
  - Using the participant pattern* 379
  - *Implementing the JDBC transaction aspect: the second version* 382
  - Testing the solution* 385
- 11.5 Using AspectJ with advanced transaction-management systems 387
- 11.6 Summary 390

## 12 **Implementing business rules** 391

- 12.1 Using business rules in enterprise applications 392
- 12.2 An overview of business rule implementation 393
- 12.3 Current mechanisms 393
- 12.4 Introducing a solution using AspectJ 394
  - The template* 394
- 12.5 Example: the banking system 396
  - Implementing the core business logic* 396
  - *Implementing the first business rule* 401
  - *Implementing the second business rule* 403
  - *Writing a test program* 406
- 12.6 Implementing business rules with a rule engine 411
  - An overview of the rule engine* 412
  - *Using a rule engine* 412
  - *Modularizing with AspectJ* 415
- 12.7 Example: a banking system with a rule engine 417
  - A brief overview of Jess (Java Expert System Shell)* 417
  - Specifying rules* 418
  - *Understanding the rule invocation aspect* 420
- 12.8 Summary 423

## 13 **The next step** 425

- 13.1 Applying AspectJ to new problems 426
  - Talking the talk* 426
  - *Walking the walk* 427



- 13.2 Employing AspectJ in development phases 427
  - AspectJ in the design phase* 428
  - *AspectJ in the implementation phase* 428
  - *AspectJ in the testing phase* 431
  - *AspectJ in the maintenance phase* 432
  - *AspectJ in legacy projects* 432
- 13.3 A word of warning 433
- 13.4 Evangelizing AspectJ 434
- 13.5 Parting thoughts 436

## **A** *The AspectJ compiler* 438

- A.1 Downloading and setting up 439
- A.2 An overview of the compiler 440
- A.3 Compiling source files 441
- A.4 Compiling source directories 441
- A.5 Weaving into JAR files 442
- A.6 Creating aspect libraries 443
- A.7 Using aspect libraries 444
- A.8 Utilizing incremental compilation mode 444
- A.9 Producing useful warnings 446

## **B** *Understanding Ant integration* 447

- B.1 Compiling source files using an Ant task 448
- B.2 Weaving into JAR files using an Ant task 451
- B.3 Creating aspect libraries using an Ant task 452
- B.4 Utilizing aspect libraries using an Ant task 453
- B.5 Utilizing incremental compilation using an Ant task 453

*resources* 455

*index* 461