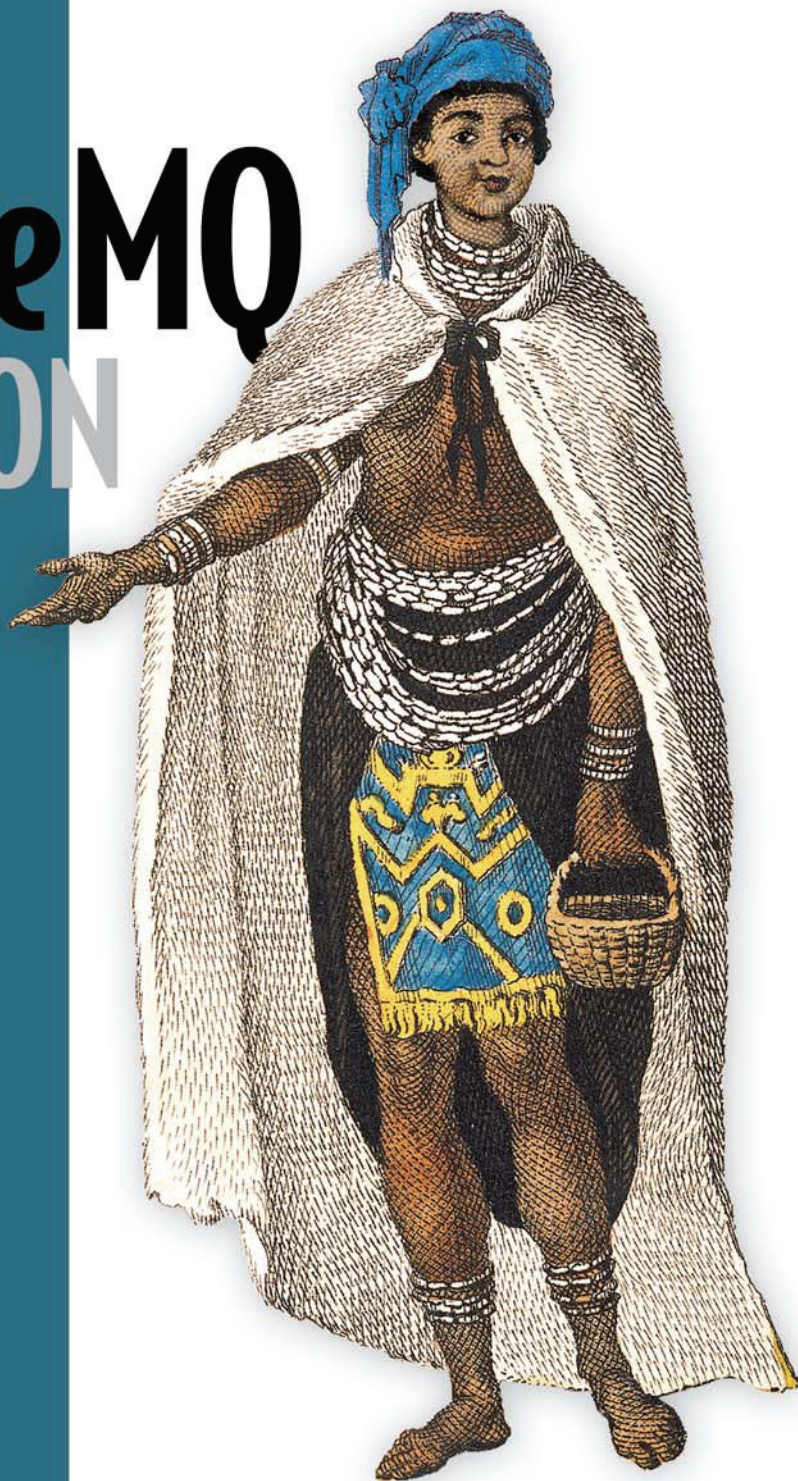


ActiveMQ

IN ACTION

Bruce Snyder
Dejan Bosanac
Rob Davies



contents

preface xv
acknowledgments xvii
about this book xix

PART 1 AN INTRODUCTION TO MESSAGING AND ACTIVEMQ1

- 1 *Introduction to Apache ActiveMQ* 3**
 - 1.1 ActiveMQ features 4
 - 1.2 Using ActiveMQ: why and when? 6
 - Loose coupling and ActiveMQ* 6 ■ *When to use ActiveMQ* 8
 - 1.3 Getting started with ActiveMQ 10
 - Downloading and installing the Java SE* 10 ■ *Downloading ActiveMQ* 11 ■ *Examining the ActiveMQ directory* 11
 - Starting up ActiveMQ* 12
 - 1.4 Running your first examples with ActiveMQ 14
 - 1.5 Summary 16

- 2 *Understanding message-oriented middleware and JMS* 17**
 - 2.1 Introduction to enterprise messaging 18
 - 2.2 What's message-oriented middleware? 20

- 2.3 What's the Java Message Service? 21
- 2.4 The JMS specification 23
 - JMS clients* 23 ▪ *Non-JMS clients* 25 ▪ *The JMS provider* 25
 - The JMS message* 25 ▪ *JMS message internals* 25 ▪ *Message selectors* 29 ▪ *JMS domains* 32 ▪ *Administered objects* 35
- 2.5 Using the JMS APIs to create JMS applications 35
 - A simple JMS application* 36 ▪ *Message-driven beans* 39
- 2.6 Summary 41

3 *The ActiveMQ in Action examples* 42

- 3.1 Downloading Maven and compiling the examples 43
- 3.2 Use case one: the stock portfolio example 45
 - Running the stock portfolio example* 46
- 3.3 Use case two: the job queue example 50
 - Running the job queue example* 51
- 3.4 Summary 53

PART 2 CONFIGURING STANDARD ACTIVEMQ COMPONENTS.....55

4 *Connecting to ActiveMQ* 57

- 4.1 Understanding connector URIs 58
- 4.2 Transport connectors 60
 - Configuring transport connectors* 60 ▪ *Adapting the stock portfolio example* 61
- 4.3 Connecting to ActiveMQ over the network 63
 - Transmission Control Protocol (TCP)* 64 ▪ *New I/O API protocol (NIO)* 66 ▪ *User Datagram Protocol (UDP)* 68
 - Secure Sockets Layer Protocol (SSL)* 70 ▪ *Hypertext Transfer Protocol (HTTP/HTTPS)* 77
- 4.4 Connecting to ActiveMQ inside the virtual machine (VM connector) 79
- 4.5 Network connectors 81
 - Static networks* 83 ▪ *Dynamic networks* 88
- 4.6 Summary 94

- ## 5 *ActiveMQ message storage* 96
- 5.1 How are messages stored by ActiveMQ? 97
 - 5.2 The KahaDB message store 98
 - The KahaDB message store internals* 99
 - *The KahaDB message store directory structure* 100
 - *Configuring the KahaDB message store* 101
 - 5.3 The AMQ message store 103
 - The AMQ message store internals* 103
 - *The AMQ message store directory structure* 104
 - *Configuring the AMQ message store* 105
 - 5.4 The JDBC message store 107
 - Databases supported by the JDBC message store* 107
 - *The JDBC message store schema* 108
 - *Configuring the JDBC message store* 109
 - *Using the JDBC message store with the ActiveMQ journal* 111
 - 5.5 The memory message store 111
 - Configuring the memory store* 112
 - 5.6 Caching messages in the broker for consumers 113
 - How message caching for consumers works* 113
 - *The ActiveMQ subscription recovery policies* 114
 - *Configuring the subscription recovery policy* 115
 - 5.7 Summary 116
- ## 6 *Securing ActiveMQ* 117
- 6.1 Authentication 118
 - Configuring the simple authentication plug-in* 118
 - Configuring the JAAS plug-in* 121
 - 6.2 Authorization 123
 - Destination-level authorization* 124
 - *Message-level authorization* 127
 - 6.3 Building a custom security plug-in 131
 - Implementing the plug-in* 132
 - *Configuring the plug-in* 133
 - Testing the plug-in* 134
 - 6.4 Certificate-based security 135
 - Preparing certificates* 136
 - *Creating a truststore* 136
 - Configuring the broker* 138
 - *Authorization explained* 139
 - Testing it out* 139
 - 6.5 Summary 142

PART 3 USING ACTIVEMQ TO BUILD MESSAGING APPLICATIONS 143

- ### 7 *Creating Java applications with ActiveMQ* 145
- 7.1 Embedding ActiveMQ using Java 146
 - Embedding ActiveMQ using the BrokerService* 147
 - Embedding ActiveMQ using the BrokerFactory* 149
 - 7.2 Embedding ActiveMQ using Spring 150
 - Pure Spring XML* 151 ▪ *Using the BrokerFactoryBean* 153
 - Using Apache XBean with Spring* 154 ▪ *Using a custom XML namespace with Spring* 156
 - 7.3 Implementing request/reply with JMS 158
 - Implementing the server and the worker* 160 ▪ *Implementing the client* 162 ▪ *Running the request/reply example* 164
 - 7.4 Writing JMS clients using Spring 165
 - Configuring JMS connections* 166 ▪ *Configuring JMS destinations* 167 ▪ *Creating JMS consumers* 167
 - Creating JMS producers* 168 ▪ *Putting it all together* 171
 - 7.5 Summary 172
- ### 8 *Integrating ActiveMQ with application servers* 174
- 8.1 The sample web application 176
 - 8.2 Integrating with Apache Tomcat 181
 - Using local JNDI to integrate ActiveMQ with Tomcat* 182
 - Using global JNDI to integrate ActiveMQ with Tomcat* 184
 - 8.3 Integrating with Jetty 187
 - Using local JNDI to integrate ActiveMQ with Jetty* 187
 - Using global JNDI to integrate ActiveMQ with Jetty* 189
 - 8.4 Integrating with Apache Geronimo 192
 - Installing Geronimo and configuring the ActiveMQ plug-in in Geronimo* 192 ▪ *Configuring the ActiveMQ JMS resources in Geronimo* 196 ▪ *Preparing the sample application for deployment in Geronimo* 202 ▪ *Deploying and verifying the sample application in Geronimo* 205
 - 8.5 Integrating with JBoss 208
 - Installing JBoss and configuring the ActiveMQ resource adapter in JBoss* 209 ▪ *Configuring the ActiveMQ JMS resources in JBoss* 212 ▪ *Preparing the sample application for deployment*

- in JBoss* 212 ▪ *Deploying and verifying the sample application in JBoss* 215
- 8.6 ActiveMQ and JNDI 217
 - Client-side JNDI configuration* 217
- 8.7 Summary 220

9 *ActiveMQ messaging for other languages* 221

- 9.1 Adapting the stock portfolio example 222
- 9.2 Messaging for scripting languages 224
 - STOMP protocol basics* 224 ▪ *Configuring STOMP transport* 226 ▪ *Ruby STOMP consumer* 227 ▪ *Python STOMP consumer* 229 ▪ *PHP STOMP consumer* 233 ▪ *Perl STOMP consumer* 234 ▪ *Advanced messaging with STOMP* 236
- 9.3 Messaging for compiled languages 241
 - Writing a C# consumer (using the NMS API)* 242
 - Writing a C++ consumer (using the CMS API)* 244
- 9.4 Messaging on the web with ActiveMQ 247
 - Using the ActiveMQ REST API* 248 ▪ *Using the ActiveMQ Ajax API* 250
- 9.5 Summary 254

PART 4 ADVANCED FEATURES IN ACTIVEMQ255

10 *Deploying ActiveMQ in the enterprise* 257

- 10.1 Configuring ActiveMQ for high availability 258
 - Shared nothing master/slave* 258 ▪ *Shared storage master/slave* 261
- 10.2 How ActiveMQ passes messages across a network of brokers 263
 - Store and forward* 264 ▪ *Network discovery* 266
 - Network configuration* 268
- 10.3 Deploying ActiveMQ for large numbers of concurrent applications 272
 - Vertical scaling* 272 ▪ *Horizontal scaling* 275
 - Traffic partitioning* 275
- 10.4 Summary 276

- 11 ActiveMQ broker features in action 277**
- 11.1 Wildcards and composite destinations 278
 - Consume from multiple destinations using wildcards* 278
 - Sending a message to multiple destinations* 279
 - 11.2 Advisory messages 280
 - 11.3 Supercharge JMS topics by going virtual 284
 - 11.4 Retroactive consumers 286
 - 11.5 Message redelivery and dead-letter queues 287
 - 11.6 Extending functionality with interceptor plug-ins 288
 - Visualization* 288
 - *Enhanced logging* 290
 - *Central timestamp messages with the timestamp interceptor plug-in* 291
 - Statistics* 291
 - 11.7 Routing engine with Apache Camel framework 292
 - 11.8 Summary 294
- 12 Advanced client options 295**
- 12.1 Exclusive consumers 296
 - Selecting an exclusive message consumer* 296
 - *Using exclusive consumers to provide a distributed lock* 297
 - 12.2 Message groups 298
 - 12.3 ActiveMQ streams 301
 - 12.4 Blob messages 303
 - 12.5 Surviving network or broker failure with the failover protocol 305
 - 12.6 Scheduling messages to be delivered by ActiveMQ in the future 309
 - 12.7 Summary 311
- 13 Tuning ActiveMQ for performance 312**
- 13.1 General techniques 313
 - Persistent versus nonpersistent messages* 313
 - Transactions* 314
 - *Embedding brokers* 315
 - *Tuning the OpenWire protocol* 318
 - *Tuning the TCP transport* 319
 - 13.2 Optimizing message producers 319
 - Asynchronous send* 319
 - *Producer flow control* 320
 - 13.3 Optimizing message consumers 323

- Prefetch limit* 323 ▪ *Delivery and acknowledgment of messages* 325 ▪ *Asynchronous dispatch* 326
- 13.4 Tuning in action 327
- 13.5 Summary 330

14 *Administering and monitoring ActiveMQ* 331

- 14.1 The JMX API and ActiveMQ 332
 - Local vs. remote JMX access* 332 ▪ *Exposing the JMX MBeans for ActiveMQ* 334 ▪ *Exploring broker properties using the JMX API* 336 ▪ *Advanced JMX configuration* 339 ▪ *Restricting JMX access to a specific host* 340 ▪ *Configuring JMX password authentication* 341
- 14.2 Monitoring ActiveMQ with advisory messages 344
 - Configuring advisory support* 344 ▪ *Using advisory messages* 345 ▪ *Conclusion* 350
- 14.3 Tools for ActiveMQ administration 350
 - Command-line tools* 350 ▪ *Command agent* 355
 - JConsole* 357 ▪ *Web console* 359
- 14.4 Configuring ActiveMQ logging 360
 - Broker logging* 361 ▪ *Client logging* 362 ▪ *Internal broker event logging* 365
- 14.5 Summary 366
 - index* 367