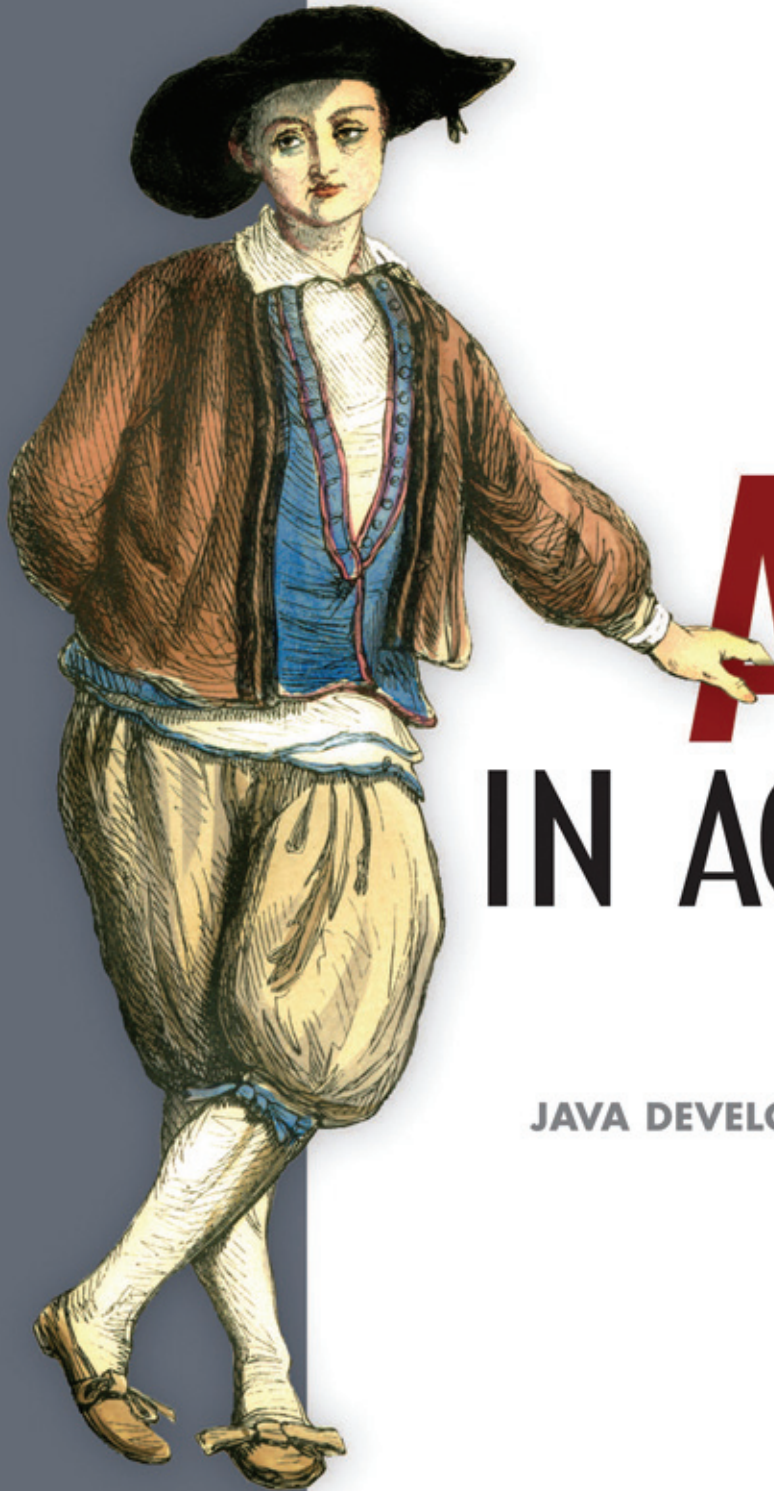


COVERS ANT 1.7



Steve Loughran
Erik Hatcher

ANT IN ACTION

Second Edition of
JAVA DEVELOPMENT WITH ANT

 MANNING

contents

preface to the second edition xix

foreword to the first edition xxi

preface to the first edition xxiii

acknowledgments xxv

about this book xxvii

about the authors xxxi

about the cover illustration xxxii

Introduction to the Second Edition 1

Part 1 Learning Ant 3

1 Introducing Ant 5

- 1.1 What is Ant? 5
 - The core concepts of Ant 6 ♦ Ant in action:
an example project 8
- 1.2 What makes Ant so special? 11
- 1.3 When to use Ant 12
- 1.4 When not to use Ant 13
- 1.5 Alternatives to Ant 13
 - IDEs 13 ♦ Make 14 ♦ Maven 16
- 1.6 The ongoing evolution of Ant 16
- 1.7 Summary 17

2	<i>A first Ant build</i>	19
2.1	Defining our first project	19
2.2	Step zero: creating the project directory	20
2.3	Step one: verifying the tools are in place	20
2.4	Step two: writing your first Ant build file	21
	Examining the build file	21
2.5	Step three: running your first build	23
	If the build fails	23
	◆ Looking at the build in more detail	25
2.6	Step four: imposing structure	27
	Laying out the source directories	28
	◆ Laying out the build directories	29
	◆ Laying out the distribution directories	29
	Creating the build file	31
	◆ Target dependencies	32
	Running the new build file	33
	◆ Incremental builds	34
	Running multiple targets on the command line	35
2.7	Step five: running our program	36
	Why execute from inside Ant?	36
	◆ Adding an "execute" target	37
	◆ Running the new target	38
2.8	Ant command-line options	39
	Specifying which build file to run	40
	◆ Controlling the amount of information provided	41
	◆ Coping with failure	42
	Getting information about a project	42
2.9	Examining the final build file	43
2.10	Running the build under an IDE	44
2.11	Summary	45
3	<i>Understanding Ant datatypes and properties</i>	47
3.1	Preliminaries	48
	What is an Ant datatype?	48
	◆ Property overview	48
3.2	Introducing datatypes and properties with <javac>	49
3.3	Paths	52
	How to use a path	53
3.4	Filesets	53
	Patternsets	54
3.5	Selectors	58
3.6	Additional Ant datatypes	59
3.7	Properties	61
	Setting properties with the <property> task	62
	◆ Checking for the availability of files: <available>	66
	◆ Testing conditions with <condition>	67
	◆ Creating a build timestamp with <tstamp>	69
	◆ Setting properties from the command line	70

3.8	Controlling Ant with properties	70
	Conditional target execution	71 ♦ Conditional build failure
	Conditional patternset inclusion/exclusion	72
3.9	References	73
	Viewing datatypes	73
3.10	Managing library dependencies	75
3.11	Resources: Ant's secret data model	76
3.12	Best practices	76
3.13	Summary	77
4	<i>Testing with JUnit</i>	79
4.1	What is testing, and why do it?	80
4.2	Introducing our application	81
	The application: a diary	81
4.3	How to test a program	83
4.4	Introducing JUnit	84
	Writing a test case	86 ♦ Running a test case
	Asserting desired results	87 ♦ Adding JUnit to Ant
	Writing the code	92
4.5	The JUnit task: <junit>	93
	Fitting JUnit into the build process	94 ♦ Halting the build when tests fail
	Viewing test results	96
	Running multiple tests with <batchtest>	98
4.6	Generating HTML test reports	99
	Halting the builds after generating reports	101
4.7	Advanced <junit> techniques	102
4.8	Best practices	106
	The future of JUnit	107
4.9	Summary	108
5	<i>Packaging projects</i>	110
5.1	Working with files	111
	Deleting files	112 ♦ Copying files
	Moving and renaming files	114
5.2	Introducing mappers	114
5.3	Modifying files as you go	119
5.4	Preparing to package	120
	Adding data files to the classpath	121 ♦ Generating documentation
	Patching line endings for target platforms	124

- 5.5 Creating JAR files 126
 - Testing the JAR file 128 ♦ Creating JAR manifests 129
 - Adding extra metadata to the JAR 131 ♦ JAR file best practices 132 ♦ Signing JAR files 132
- 5.6 Testing with JAR files 135
- 5.7 Creating Zip files 136
 - Creating a binary Zip distribution 137 ♦ Creating a source distribution 138 ♦ Zip file best practices 139
- 5.8 Packaging for Unix 139
 - Tar files 139 ♦ Generating RPM packages 143
- 5.9 Working with resources 143
 - A formal definition of a resource 143 ♦ What resources are there? 144 ♦ Resource collections 145
- 5.10 Summary 147

6 *Executing programs* 149

- 6.1 Running programs under Ant—an introduction 149
 - Introducing the <java> task 151 ♦ Setting the classpath 152
 - Arguments 153 ♦ Defining system properties 155
 - Running the program in a new JVM 156 ♦ JVM tuning 157
 - Handling errors 158 ♦ Executing JAR files 160
- 6.2 Running native programs 161
 - Running our diary as a native program 162 ♦ Executing shell commands 162 ♦ Running under different Operating Systems 163 ♦ Probing for a program 166
- 6.3 Advanced <java> and <exec> 167
 - Setting environment variables 167 ♦ Handling timeouts 168
 - Running a program in the background 169 ♦ Input and output 170 ♦ Piped I/O with an I/O redirector 171
 - FilterChains and FilterReaders 172
- 6.4 Bulk operations with <apply> 174
- 6.5 How it all works 176
 - <java> 176 ♦ <exec> and <apply> 177
- 6.6 Best practices 177
- 6.7 Summary 178

7 *Distributing our application* 179

- 7.1 Preparing for distribution 180
 - Securing our distribution 181 ♦ Server requirements 183

- 7.2 FTP-based distribution of a packaged application 183
 - Uploading to Unix 184 ♦ Uploading to a Windows FTP server 185 ♦ Uploading to SourceForge 186
 - FTP dependency logic 187
 - 7.3 Email-based distribution of a packaged application 188
 - Sending HTML messages 191
 - 7.4 Secure distribution with SSH and SCP 192
 - Uploading files with SCP 193 ♦ Downloading files with <scp> 195 ♦ Remote execution with <sshexec> 197
 - Troubleshooting the SSH tasks 197
 - 7.5 HTTP download 198
 - How to probe for a server or web page 199 ♦ Fetching remote files with <get> 200 ♦ Performing the download 201
 - 7.6 Distribution over multiple channels 203
 - Calling targets with <antcall> 203 ♦ Distributing with <antcall> 206
 - 7.7 Summary 208
- 8 *Putting it all together* 209
- 8.1 How to write good build files 209
 - 8.2 Building the diary library 210
 - Starting the project 210 ♦ The public entry points 211
 - Setting up the build 212 ♦ Compiling and testing 216
 - Packaging and creating a distribution 218 ♦ Distribution 222
 - 8.3 Adopting Ant 225
 - 8.4 Building an existing project under Ant 228
 - 8.5 Summary 230

Part 2 Applying Ant 231

- 9 *Beyond Ant's core tasks* 233
- 9.1 The many different categories of Ant tasks 234
 - 9.2 Installing optional tasks 236
 - Troubleshooting 238
 - 9.3 Optional tasks in action 239
 - Manipulating property files 239 ♦ Improving <javac> with dependency checking 241
 - 9.4 Software configuration management under Ant 243

9.5	Using third-party tasks	245
	Defining tasks with <taskdef>	246 ♦ Declaring tasks defined in property files 247 ♦ Defining tasks into a unique namespace 248 ♦ Defining tasks from an Antlib 249
9.6	The Ant-contrib tasks	250
	The Ant-contrib tasks in action	253
9.7	Code auditing with Checkstyle	259
9.8	Summary	263
10	<i>Working with big projects</i>	264
10.1	Master builds: managing large projects	265
	Introducing the <ant> task	266 ♦ Designing a scalable, flexible master build file 268
10.2	Controlling child project builds	270
	Setting properties in child projects	270 ♦ Passing down properties and references in <ant> 272
10.3	Advanced delegation	275
	Getting data back	276
10.4	Inheriting build files through <import>	277
	XML entity inclusion	277 ♦ Importing build files with <import> 278 ♦ How Ant overrides targets 279
	Calling overridden targets	280 ♦ The special properties of <import> 281
10.5	Applying <import>	283
	Extending an existing build file	283 ♦ Creating a base build file for many projects 284 ♦ Mixin build files 286
	Best practices with <import>	287
10.6	Ant's macro facilities	288
	Redefining tasks with <presetdef>	288 ♦ The hazards of <presetdef> 290
10.7	Writing macros with <macrodef>	291
	Passing data to a macro	292 ♦ Local variables 294
	Effective macro use	295
10.8	Summary	296
11	<i>Managing dependencies</i>	297
11.1	Introducing Ivy	299
	The core concepts of Ivy	299
11.2	Installing Ivy	301
	Configuring Ivy	302

11.3	Resolving, reporting, and retrieving	304
	Creating a dependency report	305 ♦ Retrieving artifacts
	Setting up the classpaths with Ivy	307
11.4	Working across projects with Ivy	308
	Sharing artifacts between projects	308 ♦ Using published artifacts in other projects
	Using Ivy to choreograph builds	313
11.5	Other aspects of Ivy	315
	Managing file versions through Ivy variables	315
	Finding artifacts on the central repository	316
	Excluding unwanted dependencies	317
	Private repositories	317 ♦ Moving to Ivy
		318
11.6	Summary	318
12	<i>Developing for the Web</i>	320
12.1	Developing a web application	321
	Writing a feed servlet	323 ♦ Libraries in web applications
	Writing web pages	325
	Creating a web.xml file	327
12.2	Building the WAR file	328
12.3	Deployment	329
	Deployment by copy	330
12.4	Post-deployment activities	331
	Probing for server availability	331 ♦ Pausing the build with <sleep>
		333
12.5	Testing web applications with HttpUnit	333
	Writing HttpUnit tests	334 ♦ Compiling the HttpUnit tests
	Running the HttpUnit tests	338
12.6	Summary	339
13	<i>Working with XML</i>	340
13.1	Background: XML-processing libraries	341
13.2	Writing XML	341
13.3	Validating XML	343
	Validating documents using DTD files	345 ♦ Validating documents with XML Schema
	Validating RelaxNG documents	349
13.4	Reading XML data	352
13.5	Transforming XML with XSLT	353
	Defining the structure of the constants file	354

	Creating the constants file	355	◆	Creating XSL style sheets	355	◆	Initializing the build file	358
13.6	Summary	362						
14	<i>Enterprise Java</i>	363						
14.1	Evolving the diary application	364						
14.2	Making an Enterprise application	365						
14.3	Creating the beans	366						
	Compiling Java EE-annotated classes	368	◆	Adding a session bean	369			
14.4	Extending the web application	371						
14.5	Building the Enterprise application	373						
14.6	Deploying to the application server	378						
14.7	Server-side testing with Apache Cactus	378						
	Writing a Cactus test	379	◆	Building Cactus tests	380			
	The Cactus Ant tasks	381	◆	Adding Cactus to an EAR file	382	◆	Running Cactus tests	383
	Diagnosing EJB deployment problems	384						
14.8	Summary	385						
15	<i>Continuous integration</i>	387						
15.1	Introducing continuous integration	388						
	What do you need for continuous integration?	390						
15.2	Luntbuild	391						
	Installing Luntbuild	393	◆	Running Luntbuild	393			
	Configuring Luntbuild	394	◆	Luntbuild in action	400			
	Review of Luntbuild	401						
15.3	Moving to continuous integration	402						
15.4	Summary	404						
16	<i>Deployment</i>	406						
16.1	How to survive deployment	407						
16.2	Deploying with Ant	410						
16.3	Database setup in Ant	411						
	Creating and configuring a database from Ant	412						
	Issuing database administration commands	413						
16.4	Deploying with SmartFrog	415						
	SmartFrog: a new way of thinking about deployment	415						
	The concepts in more detail	417	◆	The SmartFrog components	425			

- 16.5 Using SmartFrog with Ant 426
 - Deploying with SmartFrog 428 ♦ Deploying with the <deploy> task 433 ♦ Summary of SmartFrog 435
- 16.6 Embracing deployment 436
- 16.7 Summary 438

Part 3 Extending Ant 441

17 Writing Ant tasks 443

- 17.1 What exactly is an Ant task? 444
 - The life of a task 445
- 17.2 Introducing Ant's Java API 446
 - Ant's utility classes 451
- 17.3 A useful task: <filesize> 453
 - Writing the task 453 ♦ How Ant configures tasks 455
 - Configuring the <filesize> task 457
- 17.4 Testing tasks with AntUnit 458
 - Using AntUnit 458 ♦ Testing the <filesize> task 460
 - Running the tests 461
- 17.5 More task attributes 463
 - Enumerations 463 ♦ User-defined types 465
- 17.6 Supporting nested elements 465
- 17.7 Working with resources 467
 - Using a resource-enabled task 470
- 17.8 Delegating to other tasks 471
 - Setting up classpaths in a task 472
- 17.9 Other task techniques 476
- 17.10 Making an Antlib library 478
- 17.11 Summary 481

18 Extending Ant further 483

- 18.1 Scripting within Ant 484
 - Writing new tasks with <scriptdef> 486
 - Scripting summary 489
- 18.2 Conditions 490
 - Writing a conditional task 492
- 18.3 Writing a custom resource 493
 - Using a custom resource 496 ♦ How Ant datatypes handle references 496

- 18.4 Selectors 497
 - Scripted selectors 499
- 18.5 Developing a custom mapper 499
- 18.6 Implementing a custom filter 501
- 18.7 Handling Ant's input and output 503
 - Writing a custom listener 505 ♦ Writing a custom logger 509
 - Using loggers and listeners 511 ♦ Handling user input with an InputHandler 512
- 18.8 Embedding Ant 512
- 18.9 Summary 514

appendix A Installation 516

- A.1 Before you begin 516
- A.2 The steps to install Ant 517
- A.3 Setting up Ant on Windows 517
- A.4 Setting up Ant on Unix 518
- A.5 Installation configuration 520
- A.6 Troubleshooting installation 520

appendix B XML Primer 525

- B.1 XML namespaces 529

appendix C IDE Integration 531

- C.1 How IDEs use Ant 531
- C.2 Eclipse <http://www.eclipse.org/> 533
- C.3 Sun NetBeans <http://www.netbeans.org/> 539
- C.4 IntelliJ IDEA <http://intellij.com/> 543
- C.5 Building with Ant and an IDE 546

index 549