

# index

---

## Symbols

<> resource collection 145  
\${...} notation 51

## A

acceptance tests 84

<and> condition 67

### Ant

adopting Ant 225

automating repetitive builds  
387

build, failure actions 8

build, successful 23

calling targets directly 203

catching exceptions in a build  
255

clean build 309

command line 33

command-line options 39

committer 16

concepts 6

creating Java source during a  
build 355

error messages 23

extending 7

failure handling 13

generated files 359

going offline 199

history 16

IDE support 12

installation configuration 520

installation, bad 21

installing 20, 516

iteration 256

Java API. *See* Java API

migrate after a deadline 229

migrating to 228

overview 5

performance impact of

<antcall> 208

project has deliverables 225

proxy setup in ANT\_OPTS  
520

return codes 397

running 23

running the same target twice  
203, 206

scalability 313

scripting support. *See* script

setting up libraries 91, 181

steps to migrate to 229

using from an IDE 546

when not to use 13

why tests matter 210

writing good build files 209

XML support 340

### Ant project

directory setup 20

*See also* directory layout

distribution directories 29

target dependencies 32

### Ant task

definition 444

lifecycle 445

task failure halts the build 25  
writing 443

<ant> task 138, 266, 296

implementation 448

inheritall attribute 270

ordering 267–268

passing datatypes 273

passing down properties and  
references 270, 272

preset and macro tasks inherited  
290

properties not passed back to  
caller 203, 205

property element 273

property inheritance rules 271

propertyset element 274

reference element 273

repeated calls using <for> 256

ANT\_ARGS 520

not supported under IDEs 532

ANT\_HOME environment vari-  
able 517

troubleshooting 522

ANT\_OPTS 341, 520

adding a custom logger 510

environment variable

troubleshooting 523

not supported under IDEs 532

setting in a continuous integra-  
tion build 397

<antcall> task 203, 206, 208,  
211, 266

best practises 207

- <antcall> task (*continued*)
    - command-line properties
      - passed down 206
    - datatype inheritance 205
    - dependencies called 204
    - dependency logic 203
    - forwarding targets to other
      - build files 269
    - implementation 448
    - inheritall attribute 205, 270
    - inheritance model 205
    - inheritrefs attribute 206
    - listening to 508
    - performance 208
    - properties not passed back to
      - caller 203, 205
    - property inheritance rules 271
    - setting properties 204
    - targets designed for 222
  - <antcallback> task (ant-contrib) 250, 276
  - ant-contrib project 250
  - <antfetch> task (ant-contrib) 276
  - Antlib
    - ant-contrib 250
    - antlib.xml file 479, 482
    - AntUnit 458
    - declaring 249, 480
    - defining conditions in
      - antlib.xml 491
    - definition 236
    - Ivy 301
    - loading by <typedef> task 253
    - loading by XML namespace
      - declaration 253
    - loading process 479
    - making 478
    - namespace URIs 480
    - Subversion 245
    - supporting <scriptdef> tasks 488
    - task.properties files 478
    - URIs 249
  - antlib.xml, using <presetdef> and <macrodef> 481
  - <antlr> task 235
  - .antrc file, in Linux 519
  - AntUnit 482
    - home page 458
    - introducing 458
    - limitations 463
    - list of assertions 459
    - no support for listeners and loggers 507
    - running the tests 461
    - setUp and tearDown 458, 460
    - tasks 459
    - testing
      - custom condition 491
      - custom filter 503
      - custom mapper 501
      - custom selector 498
      - property 460
      - <scriptdef> tasks 487–488
      - scripts 485
      - unit testing for Ant tasks 458
      - using 458, 460
  - <antunit> task (AntUnit) 458–459
  - <antversion> condition 67
  - Apache Forrest. *See* Forrest
  - Apache Ivy. *See* Ivy
  - Apache Maven. *See* Maven
  - application server 364
    - Glassfish 364
    - JBoss 364
  - application.xml 374, 376, 382
    - listing libraries 375
  - <apply> class, implementation 475
  - <apply> task 174
    - best practices 177
    - dependency rules 175
    - implementation 177
    - mappers 114
  - <artifactproperty> task (Ivy) 301
  - <artifactreport> task (Ivy) 301, 305
  - <assertDestIsOutOfDate> task (AntUnit) 460
  - <assertDestIsUpToDate> task (AntUnit) 460
  - <assertEquals> task (AntUnit) 459, 503
  - <assertFalse> task (AntUnit) 459, 498
  - <assertFileDoesntExist> task (AntUnit) 460
  - <assertFileExists> task (AntUnit) 459
  - <assertFilesDiffer> task (AntUnit) 460
  - <assertFilesMatch> task (AntUnit) 460
  - <assertLogContains> task (AntUnit) 459–461, 464, 488
  - <assertPropertyContains> task (AntUnit) 459
  - <assertPropertyEquals> task (AntUnit) 459
  - <assertPropertySet> task (AntUnit) 459–460, 470
  - <assertReferenceIsType> task (AntUnit) 460
  - <assertReferenceSet> task (AntUnit) 460
  - <assertTrue> task (AntUnit) 459
  - Atom
    - described in RelaxNG 351
    - feeds are XML 349
    - Java support 322
    - publishing an Atom feed 323
    - validating with <jing> 351
  - attributes, setting in custom task 456
  - autoproxy 39
  - <available> condition 67, 72, 128, 138, 166
  - <available> task 66, 166
    - as a condition 221
- ## B
- basedir 61
  - <basename> task 195, 225, 294
  - Bean Scripting Framework (BSF) 484
  - big projects 264
    - absolute properties work better 272
    - choreographing projects 268
    - delegating with <ant> 267

- big projects (*continued*)
    - eliminating repetition 288
    - managing child projects 270
    - master build files 265
    - property inheritance 270
    - shared property files 274, 285
    - sharing build files with
      - <import> 277
  - BSF. *See* Bean Scripting Framework (BSF)
  - build failed
    - circular dependency 33
    - compile failed 25
    - failed to create task or type 23
    - Java source errors 25
    - reporting a failure 24
    - unsupported attribute 24
  - build file 7
    - all out-of-target tasks are run
      - first 212
    - build files are XML 21
    - CDATA sections 343
    - closing 225
    - definition 21
    - designing for easy overriding 272
    - encoding 528
    - entry points 211, 226
    - evolving 228–229
    - example 9
    - extending with <import> 277
    - initialization 212
    - master builds 265
    - order of targets 32
    - parameters specified as nested
      - elements 23
    - parser errors 526
    - philosophy 209
    - <project> element 22
    - public targets 211
    - simple example 21
    - specifying which to run 40
    - structure 21
    - <target> element 22
    - task parameters 23
    - top of the file 210
    - tree representation 21
    - See also* XML
  - buildfile 39–40
  - <buildlist> task (Ivy) 301, 313–314, 318
    - defects in Ivy 1.4.1 318
    - indirect dependencies 315
  - <buildnumber> task (Ivy) 240, 302, 308, 310
  - build.properties file 212
    - overriding build.xml definitions 213
    - relative filenames 214
    - skipping <input> tasks 220
  - <bzip2> task 141
  - <bzip2resource> resource 144
- C**
- C++
    - macros 288
    - mixin classes 287
  - <cab> task 235
  - <cachefilepath> task (Ivy) 306
  - <cachefileset> task (Ivy) 302, 306
  - <cachepath> task (Ivy) 302, 412
  - <cactifyear> task (Cactus) 381–382
  - <cactifywar> task (Cactus) 381–382
  - Cactus 108, 378
    - adding to an EAR or WAR file 382
    - Ant tasks 381
    - architecture 379
    - compiling tests 380
    - deploying with Cargo 410
    - fetching through Ivy 380
    - test cases 379
    - testing with HSQLDB 411
  - <cactus> task 381, 383
  - Cargo 410
  - CDATA 527
  - chained mapper 118
  - Checkstyle 259
    - installing 261
  - <checkstyle> task 259
  - <checksum> condition 67
  - <checksum> task 180–181, 222
    - as a condition 196, 225
    - creating checksum files 182
    - parameters 182
    - saving to a property 182
    - validating downloads 201
  - checksums
    - creating checksums 181
    - verifying checksums 196
  - <chmod> task 133, 220
    - implementation 165, 177
    - preparing scripts for <exec> 162
  - circular dependency 33
  - classloaders
    - BEA Weblogic 374
    - endorsed directories 375
    - Java EE 374
    - JBoss 374
    - JDBC Drivers 375
  - CLASSPATH
    - environment variable 518
    - troubleshooting 522
  - classpath, setting up a forked
    - <junit> 104
  - classpaths
    - adding data files 121
    - adding JAR files 135
    - adding tests 95
    - Ant's classpath 152
    - controlling library versions
      - through properties 316
    - creating compile and run classpaths 152
    - error messages from <java> 151
    - limitations of filesets 298
    - management 75, 297, 302
    - setting up with filesets 298
    - setting up with Ivy 306–307
  - ClearCase 243
  - code auditing with Checkstyle 259
  - command line
    - controlling verbosity 41
    - multiple targets 35
    - options 39
    - running multiple targets 35, 211
    - running the default target 33

- command line (*continued*)
  - selecting build file options.
    - See* -buildfile
  - setting properties from 70
  - setting properties. *See* -D and -propertyfile
- composite mapper 118
- Concurrent Version System (CVS)
  - binding Luntbuild to a CVS repository 395
  - tasks 244
- <condition> task 67, 129, 166, 216
  - <os> test 67
  - using a custom condition 491
  - See also* <waitfor>
- conditional build failure 72
- conditional logic 67
- conditional targets 71, 166
  - common mistakes 71
  - evaluation process 71
- conditions 490
  - AntUnit assertions, use in 460
  - <checksum> 196
  - <http> 199, 331
  - <if><then><else>, use in 254
  - <isreachable> 200
  - <socket> 199, 331
  - supported conditions 67
  - <typefound> 237
  - writing a conditional task 492, 515
- <configure> task (Ivy) 302, 304
- <contains> condition 67
- continuous deployment 438–439
- continuous integration 387
  - Anthill 390
  - Bamboo 390
  - Beetlejuice 390
  - blame assignment 391, 399, 402–403
  - build failures matter 402
  - build triggers 399
  - Continuum 390
  - cross-platform testing 403
  - CruiseControl 388, 390, 405
  - dedicated server 390, 398
  - definition 388
  - developers 403
  - Gump 390
  - hosting under an application server 393
  - Hudson 390
  - Luntbuild 390
  - managers 403
  - mistakes 404
  - moving to 402
  - notification 388, 394–395, 399
  - one server for every branch 403
  - open source 390, 404
  - post-build actions 399, 401
  - preparations 390
  - products 389
  - publishing test results 397, 401
  - reporting and status pages 400–401, 404
  - running builds by hand 400
  - schedules 398
  - shared responsibility 403
  - start with a working build 391
  - TeamCity 390
  - test runs 400
  - time to set up 391
  - versus nightly builds 388
  - virtual developer 391
  - why not use OS schedulers 388
  - working with Ivy 400
  - See also* Luntbuild
- Continuous 243
- <copy>
  - broken by the global filter 120
  - dependency checking 113
  - file permissions 114
- <copy> task 113, 500
  - backups, making 116–117
  - bulk copies 113
  - copying
    - file to a directory 113
    - fileset 48
    - pattern 121
    - resources 496
    - single file 113
  - expanding properties in an XML file 377
  - extracting content from a compressed file 146
  - file permissions 134
  - fileset element 113, 116–117
  - filtering files 119
  - forcing overwrites 113
  - mappers 114–115
  - overwrite attribute 113, 120
  - preservelastmodified attribute 113
  - preserving timestamps 113
  - resource support 146
  - todir attribute 113
  - tofile attribute 113
  - using a custom selector 498
  - using a scripted mapper 500
  - using filtersets 218
  - using selectors 59
  - working with Java packages 117
- copying files 113
- Core task 234
- cross-platform directory and path separators 52
- custom condition 490, 515
  - defining 490
    - with <typedef> 491
  - implementing 490
    - in a scripting language 492
  - requirements 490
- custom filter 501, 515
  - implementing 501
  - requirements 502
  - scripting 503
  - testing 502–503
- custom listener 515
  - avoiding trouble 511
  - implementing 505
  - using 507–508
  - using with <trycatch> 509
- custom logger 509, 515
  - avoiding trouble 511
  - implementing 509
  - switching to 510
  - using 510
- custom mapper 515
  - declaring in an Antlib 500
  - requirements 500

- custom mapper (*continued*)
    - scripting 500
    - testing 501
  - custom resource
    - copying 496
    - references 495–496
    - touchable resources 497
    - using in a build file 496
  - custom selector 515
    - attributes 498
    - defining a custom selector 498
    - implementing 498
    - testing 498
  - custom task 443
    - add() 465
    - addConfiguredXXX 465
    - adding a custom JAR to Ant's classpath 480
    - addText() method 413, 477
    - addXXX 465
    - Boolean attributes 456
    - character attribute 457
    - compiling 454
    - conditional tasks must extend ConditionBase 493
    - configuring 455
    - createXXX 465
    - custom attribute types 465
    - debugging 449, 462, 477
    - declaring an Antlib 479
    - declaring through property files 478
    - defining with <taskdef> 444, 455, 479
    - definition 234, 444
    - delegating to <java> 475
    - delegating to other tasks 471
    - enumerations 463
    - error handling 476
    - example 444
    - failOnError attribute 456, 476
    - file attributes 457
    - handling inline text 413, 477
    - implementing a conditional task 492
    - loading classes 472
    - logging. *See* Java API logging
    - nested elements 465, 487, 493
    - numeric attributes 456
    - Object/XML mapping 456
    - order of setter invocation 473
    - packaging 478
    - Path
      - attributes and elements 472
      - reference 473
    - resources with 467, 493
    - setting attributes 456
    - string attributes 456
    - supporting conditions 492
    - testing resource use 470
    - testing. *See* AntUnit
    - using 445, 455, 473, 475, 477
  - CVS. *See* Concurrent Version System (CVS)
  - < cvs > task 235, 244, 391
    - checking out source 244
    - failonerror attribute 244
    - troubleshooting 245
  - < cvs changelog > task 244
  - < cvs tagdiff > task 244
- D**
- D 39, 65, 103, 185
    - choosing distribution destinations 186, 193
    - command-line properties
      - always propagate to < ant-call > and < ant > 206
    - command-line properties are always inherited 271
  - d. *See* -debug
  - databases 411
    - HSQLDB 365
    - installing MySQL 411
    - JDBC drivers 375
    - setting up with < sql > 412
  - datatype 48
    - accessing inside a task 448
    - best practices 76
    - binding to a project 447
    - converting to a string 74
  - dirset. *See* dirset
  - element naming conventions 57
  - filelist. *See* filelist
  - fileset. *See* fileset
  - filterset. *See* filterset
  - inheritance across projects 273
  - patternset. *See* patternset
  - references 73
  - selectors. *See* selectors
  - < tarfileset >. *See* < tarfileset >
  - third-party 247
  - viewing 73
  - < zipfileset >. *See* < zipfileset >
  - debug 39, 41, 449, 505
  - debugging an Ant build 533, 540
  - < defaultexcludes > task 56
  - defect tracking, deployment problems 410
  - < delete > task 31, 112, 211
    - dangers 112
    - dir attribute 112
    - failonerror attribute 112, 121
    - file attribute 112
    - filesets 112
    - quiet attribute 112
    - verbose attribute 113
  - deleting files 112
  - < deliver > task (Ivy) 302
  - < depend > task 235, 241–242
    - caching with the cache attribute 242
    - checking dependencies against JAR files 242
    - srcdir and destdir are required attributes 242
    - transitive dependencies 242
  - dependency
    - checking using the < uptodate > task 118
    - graph 8
    - management 297
    - management problems 297, 318
  - dependency injection 370, 372
  - deployment 406
    - automating 407–408, 415, 435, 438–439
    - binding to system properties 432
  - Cargo 410
  - choreography 415

- deployment (*continued*)
    - configuration is 416
    - continuous 438–439
    - database configuration 412, 416, 418
    - databases 409, 411, 435
    - definition 406, 416
    - deploy-by-copy 330, 339, 378, 406
    - deploying to a cluster 421–422
    - deploying with Ant 410
    - deploying with SmartFrog 415
    - EAR files 378
    - embracing 436
    - inserting a pause 333
    - integrating with development process 436
    - operations teams 407–408
    - probing server availability 331
    - SmartFrog 411
    - testing 407, 409, 430, 434–435, 437, 439
    - track defects 407, 410, 439
    - troubleshooting JNDI problems 384
    - waiting for 332
    - web applications 329, 406, 435
  - <description> element 42–43, 210
  - diagnostics 39, 91, 181, 239, 520, 522
    - IDE support 532
    - implementation 409
    - XML parser information 341
    - See also* <diagnostics>
  - <diagnostics> task 341
    - IDE analysis 532
  - diary application
    - as a web application 320
    - child projects 265
    - clean builds 120
    - entry point 150
    - Event class 92
    - introduction 81
    - packaging 110
    - patching the documentation 119
  - persistence design 366
  - scaling up 264–265
  - standard entry points for build files 266
  - WAR libraries 325
  - web application directory structure 322
  - directories
    - creating 111
    - deleting 112
  - directory layout/structure 27
    - build directories 28–30
    - classes directory 29
    - deployment configurations 429
    - distributables 29
    - distribution directories 28–30
    - final structure 30
    - intermediate files 29
    - laying out the project 226
    - managing JAR files 75
    - naming in properties 77
    - requirements for dependency checking 28
    - simple structure 20
    - source directories 28
    - source file directory structure 28
    - structuring 27
    - test files 93
    - web applications 322
  - <dirname> task 282
  - dirset 60
  - distribution 179
    - creating file checksums 181
    - distributing
      - by email 180, 188, 203
      - over multiple channels 203, 206
      - with FTP 206
      - with SSH 180, 192, 203, 206
    - FTP 180, 183–185, 203
    - HTTP download 180, 198, 200
    - probing server availability 199
    - security 181–182, 192, 208
    - server setup 183
  - SSH 201
  - tasks 180
  - through SourceForge 186
- ## E
- EAR files 373–375
    - building with the <ear> task 377
    - classpath setup 374
    - deployment 378
    - overview 126
    - patching with Cactus 382
    - publishing under Ivy 309
  - <ear> task 374, 377, 385
    - dependency logic 128
  - <echo> task 352, 362, 449, 525–526
  - embedding 514
  - generating XML 342, 345
  - implementation 448
  - printing a datatype 73
  - printing at different log levels 41
  - printing paths 314
  - use in debugging 74
  - verbose logging 291–292
  - <echoproperties> task 62, 353
  - <echoxml> task 293, 341, 362
  - limitations 342
  - Eclipse
    - Ant project, and 534–536
    - debugging a build 533–534
    - editing a build file 534
    - IDE is in charge 533, 539
    - <import> trouble 535, 538
    - selecting targets 535
    - SmartFrog plugin 426
    - using compiler outside the IDE 538
  - EJB. *See* Java EE
  - <ejb-jar> task 371
  - emacs 39, 41–42, 505
  - email. *See* <mail> task
  - embedding ant 512–515
  - Enterprise Java 363
  - Enterprise Java Beans. *See* Java EE
  - Entity Bean, definition 365

- environment variables
    - accessing as properties 65, 166
    - <java> and <exec> 167
    - searching the PATH 166
  - <equals> condition 67
  - error messages 236–238
  - <exec> class, implementation 475
  - <exec> task 161, 223
    - best practices 177
    - capturing output 170
    - changing behavior with <pre-setdef> 288
    - configuring the input source 170
    - environment variables 167
    - error attribute 170
    - errorproperty attribute 170
    - failIfExecuteFails attribute 162
    - failonerror attribute 162, 171, 288
    - I/O 170
    - I/O redirector 171
    - implementation 177
    - input attribute 170
    - inputstring attribute 170
    - issuing CVS commands 245
    - logerror attribute 170–171
    - os attribute 163, 289
    - osfamily attribute 164
    - output attribute 170
    - outputProperty attribute 170
    - post-processing output 172
    - resultproperty attribute 171
    - running in the background 169
    - running platform-specific programs 163
    - shell scripts 162
    - spawn attribute 169
    - timeout attribute 168
    - why wild cards don't work 162
  - executing programs 161
    - best practices 177
    - searching for executables 166
  - <expandproperties>
    - filter implementation 503
    - filterchain 377
  - <expectfailure> task (AntUnit) 458–459, 461–462
- F**
- f 39, 91
  - <fail> task 72, 102, 129, 184–185, 199, 217, 258, 332, 338
    - checking for a task 237
    - failing when tests fail 218
    - if and unless attributes 72
    - looking for files 221
    - nested condition 68
    - using a custom condition 491
    - validating downloads 196, 201
    - See also* <waitfor>
  - failonerror attribute 159
  - <faultingwaitfor> task (SmartFrog) 432
  - fetch.xml 91
  - file 61
  - <file> resource 144
  - filelist 59
    - as a resource 143
    - resource view 76
  - <filelist> resource collection 145, 147
  - files 111
    - dependency checking 118
    - filtering 119
    - moving, copying, deleting 112, 120
  - <files> resource collection 145
  - fileset 53
    - as a resource 143
    - classpath setup 298, 306
    - defaultexcludes attribute 56, 112
    - examples 57
    - excluding SCM files 56
    - implicit filesets 57
    - includes attribute 54
    - ordering of files 52
    - patternset support 54
    - resolution 57
    - resource view 76
    - selectors 58–59
    - zipfileset extension 138
  - fileset datatype, building WAR files 328
  - <fileset> resource collection 145, 147
  - <filesmatch> condition 67, 128–129
  - <filter> 120
  - FilterChain 172
  - filtering
    - dangers of the global filter 120
    - token substitution 119
  - FilterReaders 172
  - filterset 60
    - use in <copy> 218
  - <filterset> datatype 77
  - find 39–40
  - <findrevision> task (Ivy) 302
  - <fixCrLf> task 124, 218
    - as a filter reader 173
    - dependency checking 124
    - Java source files 125
    - tab to space conversion 125
    - targeting multiple platforms 124
  - flatten mapper 116, 118
  - <for> task (ant-contrib) 251, 256
  - <foreach> task 256–257
  - Forrest 125
  - <ftp> task 180, 183, 235
    - dangers 188
    - dependency logic 187
    - depends attribute 187–188
    - library dependencies 181
    - timediffauto attribute 187
    - upload
      - to SourceForge 186
      - to Unix 184
      - to Windows 185
  - FTP uploads 183
  - functional tests 84
  - <functionaltest> task, SmartFrog 437
- G**
- <genkey> task 134, 220
    - implementation 471
  - <get> task 180, 200, 254, 339
    - dependency logic 200
    - fetching XML 350

- <get> task (*continued*)
  - handling failures with
    - <trycatch> 255
  - testing system health 409
- Glassfish 364
- Glob mapper 116, 118
  - use in <apply> 176
- <gzip> task 141
- <gzipresource> resource 144

## H

- h. *See* -help
- happy.jsp
  - checking in a deployment 425
  - checking in production 409
  - fetching with SmartFrog 425
- <hasreespace> condition 67
- <hasmethod> condition 67
- help 39
- HTTP
  - downloading files and pages 200
  - error codes 199
- <http> condition 67, 199–200, 331–333, 339
- HttpUnit 334
  - excluding unwanted dependencies 317
  - expecting a failure 336
  - fetching a page 335
  - following a link 336
  - installing 334
  - running with <junit> 338
  - test results 338
  - testing system health 409
  - using with JUnit 334

## I

- I/O redirector 171
- <icontract> task 235
- IDE
  - debugging an Ant build 532–533, 540, 543
  - Eclipse. *See* Eclipse
  - IntelliJ IDEA. *See* IDEA

- IDEA 543
  - Ant integration 543
  - background builds 546
  - binding to an Ant project 544
  - configuring Ant 546
  - detecting unset properties 543
  - running Ant targets within an IDE build 545–546
- identity mapper 115, 118
- IDEs 13
  - Ant hosting 44
  - Ant integration 531
  - Ant versions 532
  - best practises 547
  - <input> task 532, 539
  - integration with Ant 531, 546
  - limitations 13
  - listeners and loggers 532
  - NetBeans. *See* NetBeans
  - public targets 211
  - role 12
  - setting up Ant's classpath 531–532, 536, 543, 546
  - using the <diagnostics> task 239
  - XML parser support 341
- <if> task (ant-contrib) 254
- <image> task 235
- <import> task 277–278, 296
  - applying 283
  - best practices 287
  - building a web application 323
  - calling overridden targets 280
  - comparison with XML entities 279
  - creating a common build file 283–285, 296
  - Eclipse workarounds 539
  - extending a build file 283, 296
  - extending an existing build file 330, 350
  - imported targets 279
  - importing macro definitions 295
  - milestone targets 285–286, 296, 323
  - mixin build files 283, 286, 295–296, 323

- override rules 279
- overriding targets 280, 361
- problems under Eclipse 535, 538
- projects need different names 282
- properties 281
- incremental builds 34
- info 449
- <info> task (Ivy) 302
- <input> task 132, 220, 512
  - bypassing the prompt 133
  - IDE support 532
  - input handlers 39
  - unattended builds 133
  - under NetBeans 539–540
- inputhandler 39, 133, 512
- <install> task (Ivy) 302, 317
- installing Ant 516
  - from RPM 519
- Integrated Development Environments. *See* IDEs
- <intersect> resource collection 146
- <isfailure> condition 67
- <isfalse> condition 67
- <isfileselected> condition 67
- <isreachable> condition 67, 200
- <isreference> condition 68
- <isset> condition 68, 216
- <issigned> condition 68
- <istrue> condition 68
- Ivy 276
  - actions when publishing 309
  - artifacts
    - declaring to publish 309
    - finding 316
    - missing are downloaded 305, 312
    - publishing 308, 310, 318
    - resolving 304, 311
    - retrieved are cached 305, 307
    - retrieving 302, 306, 318, 376
    - retrieving published 310
    - setting up classpaths with retrieved 307

- Ivy (*continued*)
  - sharing between projects 308, 311
  - building a web application 325
  - building an EAR 375
  - <builddist> task 314–315
  - choreographing builds 313, 318
  - concepts 299
  - configurations 299–300, 310
  - configuring 302–304
  - conflict managers 301
  - controlling versions through properties 316, 318
  - custom configurations 311
  - dependencies 300, 307, 317–318
    - conflict 312–313
    - management through configurations 310
    - resolution 300, 303, 316
  - directory synchronization 307
  - downloading JDBC drivers 412
  - dynamic dependencies 310
  - eviction 312–313
  - excluding libraries 337
  - excluding unwanted dependencies 317, 380
  - importing the Servlet API 325
  - installation 301
  - installing a library to a team repository 368
  - integration with continuous integration 400
  - <ivy report> task 317
  - Ivy variables 315–318
  - ivyconf-local.xml file 303
  - ivyconf.xml file 302–304, 316, 318
  - ivy.xml 375
  - ivy.xml file 299–300, 302, 308, 316, 318–319
  - JavaEE development 368
  - latest-integration version 310
  - missing libraries 368
  - ordering module builds 313
  - overriding library versions 316
  - printing the build order 314
  - private configurations 311
  - private repositories 317
  - public configurations 310
  - reporting 301–302, 305–306, 312, 315, 317
  - <resolve> task 304–305
  - resolvers 302–303, 308
  - retrieved metadata is cached 310
  - retrieving 301
  - retrieving a third-party task 350
  - retrieving the libraries for a WAR file 325
  - security 317
  - setting up JAR files for an IDE 532
  - setting up the HttpUnit class-path 337
  - shared configurations 311, 318
  - task list 301
  - team 299
  - transient dependencies 299
  - transitive dependencies 300, 310, 317
  - using the Maven2 repository 303
  - version numbering 308
- J**
  - JAR files 126
    - adding metadata 131
    - best practices 132
    - Class-Path entry 374
    - duplicate entries 128
    - executing a JAR 160
    - expanding 128
    - manifests 129
    - sealed JARs 130, 523
    - signing 132, 221
    - validating 129
    - WinZip problems 128
  - JAR manifests
    - adding signatures 135
    - bypassing Ant fixup 131
    - line length rules 131
    - sections 131
    - specification 130
  - jar program 517
  - <jar> task 126, 147, 216
    - adding metadata 131
    - best practices 132
    - bug reports 131
    - building an Antlib 478
    - compressing files 127
    - dependency logic 128
    - dependency rules 34
    - duplicate attribute 128, 132
    - <ear>, comparison to 377
    - handling duplicate files 128
    - manifest attribute 130
    - manifest creation 217
    - <manifest> element 130
    - manifests 127
    - <metainf> fileset 131
    - packaging Enterprise beans 371
    - specifying the output file 127
    - update attribute and signatures 135
    - WAR files 328
  - jasigner program 134
  - Java
    - application release process 110
    - built-in XML support 341
    - dependency checking. *See* <depend> task
    - executing. *See* <java> task
    - installing the JDK 516
    - package names 28
    - RMI 417, 421–422
  - Java 6 133
    - Ant features 489
  - Java API 446
    - Ant's main classes 446
    - Ant's security manager 475
    - AntClassLoader 447, 472, 474, 479
    - API of Ant tasks 471
    - binding a Task to a Project 450
    - BuildEvent 505
    - BuildEvent class 504

- Java API (*continued*)
  - BuildException 445–446, 451, 453, 458–459, 467, 470, 476
  - BuildListener 503–505
  - BuildLogger 449, 503–505, 509
  - classes and classloaders 472
  - classpaths 472
  - comparing file timestamps 452
  - copying resources 452
  - creating a classloader 447
  - creating a datatype 448
  - creating a task 447, 471, 475
  - datatype references 447, 495–496
  - DateUtils class 451
  - embedded Ant 512
  - exceptions 451
  - executing targets 448–449
  - expanding properties 448
  - files
    - closing robustly 452
    - copying 452
    - resolving 452
  - FileUtils class 452
  - getting a project property 448
  - InputHandler 512
  - JavaEnvUtils class 451
  - logging 444, 448–449
  - mapper interface 500
  - Path class 470, 472, 474
  - paths 472
  - Project class 447
  - ProjectComponent base class 446, 449
  - Project.setNewProperty 485
  - resolving a datatype reference 448
  - Resource class hierarchy 468
  - Resources 467, 493
  - ResourceUtils class 452
  - setting properties 448, 485–486, 489
  - StringUtils class 452
  - supporting datatype references 473
  - Target class 449
  - Task class 450, 453
  - touchable resources 497
  - Union resource 466
  - utility classes 451
- <java> class implementation 475
- Java EE 363
  - application.xml 374, 376, 382
  - classloaders 374
  - compiling beans 368
  - database binding 366
  - dependency injection 370, 372
  - differences between implementations 374–375
  - EAR files 373
  - Enterprise Java Beans 365
  - Entity Beans 365–366
  - installing 364
  - Java Persistence API 366
  - JSP tag libraries, and 327
  - Message-Driven Bean 365
  - packaging 371–373, 385
  - persistence.xml 371
  - session beans 365, 369–372
  - testing. *See* Cactus
  - troubleshooting 384
  - web application integration 371
  - web services support 369
- Java Persistence API. *See* Java EE
- Java program 519, 523
  - entry point 150
  - integrate with Ant 177
  - with sealed JARs 130
  - with signed JARs 135
- Java SE Software Development Kit 516
- <java> task 37, 151
  - <arg> element 153
  - arguments 153
  - <assertions> element 105, 218
  - best practices 177
  - building multiple source trees 53
  - capturing results 160
  - classname attribute 151
  - classpath inheritance 153
  - classpath setup 152
  - classpathref attribute 152
  - creating in another task 447, 471, 475
  - D arguments 155
  - debugging 176
  - debugging problems 158
  - debugging under NetBeans 541
  - environment variables 167
  - error attribute 170
  - error messages 151
  - errorproperty attribute 170
  - executing JAR files 160
  - failonerror attribute 159–160
  - file arguments 154, 158
  - forking the JVM 156
  - handling errors 158
  - I/O 170
  - I/O redirector 171
  - IDE and 532
  - implementation 176
  - input attribute 170
  - inputstring attribute 170
  - JVM tuning 157
  - line arguments 155
  - output 38
  - output attribute 170
  - output on Linux 38
  - outputProperty attribute 170
  - path arguments 154
  - pathref arguments 154
  - <permission> element 176
  - post-processing output 172
  - resultproperty attribute 160
  - running Ant embedded 514
  - running in the background 169
  - sealed JARs and 130
  - setting system properties 155
  - setting the entry point 151
  - signed JARs and 135
  - spawn attribute 169
  - src elements 53
  - <sysproperty> element 104, 155, 218
  - timeout attribute 168
- JAVA\_HOME environment variable 521

- javac program 519, 521
    - command-line switches 50
    - srcdir 51
  - <javac> task 21, 23, 49, 77, 216
    - attributes 50
    - bootclasspath element 50
    - classpath element 50
    - clean builds 121
    - compiling code 361, 368
    - compiling Java EE code 369
    - compiling web applications 324
    - debug attribute 50, 216
    - default action 23
    - dependency checking and directory structure 28–29
    - dependency logic 26, 241
    - dependency rules 361
    - destdir attribute 50
    - encoding attribute 50
    - extdirs element 50
    - implementation 471
    - nowarn attribute 50
    - output 23
    - setting up classpaths 73, 75, 214
    - source attribute 50
    - src element 50
    - srcdir attribute 51, 361
    - target attribute 50
    - troubleshooting 521
    - verbose attribute 50
  - <javacc> task 235
  - <javadoc> task 122, 218
    - retrofitting dependency logic 258
    - setting up source 122
    - troubleshooting 521
  - <javah> task 235
  - javap 176
  - <javaresource> resource 145
  - <jbjar> task 235
  - JBoss 364
    - deploy-by-copy 330, 339
    - deploying web applications 331
    - session bean injection, and 372
    - undeployment 331, 435
  - <jdepend> task 235
  - <jing> task
    - downloading with Ivy 350
    - third-party 344, 362
    - validating Atom feeds 351
  - JNDI 372, 385
  - JPA. *See* Java EE
  - JPackage 518
  - JSP Documents, Files, Pages 326
  - <jspc> task 235
  - JUnit 416
    - Ant integration 90
    - architecture 84–85
    - AssertionFailedError 87, 97
    - assertions 87, 97
    - Cactus test cases 379
    - compiling test cases 95
    - fetching with Ivy 300, 304, 307, 312
    - running tests
      - from the command line 86
      - in a Swing GUI 86
      - under Ant 93, 217
    - test case lifecycle 89
    - test runners 86
    - TestCase class 86
    - testing against signed JARs 135
    - testing inside an application server 379
    - testing web pages 334–335
    - versions 84, 107
    - writing a test case 86
  - JUnit 4 384
  - <junit> task 217–218, 234–235
    - <assertions> element 105, 218
    - <batchtest> attribute 98, 218
    - <batchtest> attribute needs standard test names 227
    - creating HTML reports 99–100, 218
    - dependency logic 100
    - enabling Java assertions 105
    - errorProperty attribute 101
    - failureProperty attribute 101
    - forking behavior 104
    - forking the JVM 103
    - forkMode attribute 104
    - functional testing 437
  - halting the build 96, 101
  - haltonfailure 96
  - haltOnFailure attribute 96, 99–101, 218
  - implementation 156
  - limitations 100
  - NetBeans integration 540
  - passing properties 104, 218, 335, 338
  - printing results to the console 97
  - printsummary 96
  - publishing the results 397, 401
  - result analysis and formatters 96
  - running a single test case 103
  - running Cactus tests 383
  - skipping tests if the deployment failed 385
  - support for JUnit 4 85, 107
  - <sysproperty> attribute 338
  - <sysproperty> element 104, 200, 218, 383
  - testing against signed JARs 135
  - timeout attribute 401
  - viewing program output 98
  - XML formatter 99, 102, 117
  - <junitreport> task 99–102, 217–218, 234–235, 353
    - customizing reports 106
    - limitations 100
  - JVM forking 103
  - python indentation rules 486
- K**
- k. *See* -keep
  - Kaffe 518
  - keep 39, 42
- L**
- <length> condition 68
  - lib 91
    - classpath implications 152
    - loading an Antlib 262–263
    - loading custom tasks 480

- library management 75, 265
  - adding the compiled source to the classpath 152
  - fetch.xml 91
  - JAR files 75
  - setting up classpaths 214
- line endings, adjusting for platforms 124
- listener 40, 507
- listeners and loggers, using 505, 511
- <listmodules> task (Ivy) 302
- <loadfile> task 191
- <loadproperties> task 184, 193
- <loadresource> task
  - using a custom resource 496–497
- log() methods 449
- Log4J
  - configuration 416, 425
  - fetching with Ivy 300, 304–307
- <logcontains> condition (AntUnit) 459
- logfile 40
- logger 40, 505, 510, 520
- loggers
  - best practices for tasks 449
  - writing custom 509
- Luntdownload 391, 401–402
  - binding to a repository 395
- builder
  - Ant and 397
  - creating 396
  - definition 392
  - timeouts 401–402
- check the jetty address 393
- configuring 394
- creating a project 395
- installing 393
- manual trigger 400
- project 392, 395
- Proxy set-up 393
- running 393
- schedule 392, 398
- user 392
  - creating 394
- User setup 399

- VCS Adaptor 392
  - configuring 395

## M

- <macrodef> task 291, 296, 323
  - Antlibs and 249, 479, 481
  - AntUnit and 459
  - attributes with defaults are
    - optional 291
  - attributes, adding 292, 294
  - backtrace attribute 481
  - best practices 295
  - elements, adding 292
  - implementation 444
  - implicit elements, adding 293
  - local variables with
    - ant-contrib's <var> task 294
  - namespaces 292, 295
  - redefinition rules 290
  - supporting nested text 292
  - uri attribute 292, 295
- macros, using <presetdef> 288
- <mail> task 180, 188, 222, 391
  - gmail servers 190
  - library dependencies 181, 188
  - message element 189, 192
  - messagefile attribute 189, 192
  - messagemimetype attribute 189, 191
  - sending messages 189, 191
  - storing recipients in a file 191
  - with missing libraries 238
- main 40, 512
- Make 14
  - comparison to Ant 15
- <manifest> task 129
- mappers 114
  - <apply> task 176
  - chainedmapper 118
  - compositemapper 118
  - custom mapper 499
  - flatten mapper 118
  - flattenmapper 116
  - globmapper 116, 118, 176
  - identity mapper 118
  - identitymapper 115
  - merge 118

- <outofdate> 258
- packagemapper 117
- propertysets 156
- regexpmapper 116
- script mapper 119
- scripting a mapper 500
- master build
  - bulk delegation with <subant> 275
  - configuring child projects 272
  - controlling child projects 270
  - getting data back 276
- <matches> condition 68
- Maven 16, 298
  - comparison with Ant 16
  - conformance over configuration 298
  - limitations 298, 317
  - POM file 16, 298, 300
  - standard project configurations 311
- md5sum program 182
- Merge mapper 118
- Message-Driven Bean 365
- <mkdir> task 29–30, 111
  - directories 34
  - outside targets 215
- <move> task 114
  - filtering files 119
  - mappers 114–115
- moving files 114
- MSBuild 6
- MySQL 401, 411
  - JDBC driver 411–412, 417–418

## N

- <native2ascii> task 235
- <nbjpdastart> target (NetBeans) 541
- NetBeans 539
  - binding to an Ant project 540
  - configuring Ant 542
  - debugging a build 540
  - JUnit integration 540
  - managing Ant's classpath 543

NetBeans (*continued*)  
  selecting targets 541  
  SmartFrog plugin 426  
<netrexcc> task 235  
-nice 40  
NoBannerLogger 520  
-noclasspath 40, 91  
  classpath implications 152  
<not> condition 68  
-nouserlib 40, 91

## O

optional tasks 234–235, 239  
  best practices 263  
  custom Ant builds may be  
  incomplete 238  
  how Ant reports a missing  
  dependency 236  
  installation 236  
  library version problems  
  238  
  missing implementation classes  
  238  
  troubleshooting 238  
<os> condition 68, 165  
  confused by new operating sys-  
  tems 166  
<outofdate> task (ant-contrib)  
  257

## P

package mapper 117  
packaging 110  
  Ant tasks for working with Zip  
  and tar files 126  
  bzip2 files 141  
  deb files 139  
  gzip files 141  
  install scripts 124  
  JAR files 126  
  JPackage project 143  
  rpm files 139  
  RPM packages 143  
  tar files 139  
  Unix formats 139  
  Zip files 136

<parallel> task 424  
  listening to 508  
  thread-safety 511  
<parsersupports> condition 68  
path 52  
  cross-platform handling 52  
  id attribute 53  
  location attribute 53  
  path elements 51  
  referencing in other paths 73  
  refid attribute 53  
path datatype 152  
  setting up a classpath 298  
  using with Ivy 307  
<path> datatype, setting up a class-  
  path 75  
PATH environment variable  
  516–519  
<path> resource collection 145,  
  147  
paths datatype, setting up a class-  
  path 306  
patternset 54  
  attributes 55  
  conditional patterns 72  
  elements 55  
  if and unless attributes 72  
  nested patternsets 54  
  nesting using references 74  
  precedence 55  
Perforce 243  
Perl, why XSD is like 347  
persistence.xml 371, 374  
<presetdef> task 288, 296, 314,  
  323  
  Antlibs, and 249, 479, 481  
  best practices 290  
  database administration 412  
  extending another preset task  
  289  
  fixing <exec> 288  
  hazards 290  
  implementation 444  
  limitations 291  
  namespaces 289  
  predefining an XML validation  
  task 349  
  redefinition rules 290  
  renaming an existing task can  
  break old code 447, 450  
  uri attribute 289  
<project> 21  
  basedir 61  
  default target 33  
project 7–8  
<project> element 525  
-projecthelp 40, 42, 210  
  how to avoid having side effects  
  215  
properties 9, 48, 61  
  accessing inside a logger  
  509–511  
  accessing inside a task 448  
  best practices 76  
  built-in properties 61  
  controlling Ant 70  
  copying with <propertycopy>  
  253  
  enabling and disabling targets  
  71  
  expansion 61  
  immutability rule 49, 64, 294  
  implementation 448  
  listing 62  
  loading environment variables  
  65  
  loading from files 63  
  overriding 64  
  passing down through <ant>  
  270  
  property inheritance across  
  projects 271  
  relation to datatype references  
  73  
  setting 62  
  setting in a task 448, 485–486,  
  489  
  setting to a filename 63  
  setting value 62  
  undefined properties 66  
  viewing in an IDE 533  
property file  
  build.properties file 212–213  
  creating property files. *See*  
  <propertyfile> task  
  escaping windows paths 185

- property file (*continued*)
  - loading with <loadproperties> 184
  - making distribution flexible 187
  - storing build information in applications 240
- <property> resource 145
- <property> task
  - env attribute 166
  - environment attribute 65
  - file attribute 63, 212, 274, 285
  - loading passwords 133
  - location attribute 63, 77
  - refid attribute 314
  - using <loadproperties> 184
  - value attribute 48, 63
- <propertycopy> task (ant-contrib) 253
- propertyfile 10, 40, 70
- <propertyfile> task 235, 239
  - comments get stripped 241
  - date entry 240
  - entry operation attribute 241
  - int entry 240
  - string entry 240
- proxies, setting up the continuous integration tool 393
- proxy setup 39
- <publish> task (Ivy) 302, 308–310, 312, 318
- PVCS 243
- PXE Preboot environment 409–410

## Q

- q. *See* -quiet
- quiet 40–41, 449, 505

## R

- <record> task 511
- references 73
- refid attribute 73
- Regexp mapper 116
- <replaceregexp> task 235

- <report> task (Ivy) 302, 305–306, 312
- <repreport> task (Ivy) 302, 315
- <resolve> task (Ivy) 302, 304, 311
  - reporting failure 368
- resolveFile() 452
- resource collections 144–145
- <resource> resource 144
- resource using in a custom task 466
- <resourcecount> condition 68
- resources 76, 143
  - resource collections 145
  - resource-enabled task 144
  - touchable resources 144
  - using 146
- <resources> resource collection 145
- <resourcesmatch> condition 68
- <restrict> resource collection 146
- <retrieve> task (Ivy) 302, 306
  - sync attribute 307
- <rexec> task 181, 235
- <rmic> task implementation 471
- Rome API 322
  - library dependencies 325
  - packaging in a WAR 325
  - using in a servlet 323
- <rpm> task 143, 235, 409
- <runservertests> task (Cactus) 381

## S

- Schematron 344
- <schemavalidate> task 235, 344, 347–348, 362
  - noNamespaceFile attribute 348
  - output in -verbose mode 348
  - schema element 349
  - validating namespaced XML 349
  - validating WSDL files 349
- SCM 243, 395
- <scp> task 181, 201, 222–223, 235
  - creating a remote path 194
  - dependency logic 195

- downloads 195
- jsch incompatibilities 181
- keyfile attribute 194, 196, 198
- library dependencies 181, 193
- localToFile attribute 196
- passphrase attribute 194, 196, 198
- remoteFile attribute 196
- remoteToDir attribute 194, 196
- troubleshooting 197
- uploads 193
- script
  - bean scripting framework 484
  - beanshell 484–485, 499–500
  - defining new tasks with
    - <scriptdef> 486
    - inline script 485
    - Java 6 support 489
    - Java1.6 support 484
    - JavaScript 489
    - python 484–486
    - python 484, 486
    - rhino 484
    - ruby 484
    - running BSF-based scripts 484
    - scripted filter reader 503
    - scripted mappers 500
    - scripted selectors 499
    - supported languages 484
    - TCL 484
  - script mapper 119
  - <script> task 235, 485, 515
    - alternatives 486
    - implicit objects 485
    - language attribute 484–485, 489
    - manager attribute 489
    - testing 485
  - <scriptcondition> condition 68, 492
  - <scriptdef> task 235, 515
    - attributes and 486
    - dynamically adding listeners 511
    - handling nested elements 488
    - implementation 444
    - language attribute 486–487

- <scriptdef> task (*continued*)
    - nested elements 487
    - raising exceptions 486–487
    - testing with AntUnit 487–488
    - use in an Antlib 479, 488
  - <scriptdef> tasks, redistributing 488
  - <scriptfilter> filter 172
  - <scriptfilter> type 503
  - scripting
    - jruby 487
    - scripting a condition 492
    - within Ant 484
  - <scriptmapper> mapper 500
  - <scriptselector> selector 499
  - SDK, installing 516
  - selectors 58
    - defining a custom selector 498
    - examples 59
    - selector containers 58
    - writing a custom selector 498
  - <sequential> task 291
    - appearance inside <foreach> 257
  - servlets
    - JSP engine 326
    - publishing an Atom feed 323
    - servlet API is needed at compile time 325
  - session bean 365, 369
    - compilation 370
    - dependency injection failure 384
    - testing 370
    - using in a web application 371
  - <setproxy> task 235
  - sha1sum program 182
  - <signjar> task 134, 147, 220–221
    - implementation 471
  - <sleep> task 333
  - SmartFrog 415, 439
    - components 417
    - composing components 419
    - concepts 416
    - daemon 417, 421, 431–433
    - deploying 433
    - deploying across multiple machines 421–422
    - distribution 422
    - implementation 417
    - introduction 415
    - JAR file downloads 418
    - LAZY references 418–419, 422, 431
    - passing down properties 431
    - ping 417, 423, 427, 434
    - preflight validation 430
    - PROPERTY references 421–422, 432
    - sfProcessHost attribute 421
    - TBD declarations 419–420, 430
    - terminating an application 435
    - types 418
    - undeployment 435
  - <socket> condition 68, 199, 331
    - probing RMI ports 432
  - <socket> task 339
  - Software Configuration Management. *See* SCM
  - <sort> resource collection 146
  - <sound> task 235
  - SourceOffSite 243
  - SourceSafe 243
  - spawning Java and native programs 169
  - <splash> task 235
  - <sql> task 412–414, 425, 439
    - expandProperties attribute 413
    - implementation 477
    - loading SQL from a file 414
    - onerror attribute 413
    - src attribute 414
  - SSH
    - authorized\_keys file 193, 198
    - creating directories before <scp> uploads 197
    - downloading files with <scp> 195
    - executing remote commands 197
    - key generation 192
    - known\_hosts file 198
    - OpenSSH 192
    - PuTTY 192, 198
    - server setup 193
    - troubleshooting 197
    - uploading files with <scp> 193
  - SSH. *See* <scp> and <sshexec> tasks
  - <sshexec> task 181, 197, 223, 235
    - creating directories 197
    - keyfile attribute 197–198
    - library dependencies 181, 193
    - passphrase attribute 197–198
    - troubleshooting 197
  - StarTeam 243
  - <stringentry> resource 145
  - <subant> task 275, 296, 301, 312
    - calling default targets 275
    - ignoring failures 275
    - implementation 448
    - inheritall attribute 275
    - Ivy integration 276
    - limitations 313
    - loops 314
    - preset and macro tasks are always inherited 290
    - setting up with Ivy 313
  - Subversion 245
  - <switch> task (ant-contrib) 248, 254
  - system tests 84
- ## T
- tar files 139, 141
    - compressing 141
    - incompatibilities with SYSV tar 523
    - untarring 142
  - tar program 519
    - long filenames 141
  - <tar> task
    - file permissions 134
    - long file support 141
    - <tarfileset> element 140
    - using <apply> for the native version 174
  - <tareentry> resource 145
  - <tarfileset> datatype 140
  - <tarfileset> resource collectione 145
  - <target> 22, 32

- targets 7
  - calling targets with `<antcall>` 203
  - circular dependencies 33
  - conditional 71, 166
  - default 22
  - dependencies 7, 32
  - dependency order 36
  - deriving from deliverables 226
  - description attribute 42–43
  - if/unless attributes 71
  - listing 42
  - naming rules 37
- Task
  - `bindToOwner()` method 447, 449–450, 475
  - `execute()` method 444–445, 450, 453, 463, 465–466, 471, 475–477
  - `init()` method 445, 450
- task 7, 22
  - arguments 38
  - attributes, Boolean values 51
  - execution 33
- `<taskdef>` task 246, 350, 458
  - Antlib, use in 479
  - declaring a single task 247
  - error handling 253
  - loaderref attribute 247
  - redefining a task 248
  - resource attribute 247
  - uri attribute 249
- `<taskfound>` condition 237
- tasks 234
  - categories of tasks 90
  - core tasks 234
  - custom tasks 234
  - library dependencies 91, 181
  - optional tasks 234
  - third-party tasks 234–235
- `<telnet>` task 180, 235
  - library dependencies 181
- `<tempfile>` task 196
- testing 79–80
  - acceptance tests 84
  - application portability 80
  - best practices 106
  - Cactus. *See* Cactus
  - configuring test runs 104, 335, 338
  - creating HTML reports 99–100, 218, 438
  - cross-platform 403
  - DbUnit 108
  - diagnosing causes of test failures** 87, 96, 98
  - final JAR files 135
  - functional testing 321, 437
  - functional tests 84
  - halting the build on failure 96, 101, 218
  - HttpUnit 108, 334
  - in the build 94, 217
  - JAR files 128
  - Java EE. *See* Cactus
  - listeners and loggers 507
  - main class 83
  - model-view architectures simplify testing 226
  - planning 226
  - regression testing 80, 84, 97
  - replicating bugs 80
  - `<script>` tasks in ant 485
  - source distribution 138
  - system tests 84
  - testing custom tasks. *See* AntUnit
  - TestNG 108
  - to enable refactoring 81
  - under SmartFrog 437
  - unit tests 84
  - versus proofs of correctness 80
  - ways to test 83
  - web applications 333
  - XmlUnit 108
- TestNG 108, 384
- third-party tasks 234–235, 245
  - best practices 263
  - declaring
    - Antlib 249, 479
    - into a new XML namespace 248
    - through property files 247, 478
    - with `<taskdef>` 246
  - defining 246
  - installing 246
  - loading tasks and types in the same classloader 247, 249
  - locating third-party 246
  - redefining 248
- tnameserv program 519
- `<tokenfilter>` filter 172
- `<tokens>` resource collection 146
- Tomcat 329
  - deploy-by-copy 330, 339
  - JSP support 326
  - undeployment 331
- toString
  - operation 74
  - printing classpaths 153
- `<touch>` task
  - custom resources 497
  - resource support 144
- `<translate>` task 235
- `translatePath()` 474
- troubleshooting 520
  - Ant not found 521
  - ANT\_HOME 522
  - ANT\_OPTS 523
  - CLASSPATH 522
  - JAVA\_HOME 521
  - JDK not installed 521
  - missing task 522
  - multiple Ant versions 522
  - sealing violation 523
  - Unix 523
- `<trycatch>` task (ant-contrib) 255
  - behavior under a custom listener 509
- `<tstamp>` task 69, 119, 215, 218
  - day of the week 69
  - ISO 8601 format 69, 120
- `<typedef>` task 246, 250, 490
  - declaring
    - condition 491
    - custom resource 496
    - custom selector 498
    - custom task 444, 455, 479
    - filter 502
    - mapper 500
  - defining Ant types 490
  - loaderref attribute 247

<typedef> task (*continued*)  
  loading an Antlib 479–480, 491  
  onerror attribute 480, 491  
  resource attribute 247  
  uri attribute 249, 480  
typedef task 250, 253  
<typefound> condition 68, 237

## U

<undeploy> task (SmartFrog) 435  
undeployment 331, 435  
  waiting for undeployment to complete 333  
<unions> resource collection 146  
Unix  
  installing Ant 518  
  line endings 125  
<unjar> task 128–129, 217  
  dependency rules 129  
unpackage mapper  
  mapping directoriesJava packages 117  
<untar> task 142  
<unwar> task 128  
<unzip> task 128, 138, 221  
  dependency rules 129  
<uptodate> condition 68  
<uptodate> task 118, 257  
  mappers 114  
<url> resource 145  
user input 132

## V

-v. *See* -verbose  
<var> task (ant-contrib) 294–295  
-verbose 25–26, 34, 40–41, 133, 292, 449, 505  
<apply> task 175  
debugging 35  
debugging <java> 158  
migrating to Ant 229  
<schemavalidate> output 348  
<xmlvalidate> output 346  
-version 20, 40, 518–519  
Version Control. *See* SCM

Virtualization 390, 408, 416  
VMWare 390, 403, 408, 416

## W

<waitfor> task 199, 258, 332, 339  
  attributes 332  
  EAR files take longer 378  
  waiting for a web page 332–333  
  waiting for an undeployment 333  
  *See also* <faultingwaitfor>  
WAR file 320  
  adding  
    classes 328  
    content 327–328  
    libraries 328  
    metadata 328  
  building with the <war> task 328  
  deployment 329, 406, 435  
  libraries in a Java EE deployment 373  
  moving to Java EE 371  
  web.xml file 321  
  *See also* web.xml file  
WAR files 126  
  patching with Cactus 382  
<war> task 328, 339, 373  
  dependency logic 128  
  duplicate attribute 329  
  needxmlfile attribute 328  
  webxml attribute 328  
web applications 320  
  building 321  
  content 325  
  deployment 329, 406, 435  
  happy page 327, 409  
  library dependencies 321, 324  
  publishing an Atom feed with Rome 323  
  setting up the classpath with Ivy 325  
  testing 321, 333, 437  
  WAR files 320  
  workflow 321

web development 320  
web services, Java EE integration 369  
web.xml file 327  
<webxmlmerge> task (Cactus) 381  
Windows  
  checking environment variables 517  
  installing Ant 517  
  line endings 125  
  setting environment variables 517  
<wix> task (dotnet antlib) 409

## X

Xalan 341  
XDoclet (third-party tasks) 327  
Xen 390, 403, 408, 416  
Xerces 341  
  excluding from a project's dependencies 317  
XML  
  accessing as Ant properties 352  
  attributes 526  
  attributes in namespaces 529  
  binary data 527  
  CDATA 192, 342–343, 345  
  CDATA section 486, 527  
  comments 528  
  creating a constants.xml file 354  
  Data Type Definitions 344–345  
  DOM 334, 341  
  DTDs 342, 344–345, 354  
  elements 525–529  
  encoding 342–343, 528  
  entities 277  
  entity references 527  
  escaping characters 527  
  HTML output through XSLT 357  
  illegal comments 529  
  ill-formed XML 526, 529  
  Java source through XSLT 355

- XML (*continued*)
    - limitations of Ant's XML support 362
    - manipulation in Ant 340
    - namespace rules 529
    - namespaces 248–249, 289, 348, 352, 529
      - Ant support 530
      - Antlib URLs 481
      - attributes 529
      - declaring 249
      - default namespace 530
      - <echoxml> 342
      - prefix declaration 529
    - nested text 526
    - parser errors 522, 526
    - parser issues 341
    - parser problems 341
    - primer 525
    - prolog 525, 528
    - quotation characters in
      - attributes 163
    - reading XML 340, 352, 362
    - text 477
    - transforming with XSLT 353
    - transforming XML 340, 353, 362
    - TRaX 341
    - unsupported features 527–528
    - valid XML 343
    - validating 340, 343, 347, 362
      - in a namespace 348
      - with DTDs 345
      - with RelaxNG 349
      - with XML Schema 344, 347–348
        - WSDL files 349
        - well-formed 343–344
        - writing 340–342, 345, 362
        - writing with <echo> 342, 345
        - writing with <echoxml> 341
      - XPath 357
      - XSL engines 341
    - XML entities 277–278
    - XML namespaces. *See* XML, namespaces
    - XML parser. *See* XML, parser errors
    - XML Schema
      - namespace-targeted XML Schema 348
      - no-namespace XML Schema 347
        - xsd dateTime 69
    - <xmlproperty> task 352
    - <xmlvalidate> task 235, 344–346, 355, 362
      - dtd element 346
      - extension by <schemavalidate> 348
      - output in -verbose mode 346
      - verifying XML is well-formed 344
    - XML Catalog support 346
    - <xor> condition 68
    - XSD 344, 347
    - XSL
      - <junitreport> stylesheets 106
      - style sheets 355
      - See also* XML
  - XSLT. *See* XML
    - <xslt> task 353, 359, 362
      - creating HTML content 357
      - creating HTML output 357
      - creating Java source 357
      - dependency rules 359
      - processing CVS logs 244
- ## Z
- Zip files 136
    - best practices 139
    - creating a binary distribution 137
    - creating a source distribution 138
    - expanding 128
    - validating 129
    - WinZip problems 128
  - <zip> task 136, 221
    - best practices 139
    - dependency logic 128
    - duplicate attribute 128, 139
    - file permissions 134
    - handling duplicate files 128
    - <zipfileset> element 139
    - <zipentry> resource 145
    - <zipfileset> datatype 138, 221
      - extra attributes 136
      - setting the prefix 137
    - zipfileset datatype
      - building WAR files 328
    - <zipfileset> resource collection 145