

Symbols

:= assignment operator 35, 62
. signifying bundle 41
= assignment operator 35

A

AccessController.doPrivileged()
 442
Ace 321
activation policy 315–316
 using 316–317
activator 14
 creating for service 17
 dependency on
 org.osgi.framework 15
 implementing 15
addingService() 142
addRepository() 327
agent, definition 269
annotations, using for
 metadata 33
answer object 243
Ant, building with 207, 513–519
antlr.* 203
Apache
 Ace. *See* Ace
 Ant. *See* Ant
 Aries. *See* Aries
 Felix. *See* Felix
 Geronimo. *See* Geronimo
 ServiceMix 21
 Tuscany 22

Apache Commons Pool project
 example, bundling tests
 232–235
Apache CXF Distributed
 OSGi 495
Apache Felix Configuration
 Admin 308
Apache Felix HTTP Service 481
Apache Felix Log Service 481
Apache Felix Web Console 225
Apache Hadoop Zookeeper 500
API, public, and modularity 30
application
 defining extensibility
 mechanism for 8
 dividing into bundles
 216–221
 by areas of
 responsibility 216
 by modules 216
 lifecycle
 servlet 71
 standard Java 71
 packaging as logically
 independent JAR files 8
 services 221–225
 sharing different
 implementations in 221
 splitting into bundles
 206–229
assembly 23, 28
AtomicReference 364
attribute
 and dependency
 resolution 56–57
 equality matching 45

 in manifest file 35
 syntax 35
Auditor class 279
authentication, role-based
 model 152
authorization, role-based
 model 152

B

Beanome, and improving OSGi
 component model 351
BeanUtils
 imports, determining 200
 Javadoc 196
 packages 195
 use of Collections classes 196
behavior, expected, testing
 237–238
BIndex 324
 integrating into build
 cycle 326
binding method
 and Hollywood principle 368
 signature 359–361
 visibility of 361
binding, creating thread to run
 shell 78
bloat, and component
 orientation 349
Blueprint 21, 374–391
 <bean> element 375
 <blueprint> element 375
 <entry> element 375
 <interface> element 376

- Blueprint (*continued*)
 - <reference-list> element 379
 - <reference> element 379
 - <service> element 375
 - <type-converters>
 - element 389
 - activation attribute 385
 - activation callbacks 385
 - advanced features 387–391
 - architecture 374–375
 - auto-export attribute 376
 - availability attribute 381
 - BlueprintContainer
 - interface 391
 - Bundle-Blueprint header 377
 - Bundle-SymbolicName manifest header 384
 - component lifecycle
 - 382, 384–386
 - management of 382
 - component. *See* Blueprint component
 - consuming services with
 - 378–382
 - defining logical
 - component 377
 - environment manager 386
 - blueprintBundle 386
 - blueprintBundleContext 386
 - blueprintContainer 386
 - blueprintConverter 386
 - example usage 375–378
 - factory bean 376
 - grace period 383–385
 - timeout value 384
 - lazy vs. eager
 - initialization 385
 - manager values 387–388
 - mandatory service
 - dependencies 381
 - manifest header 377
 - metadata 390–391
 - interface hierarchy 390
 - optional service
 - dependencies 381
 - proxies and service
 - dynamism 379
 - proxy approach 378
 - reference listeners 380
 - method signatures 380–381
 - scopes 388
 - prototype 389
 - singleton 389
 - service attributes 376
 - service damping 383
 - timeout period 383
 - service interfaces 376
 - services references 378–380
 - services required to be
 - interfaces 377
 - specification, and Spring 374
 - type converters 389–390
 - use cases 374
 - vs. Declarative Services 374
 - XML syntax 375
- Blueprint component, container
 - managers 374
 - bean 374
 - environment 375
 - reference 374
 - reference list 375
 - service 375
- bnd 200–202
 - bndwrap task 233
 - build task 201
 - Include-Resource 200
 - instruction files 209–210
 - integrating with build 208
 - package instructions 201
 - Private-Package 200
 - pull approach to assembling
 - bundles 217
 - resource files 210
 - version ranges,
 - computing 219
- bnd tool 513
 - and brackets 517
 - directives 516–517
 - classpath 516
 - donotcopy 516
 - exportcontents 516
 - failok 516
 - include 516
 - manifest 516
 - nouses 516
 - output 516
 - plugin 516
 - removeheaders 516
 - sources 516
 - versionpolicy 516
 - headers 514–516
 - Export-Package 514
 - Import-Package 515
 - Include-Resource 515
 - Private-Package 515
 - Service-Component 515
 - instruction types 514
 - macros 517–518
 - split packages, mending 519
 - variables 517
 - version policy 518–519
- bndtools 526
- boot delegation 207
- brackets, and bnd tool 517
- brittle dependency, and export
 - attributes 160
- build, integrating bnd with 208
- bundle 31–34
 - accessing OSGi framework 15
 - accessing programmatically
 - through lifecycle layer 70
 - activation order 337–341
 - reasons to use 337
 - activator 77–79, 204–205
 - adding 77
 - declaring 215
 - exporting 195
 - need for 76
 - See also* activator
 - activator manifest entry
 - 76–77
 - ACTIVE, stopping 84
 - advantages over JAR files 10
 - and class visibility rules 33
 - and native libraries 185–187
 - and the environment
 - 183–187
 - as container of metadata 32
 - as deployment unit 32
 - as JAR file, or JAR file as
 - bundle? 49
 - as logical or physical
 - module 27
 - authenticating with User
 - Admin Service 152
 - benefit, measuring 226
 - building
 - with Ant 513–519
 - with bndtools 526
 - with Bundlor 527
 - with Maven 519–525
 - with Netisgo 527
 - with Osmorc 526
 - with PDE 526
 - with Sigil 526
 - with Tycho 527
 - BundleTracker class 483
 - chain of trust 459
 - changing its own state 99
 - class loader 54
 - class path 39, 41–42
 - flexibility 42
 - merging 179

- bundle (*continued*)
 - class search order
 - 165, 172, 179
 - class space 59
 - consistency 59–64
 - class-search order 48
 - configuration properties 86
 - configuration, initializing and updating 152
 - configuring 86–88, 299–314
 - configuration, creating 304–308
 - consumer 125
 - context 79–80
 - as unique execution context 80
 - core domain 459
 - customizer bundle.
 - See* customizer
 - debugging 259–271
 - declaring exported packages 10
 - declaring imported packages 10
 - definition 31
 - dependence on external code 44
 - dependencies, inspecting 109
 - dependency
 - declaring 171–173
 - issues with 176–177
 - dependency model 53
 - dependency resolution 53–59, 108–110
 - multiple matching package providers 57–59
 - Package Admin Service 108
 - refreshing 109
 - single package provider 55–57
 - deploying 88–92, 320–337
 - deploying with OBR 328–329
 - determining if a bundle is a fragment 425
 - determining module membership 32
 - determining which owns a class 109
 - determining which to install 422
 - distribution provider bundles 495
 - dividing code into 216–221
 - embedding JARs vs. importing 203
 - embedding source in 266
 - engine 218
 - execution context, accessing 70
 - export attributes
 - implicit 158–160
 - and brittle dependencies 160
 - mandatory 160–161
 - exported packages
 - choosing 195–199
 - determining which are required 218
 - versioning 198
 - exporting internal code 42
 - exports
 - duplicate 162–164
 - filtering 161–162
 - extender 101
 - extending access modifiers 10
 - extension 101
 - extension bundle 467
 - facade bundle 175
 - flexibility and 51
 - fragments 177–183
 - and Bundle-ClassPath 179
 - and component descriptions 357
 - and package private access 178
 - in class search order 179
 - merging metadata 178
 - using for localization 180–183
 - header 194
 - host bundle 177
 - identification
 - execution-time 81
 - reasons for many forms 82
 - identifier 81
 - identity, choosing 192–195
 - import packages, finding in manifest 201
 - imported packages
 - defining 199–203
 - optional 197, 202
 - versioning 202
 - importing a package it exports 155–158
 - importing external code 44
 - imports
 - dynamic 165–166
 - as bad practice 166
 - in class search order 165
 - dynamic vs. optional 166–167
 - logging example 167–171
 - loosening 164–171
 - optional 164–165
 - install command 88–89
 - installed, persistent cache of 76
 - installing into framework, in order to resolve 54
 - installing into OSGi framework 75, 84
 - inter- vs. intra-bundle dependencies 61
 - introduction 10
 - JAR file, as unit of deployment
 - for component 350–351
 - keeping compatible with standard JAR file 49
 - launcher 52
 - launching and embedding 413–421
 - lazy, activating 316
 - lifecycle 72–85
 - state diagram 83–84
 - lifecycle operations, defined in lifecycle layer 11
 - lifecycle operations, defining 70
 - lifecycle support 204–205
 - location 81
 - interpretation 81
 - logically encapsulating member classes 33
 - main bundle. *See* main bundle
 - management agent. *See* management agent
 - manifest file 32
 - syntax 34–35
 - mega 206–216
 - benefits 207
 - downside 216
 - jEdit example 207–215, 217–228
 - metadata 34–49, 81
 - bundle class path 39, 41–42
 - flexibility 42
 - bundle identification 34, 36–38
 - code visibility 34, 39–48
 - exported internal code 39, 42–44
 - human-readable information 34–36

- bundle (*continued*)
 - imported external code
 - 39, 44–46
 - purpose 34
 - minimizing exports and imports 216
 - multithreaded testing 241–243
 - no direct path from
 - INSTALLED to STARTING 83
 - package dependencies, narrowing 45
 - partitioning 115
 - permissions 151
 - advanced
 - management 465–471
 - asking the user 469–471
 - Condition interface 465
 - Condition.isPostponed() 468
 - Condition.isSatisfied() 468
 - custom conditions 465
 - date-based condition 466–467
 - granting based on digital signers 459
 - local permissions 464
 - postponed condition 468
 - user-input condition 467
 - preferences, distinct 313
 - previously saved state, handling 96
 - private data area, accessing 94
 - project information 194
 - protection domain 441
 - provider 125
 - providing services to and using services from 429–430
 - querying and setting start level 150
 - refreshing 110–114
 - required 171–177
 - and mutable exports 176
 - and ordering 176
 - and shadowing 176
 - in class search order 172
 - optional 172
 - package visibility 173
 - requiring execution environments 184–185
 - resolution 11
 - resolving 178
 - resolving for deployment with OBR 323
 - role in logical modularity 33–34
 - role in physical modularity 32–33
 - saving command history 95
 - services 221–225
 - sharing classes between 42
 - shouldn't try to update itself 91
 - signing digitally 457
 - BundleSignerCondition 461–463
 - certificates 458–461
 - distinguished name
 - matching 462–463
 - establishing trust 463
 - jarsigner tool 461
 - keystore 459
 - keytool command 459
 - org.osgi.framework.trust.repositories 463
 - terminology 458
 - specific, importing from 158
 - splitting application into 206–229
 - start command 89–90
 - start level 338–339
 - starting and stopping 15, 77, 79
 - starting lazily 314–317
 - state, persisting 93–96
 - stop command 90
 - storing data persistently 312
 - substitutable export 157
 - when to use 158
 - suicide 99–101
 - symbolic name 192–193
 - prefixing with domain 193
 - suffixes 193
 - system. *See* system bundle
 - term, vs. module 31
 - third-party domain 459
 - tracker 223
 - tracking 105
 - transition
 - from INSTALLED to RESOLVED 83
 - from INSTALLED to UNINSTALLED 84
 - from STARTING to ACTIVE 83
 - from STOPPED to RESOLVED 84
 - turning JARs into 192–206
 - cheat sheet 205
 - turning WAR into 207
 - uninstall command 91
 - update command 90–91
 - update inconsistencies 114
 - updated, and use of new classes 114–115
 - updating or uninstalling 110
 - uses constraints 59–64
 - using lifecycle API in 85–101
 - versioning 193–194, 293–299
 - policy 297
 - whether to create 226–229
- Bundle API,
 - getServicesInUse() 288
- bundle cache 84–85
 - impact of lifecycle operations on 85
- bundle event, types 98
- Bundle interface 80–82, 413
 - methods related to lifecycle management 80
- Bundle object, and bundle identification 81
- bundle test 251
- Bundle-ActivationPolicy 316
- Bundle-Activator 215
 - exporting 195
 - header 76
- Bundle-Category 36
- Bundle-ClassPath 207
 - .signifying bundle 41
 - and fragments 179
 - default value 41
- Bundle-ContactAddress 36
- Bundle-Copyright 36
- Bundle-Description 36, 194
- Bundle-DocURL 36
- Bundle-ManifestVersion 37, 194
- Bundle-Name 36, 194
 - not used for bundle identification 36
- Bundle-NativeCode 186
- Bundle-RequiredExecution-Environment 184
- bundle-symbolic-name 159
- Bundle-SymbolicName 36, 192
 - and Bundle-Version, as unique identifier 37
- Bundle-UpdateLocation 91
- Bundle-Vendor 36
- Bundle-Version 192–193
 - and Bundle-SymbolicName, as unique identifier 37

- bundle-version 159
 - Bundle.getBundleContext() 420
 - Bundle.getEntry() 101
 - Bundle.getHeaders() 101
 - Bundle.START_ACTIVATION_POLICY 317
 - Bundle.start() 85
 - result, depends on state of bundle 89
 - Bundle.stop()
 - result, depends on state of bundle 90
 - Bundle.uninstall() 85, 91
 - Bundle.update() 85, 91
 - BundleActivator 76–79
 - handling 78
 - start() 78
 - stop() 78
 - BundleActivator.start() 90
 - BundleContext 15, 79–80
 - accessing service registry 125
 - as unique execution context 80
 - methods 79
 - retrieving configuration properties 86
 - service-related methods 125
 - BundleContext.getAllServiceReferences() 430
 - BundleContext.getBundle() 88
 - BundleContext.getBundles() 93
 - BundleContext.getDataFile() 94–95
 - BundleContext.getProperty() 86, 350
 - BundleContext.getServiceReferences() 430
 - BundleContext.installBundle() 83–84, 89
 - BundleEvent 96
 - BundleEvent.INSTALLED 98
 - BundleEvent.RESOLVED 98
 - BundleEvent.STARTED 98
 - BundleEvent.STARTING 99
 - BundleEvent.STOPPED 98
 - BundleEvent.STOPPING 99
 - BundleEvent.UNINSTALLED 98
 - BundleEvent.UNRESOLVED 98
 - BundleEvent.UPDATED 98
 - BundleListener.bundleChanged() 98
 - BundleTracker 105
 - BundleTracker class 483
 - Bundlor 527
 - byte-code analysis tool, using to
 - determined imported packages 200
- C**
-
- capability 325
 - matching to requirement 325
 - cardinality
 - as dependency characteristic 361
 - of service dependency 354
 - roles in Declarative Services 361
 - certificate chain, definition of. *See* bundle, signing digitally
 - certificate, definition of. *See* bundle, signing digitally
 - childrenNames() 313
 - class
 - as member of a bundle 32
 - externally useful 42
 - non-useful 42
 - package-private. *See* package-private class
 - public, not necessarily externally visible 33
 - search order 48–49
 - sharing between bundles 42
 - sharing between modules 8
 - visibility rules, and bundles 33
 - class loader
 - defining class loader 280
 - for bundle 54
 - not intended as common tool 8
 - class loading
 - and TCCL 281
 - ClassNotFoundException vs. NoClassDefFoundError 272–274
 - issues, solving 271–282
 - visibility rules 278
 - class path
 - approach, lack of consistency checking 7
 - delegation 56
 - error prone 7
 - hell 7
 - purpose, with respect to modularity 40
 - vs. OSGi framework 75
 - class space 59
 - consistency 59–64
 - inconsistent 275
 - mapping 275
 - Class-Path header 220
 - Class-Responsibility-Collaboration (CRC) cards. *See* CRC cards
 - Class.forName(), avoiding 278–280
 - Class.forName(), considered harmful 279
 - Class.getName() 126
 - ClassCastException 274–275
 - ClassLoader.getResource() 416
 - ClassNotFoundException and third-party class loaders 273
 - avoiding, with OSGi 8
 - vs. NoClassDefFoundError 272–274
 - Cobertura 235
 - code sharing, with Export-Package and Import-Package 155
 - code visibility 33
 - in metadata 34, 39–48
 - cohesion 28
 - collaboration, maximizing potential for 58
 - com.sun.tools.* 203
 - compareTo() 138
 - compendium service. *See* service, compendium
 - component
 - and OSGi 349–355
 - building, in Declarative Services 356–357
 - bundle JAR file 350
 - composing application with 347
 - composite 22
 - composition 350
 - declarative 347
 - implicit 347
 - configuration, and component framework 354
 - dependency, declaring 347
 - description and bundle fragments 357
 - framework 350
 - interactions 349
 - interface, declaring 347
 - introduction to 346–348
 - lifecycle of 352
 - major, using a service between 123

- component (*continued*)
 - properties 358–359
 - configuring service dependency target filters with 363
 - priority of 358
 - sources of 358
 - provided services 353
 - reasons to use 348–349
 - required services 353
 - testing combinations 251
 - vs. module 347
 - vs. service 119
 - component bundle, lack of bundle activator 352
 - component framework
 - and managing component configuration 354
 - as execution environment 350
 - choosing 408–411
 - custom extension points 354
 - lifecycle hook 352
 - managing component lifecycle 352
 - managing service dependency 352
 - mixing and matching 410
 - paint program example 355
 - summary of features 408
 - vs. component model 346
 - component model
 - API-less 351
 - external management 348
 - separation of concerns 348
 - substitutability of providers 348
 - vs. component framework 346
 - component orientation
 - 346–349
 - bloat 349
 - diagnosis issues 349
 - potential issues 349
 - side-file syndrome 349
 - compose vs. control 72
 - concern 25
 - separation of. *See* separation of concerns
 - Conditional Permission Admin Service 151, 449–457
 - abstract conditions 449
 - allow- vs. deny-access decisions 454–455
 - BundleLocationCondition 451
 - BundleSignerCondition 451
 - ConditionalPermissionAdmin API 450–451
 - ConditionalPermission-Admin.newConditionalPermission(String) 456
 - ConditionalPermissionInfo 451
 - ConditionalPermissionInfo.getEncoded() 456
 - ConditionalPermissionUpdate.commit() 452
 - ConditionalPermissionUpdate.getConditionalPermissionInfos() 452
 - ConditionInfo 451
 - implementing a policy-file reader 456–457
 - mitigating performance cost 450
 - newConditionalPermissionUpdate() 452
 - PermissionInfo 451
 - using 452–455
 - @Configuration 247
 - configuration
 - and location binding 307
 - creating 304–308
 - flexibility, as benefit of modularity 64
 - verification, as benefit of modularity 64
 - Configuration Admin Service 152, 299–308
 - Configuration objects 300
 - configurations, creating 304
 - differentiating ManagedService vs. ManagedServiceFactory 303
 - factories 369
 - vs. component factories 371
 - ManagedService 300
 - ManagedServiceFactory 300
 - mix of non-OSGi and Pax Exam tests 250
 - configuration dictionary 302
 - Configuration object
 - creating 306
 - creating for managed service factory 307
 - interface 305
 - removing 307
 - removing for managed service factory 308
 - configuration properties 302
 - of Remote Services specification 498–499
 - configuration target 152
 - ConfigurationAdmin service
 - listConfigurations() method 306
 - ConfigurationAdmin service interface 304
 - ConfigurationAdmin.createFactoryConfiguration() 307
 - ConfigurationBaseTest 252
 - ConfigurationPlugin 152
 - ConfigurationTestBase 253
 - constraint 62
 - container
 - nested, using with TCCL 280
 - testing in as many as possible 245
 - contract. *See* service, contract
 - core service. *See* service, core
 - coupling 28
 - coupling, less, in service-oriented approach 120
 - CRC cards 119
 - createFactoryConfiguration() 307
 - customizer 141–142, 336
 - benefits of 143
 - DeploymentPackageCustomizer header 336
- ## D
-
- DA-Testing 245
 - features 246
 - dangling service 287–290
 - finding 287–288
 - protecting against 288–290
 - debugging
 - and HotSwap 266–271
 - with Eclipse 263–266
 - with jdb 261–263
 - Declarative Services 288, 355–371
 - and complexity 356
 - and extender pattern 356
 - and memory footprint 356
 - and startup time 356
 - callback methods 366
 - component description files 356

- Declarative Services (*continued*)
 - component factories 370–371
 - component lifecycle 364–371
 - callback method
 - signatures 366
 - coupled to bundle
 - lifecycle 365
 - stages of 364–366
 - ComponentContext 367
 - interface 367
 - locateService()
 - methods 368
 - components
 - building 356–357
 - immediate vs. delayed 369
 - configuration policies 368–369
 - event strategy 368
 - introduction to 355
 - lookup strategy 368
 - managing service
 - publication 355
 - providing services with 357–359
 - simple attribute types 358
 - use cases 374
 - vs. Blueprint 374
 - delegating service proxy 289–290
 - delegation 119
 - dependency
 - in service-oriented
 - approach 120–122
 - inter-bundle 61
 - intra-bundle 61
 - dependency injection
 - component based 222
 - vs. service discovery 122
 - dependency resolution 53–64
 - cascading 108
 - choosing between
 - candidates 58–59
 - Deployment Admin 330–337
 - and developing management agents 321
 - deployment package. *See* deployment package
 - focus 321
 - resource processor. *See* resource processor
 - session
 - creating 336
 - transactional aspects 336
 - deployment module. *See* module, physical
 - deployment package
 - creating 330–334
 - definition of 330
 - DeploymentPackage-Customizer header 336
 - fix package 332
 - greediness of 332
 - making tamperproof 331
 - managing 334–335
 - Resource-Processor
 - header 336
 - signing 333
 - structure of 331
 - usage strategies 331
 - deployment unit. *See* module, physical
 - DeploymentAdmin 330
 - DeploymentAdmin service, interface 334
 - DeploymentPackage-FixPack 333
 - DeploymentPackage-Missing 334
 - diagnosis, and component orientation 349
 - digital signing, definition of. *See also* bundle, signing digitally
 - directive
 - := assignment operator 35, 62
 - in manifest file 35
 - syntax 35
 - discoverRepositories() 327
 - discovery, definition of 499
 - distinguished name, definition of 460
 - See also* bundle, signing digitally
 - distributed computing, discovery 499
 - distribution provider
 - configuring for discovery 500
 - framework, implementing 502–510
 - publishing service remotely 496
 - distribution provider bundle 495
 - distribution, and component framework 348
 - documentation, embedding in bundles 266
 - Dynamic Java DA-Testing. *See* DA-Testing
 - DynamicImport-Package 165, 203
 - dynamism, inherent, in OSGi framework 76
-
- ## E
-
- EasyMock 239–240
 - Eclipse
 - bndtools 526
 - debugging with 263–266
 - Equinox 263
 - See also* Equinox
 - Gemini 374
 - HotSwap with 267–268
 - Memory Analyzer. *See* MAT
 - p2. *See* p2
 - Plug-in Development Environment (PDE) 526
 - tracing exception to bundle 265
 - EJB, and OSGi 19
 - Enterprise JavaBeans. *See* EJB
 - Equinox 263
 - as implementation of OSGi framework 9
 - diag command 277
 - underlying runtime for Eclipse IDE 9
 - error, execution time, avoiding with OSGi 8
 - event
 - listener, removing 98
 - listening for 96–99
 - types 96
 - Event Admin Service 151
 - exclude export filtering directive 161
 - execution context
 - of bundles 70
 - unique, BundleContext as 80
 - execution environment
 - OSGi standard names 184
 - requiring 184–185
 - execution-time dynamism, and service-oriented component models 351
 - export
 - attributes
 - implicit 158–160
 - and brittle dependencies 160
 - mandatory 160–161
 - duplicate 162–164

export (*continued*)
 filtering 161–162
 mutable, and required bundles 176
 substitutable 157
 when to use 158
 Export-Package 13, 155
 attributes 43
 default value 49
 definition 42
 include and exclude directives 161
 mandatory directive 161
 ExportedServiceTracker 504–508
 exporter, choosing between 58–59
 extender 101
 Extender Pattern 271
 extender pattern 101
 and Declarative Services 356
 and OSGi component frameworks 352
 and paint program 101
 extension 101

F

facade bundle 175
 factory service 146
 factory.pid 303
 Felix 225
 as implementation of OSGi framework 9
 as runtime for GlassFish 10
 Configuration Admin Service integration tests 252
 debug logging 277
 iPOJO. *See* iPOJO
 Metatype 311
 Sigil 526
 felix-cache 261
 forName(), vs. loadClass() 280
 fragment
 in class search order 179
See also bundle, fragments
 Fragment-Host 177
 framework
 as plugin mechanism 428
 configuration properties 417
 org.osgi.framework.boot-delegation 418
 org.osgi.framework.bundle.parent 418

org.osgi.framework.
 command.
 execpermission 419
 org.osgi.framework.library.
 extensions 419
 org.osgi.framework.start-level.beginning 418
 org.osgi.framework.storage 418
 org.osgi.framework.storage.
 clean 418
 org.osgi.framework.system.
 packages 418
 org.osgi.framework.system.
 packages.extra 418
 configuring 417–419
 configuring, creating, and starting 423–424
 creating 415–417
 embedded
 example 432–437
 interacting with 428
 who's in control 431–432
 embedding 427–437
 init() 419
 inside vs. outside 427–430
 installing bundles 424
 interacting with bundles 413
 launching 421–426
 determining which bundles to install 422
 vs. embedding 421
 managing with management agent 320
 null configuration 417
 publishing external service into 435
 querying and filtering on metadata 122
 reasons to create your own 413
 resolving bundles 178
 shutdown hook 422–423
 start() 419
 starting 419–420
 starting bundles 424–425
 starting main bundle 425–426
 starting with security enabled 471–475
 stop() 420
 stopping 420–421
 tracking service use 288
 using to record metadata 121
 waitForStop() 420
 waiting for shutdown 426

framework event, types 97
 framework factory, service instance, obtaining 416
 Framework interface 414–421
 framework launching and embedding API 413–414
 configuration properties 417
 org.osgi.framework.boot-delegation 418
 org.osgi.framework.bundle.parent 418
 org.osgi.framework.command.execpermission 419
 org.osgi.framework.library.extensions 419
 org.osgi.framework.start-level.beginning 418
 org.osgi.framework.storage 418
 org.osgi.framework.storage.
 clean 418
 org.osgi.framework.system.
 packages 418
 org.osgi.framework.system.
 packages.extra 418
 framework instance
 configuring 417–419
 creating 415–417
 init() 419
 start() 419
 starting 419–420
 stop() 420
 stopping 420–421
 waitForStop() 420
 launching vs. embedding 421
 org.osgi.framework.system.
 packages 430
 org.osgi.framework.system.
 packages.extra 430
 sharing packages with bundles 429
 framework resolution, vs. OBR 323
 Framework Service, registry hooks 506
 Framework.init() 419
 FrameworkEvent 96, 421
 FrameworkEvent.ERROR 98, 113
 FrameworkEvent.INFO 97
 FrameworkEvent.PACKAGES_REFRESHED 98
 FrameworkEvent.STARTED 97

FrameworkEvent.STARTLEVEL_CHANGED 98, 340
 FrameworkEvent.WARNING 98
 FrameworkFactory
 interface 416
 FrameworkFactory.newInstance() 416–417
 FrameworkListener.framework-Event() 97
 friend attribute 163

G

Geronimo, OSGi and 19
 getBundle() 109, 241
 getConfiguration() 306–307
 getExportedPackage() 109
 getFragments() 109
 getLocation() 307
 getMetaTypeInfoInformation() 311
 getPackageAdminService() 110, 131
 getRequiredBundles() 109
 getResolver() 328
 getService() 131
 counting calls compared to
 ungetService() 148
 getServiceReference 128
 getServiceReferences 129
 getServiceReferences()
 All* variant 147
 getServicesInUse() 288
 getSystemPreferences() 313
 getUserPreferences() 313
 getUsers() 313
 getUsingBundles() 288
 GlassFish
 Felix used for runtime 10
 OSGi and 19
 Google Guice peaberry 391
 granularity, modularity and
 object orientation 25

H

handshake procedure 242
 heap dump, analyzing 283–287
 heap, graphical view of, with
 MAT 285
 history command 94
 Hollywood principle, and bind-
 ing methods 368
 host bundle 177
 See also bundle, host bundle

HotSpot, PermGen heap 283
 HotSwap 266–271
 and adding methods 268
 OSGi way 269–271
 with Eclipse 267–268
 with JRebel 269
 HTTP Service 151
 HttpClient, wrapping as a
 bundle 204
 human-readable information, in
 metadata 34–36

I

IBM WebSphere. *See* WebSphere
 IDEA, Osmorc 526
 implementation
 sharing in a Java app 221
 support for multiple compet-
 ing, in service-oriented
 approach 120, 122–123
 import
 dynamic 165–166
 as bad practice 166
 in class search order 165
 dynamic vs. optional 166–167
 logging example 167–171
 loosening 164–171
 optional 164–165
 import keyword, vs. Import-
 Package 44
 Import-Package 13, 160
 and split packages 174
 and substitutable exports 157
 definition 44
 resolution directive 164
 version range 46
 vs. import keyword 44
 Import-Package header 218
 ImportedServiceFindHook 509
 ImportedServiceListenerHook
 508–509
 importing 44
 from a specific bundle 158
 include export filtering
 directive 161
 installDeploymentPackage()
 335
 integration test 251–254
 pattern 253
 intents 496–497
 and configuration 496
 hierarchy 497
 qualified, matching to
 configurations 502
 interface
 and encapsulation 121
 emphasis on, in service-
 oriented approach 120–121
 interface-based approach 12
 IoC container, and OSGi 21
 iPOJO 391–408
 @Bind 395
 @Component 394
 @Controller 403
 @Invalidate 402
 @Provide 394
 @Requires 398
 @ServiceController 403
 @ServiceProperty 394
 @Unbind 395
 @Validate 402
 bind/unbind method
 pairs 397
 bundle-context access 403
 coding service consumer
 defensively 397
 component
 @instantiate 405–406
 building 392–393
 Configuration Admin
 instance creation
 407–408
 description, canonical
 form 393
 factory service instance
 creation 406–407
 immediate 394
 instantiating 404–408
 instrumenting byte
 code 392
 lifecycle 400–404
 lifecycle participation 403
 service dependency 395
 validity of 400
 XML instance
 creation 404–405
 consuming services 395–400
 Factory interface 406
 factory.name service
 property 406
 field injection 398–400
 handlers 392
 lifecycle callback
 methods 402–403
 method injection 395–398
 null objects 399

iPOJO (*continued*)

- providing services 393–394
- proxies 395
- service dependency
 - binding policy 400–401
 - damping 402
 - dynamic 401
 - mandatory 401
 - optional 401
 - static 401
 - temporal 401–402
- vs. Blueprint 391
- vs. Declarative Services 391

J

JAR

- embedding vs. importing 203
- turning into bundles 192–206
 - cheat sheet 205

JAR file

- adding metadata to 13
- advantages of bundles over 10
- as bundle, or bundle as JAR file? 49
- bundle vs. normal 31
- converting to a module 15
- explicitly declaring packages contained in 13
- exposing classes in root-relative packages 40
- logically independent, packing application as 8
- metadata 31
- more than one providing a set of classes 7
- searching directories relative to root 40
- standard, handling 40
- with extra metadata, as bundle 10

Jar Jar Links 198

JARClassLoader 222

jarsigner tool 333

Java

- access modifiers 5
- app creation and the software lifecycle 71
- as both language and platform 27
- lack of advanced modularization support 3–8
 - compensating for 3
 - error-prone class path 7

- limited deployment and management support 7

- low-level code visibility control 5–6

package 5

- nested 6

- visibility 5

program, executing from class path 39

standard development, app lifecycle 71

using JAR files vs. using bundles 75

Java Authentication and Authorization Service (JAAS) 441

Java Business Integration. *See* JBIJava Debugger. *See* jdb

Java EE, and OSGi 19

Java Enterprise Edition. *See* Java EEJava Management Extensions. *See* JMX

Java Native Interface (JNI) 185

Java Preferences, similarity to Preferences Service 314

Java security architecture 440–444

- permissions 441

- privileged calls 442–444

- protection domains 441–442

java.lang.NullPointerException 265

java.security.Permission 440

java.security.ProtectionDomain 441

java.util.prefs.Preferences 314

java.util.ResourceBundles 177

javax.swing.* 203

JBI, and OSGi 21

JBoss

- and JMX 20

- OSGi and 19

jdb 261–263

jEdit

- changing to use dependency injection 222

- class loader 222

- JARClassLoader 228

- mega-bundle example 207–215, 217–228

- plugin framework, replacing with OSGi 222

- using services in 222

Jini, and OSGi 20

JMX

- and JBoss 20
- and OSGi 20

JOnAS, OSGi and 19

JRebel, HotSwap with 269

JSR 277

- and JSR 294 22

- and OSGi 21–22

JSR 291, “Dynamic Component Support for Java” 22

JSR 294, “Improved Modularity Support in the Java Programming Language” 22

JUnit 231

- bundling 233

JVM Tools Interface. *See* JVMTI

JVM, agents 269

JVMTI 288

K

keytool command 459

Kriens, Peter 200

- bnd tool 513

kXML 192

L

lazy delegation 260

LDAP, queries 129

library, state, adding to bundle lifecycle 204–205

lifecycle

- API 72

- introduction 77–83

- using in bundles 85–101

- management

- Bundle interface

- methods 80

- cache of installed

- bundles 76

- introduction 70–72

- reasons for 72

- of bundles. *See* bundle, lifecycle

- operations, impact on bundle cache 85

- OSGi framework's role in 75–76

- paint program,

- introducing 73–75

- software, phases 71

- state diagram 83–84

- lifecycle hook, in component frameworks 352
 - lifecycle impact, of service dependency 354
 - lifecycle layer 10–11, 14–16
 - and module layer 108–115
 - defining bundle lifecycle operations 11
 - programmatic access to bundles 70
 - purpose 70
 - lightweight container, and OSGi 21
 - LinkageError 276
 - listConfigurations() 306, 308
 - listDeploymentPackages() 335
 - listener 123
 - API for service events 136
 - benefits 140
 - fixing issues with 138–141
 - implementing ServiceListener 136
 - listing 123
 - receiving all matching service events 147
 - registering 137–138
 - listener pattern
 - implementing 123
 - services as substitute for 123
 - listRepositories() 327
 - loadClass(), vs. forName() 280
 - localization, with bundle fragments 180–183
 - location binding 307
 - Log Service 132–143
 - active, keeping track of 138
 - as compendium service 151
 - best, getting 140
 - decorating 142
 - tracking instances 141
 - logger, proxy 167–170
 - logging
 - and Log Service 132
 - example 167–171
 - threshold 252
 - logical boundary enforcement, as benefit of modularity 64
 - logical boundary, defined by module 25
 - logical module. *See* module, logical
 - LogService 239–240
- M**
-
- main bundle 423
 - invoking main class from 425
 - starting 425–426
 - Main-Class 39
 - major number 38
 - managed service
 - configuration, valid 309
 - implementing 301–302
 - metatype information 309
 - service.pid 300
 - managed service factory, implementing 302–304
 - ManagedService 300
 - ManagedServiceFactory 300
 - factory.pid 303
 - management agent 320–321
 - controlling bundle activation order 338
 - examples 320
 - managing running framework 320
 - using resource descriptions 322
 - management test 254–256
 - tasks 254
 - manifest file 32
 - attributes 35
 - directives 35
 - maintaining module metadata 33
 - syntax 34–35
 - manifest, finding import packages in 201
 - MAT 285
 - Maven
 - building with 519–525
 - dependencies, embedding 523–524
 - formatting issues 523
 - Tycho 527
 - Maven Integration 324
 - maven-bundle-plugin 207
 - artifacts
 - deploying to OBR 524–525
 - bundling non-JAR projects 525
 - configuration 521–523
 - debug logging 523
 - default translations 521
 - dependencies, embedding 523–524
 - exportcontents directive 524
 - headers
 - Embed-Dependency 523
 - Embed-Directory 524
 - Embed-StripGroup 524
 - Embed-StripVersion 524
 - Embed-Transitive 524
 - introduction to 519–521
 - mega bundle. *See* bundle, mega
 - memory leak 283–287
 - dangling service 287–290
 - finding by analyzing heap dump 283–287
 - Message class, accompanying Auditor class 279
 - META-INF/MANIFEST.MF 13, 32
 - META-INF/services 415
 - metadata 10
 - and annotations 33
 - bundle class path 39, 41–42
 - flexibility 42
 - bundle identification 34, 36–38
 - capturing fine-grained application information 122
 - code visibility 34, 39–48
 - defining bundle with 34–49
 - exported internal code 39, 42–44
 - filtering events for listener 123
 - human-readable information 34–36
 - imported external code 39, 44–46
 - placement 32–33
 - separate, benefits of 33
 - using to record semantics 121
 - version number 38
 - metatype 309
 - information, using 310–311
 - Metatype Service 299, 309–311
 - interface 310
 - MetaTypeInfo 309
 - MetaTypeProvider 309
 - method call, vs. service 118
 - method, binding. *See* binding, method
 - micro number 38
 - minor number 38
 - mock object 237–244
 - exposing race condition with 238
 - nice 241
 - replaying 241

- mock service. *See* service, mock
 - mocking 237–244
 - example 238–240
 - steps 238
 - testing expected
 - behavior 237–238
 - unexpected situations 240–241
 - modularity 25–27
 - and granularity 25
 - and public APIs 30
 - and purpose of class path 40
 - class-loading issues, solving 271–282
 - definition 4, 25
 - example, paint program 28–31, 51–52
 - benefits of 64–67
 - classes 29
 - JAR file 28
 - UI elements 29
 - improving memory consumption with 490
 - logical vs. physical 27, 177
 - managing application complexity 28
 - reasons for using 27–28
 - separation of concerns 25
 - vs. object orientation 25–27
 - Module 10
 - module 25
 - cohesion 28
 - coupling 28
 - defining logical boundary 25
 - logical
 - as bundle 27
 - bundle's role in 33–34
 - membership, determining with bundle 32
 - physical
 - aka deployment module or deployment unit 27
 - as bundle 27
 - bundle's role in 32–33
 - term, vs. bundle 31
 - versioning 293–294
 - vs. component 347
 - module layer 10–12
 - adding metadata to JAR file 13
 - and lifecycle layer 108–115
 - bundle, defining 10
 - preparing code for 13
 - module-level dependency, vs. package-level dependency 47
 - Module, definition 27
- N**
-
- name clash, avoiding using nested packages 6
 - native libraries, bundling 185–187
 - .NET
 - assembly 23, 28
 - and OSGi 22
 - net.sourceforge.cobertura
 - .datafile 236
 - NetBeans
 - and OSGi 20
 - Netisgo 527
 - Netisgo 527
 - Newton 22
 - and Nimble 321
 - nice mock. *See* mock object, nice
 - Nimble 321
 - NoClassDefFoundError
 - debugging 273
 - vs. ClassNotFoundException 272–274
 - node() 313
- O**
-
- object
 - answer. *See* answer object
 - mock. *See* mock object
 - object orientation
 - and granularity 25
 - separation of concerns 25
 - vs. modularity 25–27
 - objectClass 127
 - querying for services with 130
 - ObjectClassDefinition 309
 - ObjectWebJOnAS. *See* JOnAS
 - OBR 321–329
 - and development management agents 321
 - dependency deployment 322
 - dependency-resolution algorithm 323
 - deploying bundles 328–329
 - discovery 322
 - focus 321
 - repository, creating 323–326
 - RepositoryAdmin service 322
 - resolving bundles for deployment 323
 - simplicity of 322
 - vs. framework resolution 323
 - OBR repository 322
 - creating, with BIndex 324–326
 - describing resource and dependencies 322
 - federated structure of 322
 - relationships among entities 322
 - optionality, of service dependency 354
 - Oracle, GlassFish. *See* GlassFish
 - org.apache.tools.ant.* 203
 - org.osgi.framework, activator dependency on 15
 - org.osgi.framework.boot-delegation 235
 - org.osgi.framework.boot-delegation property 229
 - org.osgi.framework.Constants 87
 - org.osgi.framework.executionenvironment 185
 - org.osgi.framework.hooks.service.FindHook 506
 - org.osgi.framework.hooks.service.ListenerHook 506
 - org.osgi.framework.language 87
 - org.osgi.framework.os.name 87
 - org.osgi.framework.os.version 87
 - org.osgi.framework.processor 87
 - org.osgi.framework.system.packages.extra 229, 235
 - org.osgi.framework.vendor 87
 - org.osgi.framework.version 87
 - org.osgi.service.log 167
 - OSGi 4–8
 - activation policy 315–316 using 316–317
 - active start level 338–339
 - and components 349–355
 - and lightweight containers 21
 - and manifest file 34
 - main attributes 34
 - shorthand for duplicated directives and attributes 35
 - structure of attribute values 35

- OSGi (*continued*)
 - API, mocking
 - 237, 239–240, 244
 - architectural overview 9–12
 - as it relates to other
 - technologies 19–23
 - avoiding `ClassNotFoundException`
 - Exceptions with 8
 - avoiding execution-time errors with 8
 - benefits for web-application development 478
 - Blueprint specification 21
 - common pitfalls,
 - avoiding 133–136
 - Compendium services 529
 - Application Admin 530
 - Blueprint Container 530
 - Configuration Admin 529
 - Declarative Services 529
 - Deployment Admin 529
 - Device Access 529
 - DMT Admin 530
 - Event Admin 529
 - Foreign Application
 - Access 530
 - HTTP 529
 - Initial Provisioning 529
 - IO Connector 529
 - Log 529
 - Metatype 529
 - Monitor Admin 530
 - Preferences 529
 - UPnP 529
 - User Admin 529
 - Wire Admin 529
 - XML Parser 530
 - component model
 - bundle JAR file as component deployment unit 351
 - extender pattern 351
 - improving 351–354
 - vs. component framework 350
 - component, equated with module 350
 - components, dependency injection framework for 21
 - Conditional Permission
 - Admin Service Conditional Permission Admin Service Core services 528
 - Conditional Permission Admin 528
 - Package Admin 528
 - Permission Admin 528
 - Service Hooks 528
 - Start Level 528
 - URL Handlers 528
 - defining extensibility mechanism for application 8
 - dependency resolution 53–64
 - directives, defining 35
 - dynamic nature of 243
 - EJB and 19
 - Enterprise services 530
 - JDBC 530
 - JMX 530
 - JNDI 530
 - JPA 530
 - JTA 530
 - Remote Service Admin 530
 - Extender Pattern 271
 - file permissions 448
 - framework, integration
 - tests 200
 - heap dump 283
 - helping with Java's lack of modularity support 8
 - interface-based approach 12
 - Java EE and 19
 - JBI and 21
 - Jini and 20
 - JMX and 20
 - JNDI and 492
 - JSR 277 and 21–22
 - libraries are thread safe 79
 - migrating tests to 231–236
 - module layer, dynamic 350
 - native code support 186
 - .NET and 22
 - NetBeans and 20
 - no longer an acronym 9
 - package-level
 - dependency 47–48
 - packaging application as logically independent JAR files 8
 - permissions 444–448
 - AdminPermission 444, 446–447
 - BundlePermission 444–446
 - conditional 449–450
 - names and actions 444
 - PackagePermission 444–445
 - ServicePermission 444, 447–448
 - role in application servers 19
 - SCA and 22
 - security. *See* security
 - service-oriented component model. *See* service-oriented component model
 - service. *See* service
 - sharing classes between modules with 8
 - standard execution environment names 184
 - test tools 245–246
 - URL Handlers Service 492
 - version number
 - components 38
 - default 38
 - format 38
 - major number 38
 - micro number 38
 - minor number 38
 - qualifier 38
 - versioning 293–295
 - WAR files and 488–493
- OSGi Alliance, introduction 9
- OSGi Bundle Repository 199
- OSGi framework 9–12
 - and thread management 79
 - as dynamic execution environment 96
 - bundle dependency resolution 53–59
 - multiple matching package providers 57–59
 - single package provider 55–57
 - bundles, resolving 110
 - creating multiple implementation of 9
 - events, delivering
 - asynchronously 98
 - inherent dynamism 76
 - installing bundles into 84
 - introduction 9
 - kick-starting bundles 76
 - layers 10–18
 - lifecycle layer. *See* lifecycle layer
 - module layer 10–11
 - See also* module layer
 - service layer. *See* service layer
 - using in development 12
 - listening for events 96–99
 - properties 87
 - querying and setting start level 150

OSGi framework (*continued*)
 role in lifecycle 75–76
 services 149–152
 compendium 151–152
 core 150–151
 shells 73
 state, inspecting 92–93
 telling about bundle
 activator 15
 unregistering services 17
 vs. class path 75
 OSGi R4.2 Enterprise
 specification 488
 OSGi Service Platform,
 introduction 4, 9
 OSGi specification
 lifecycle API 72
 standard shell, lack of 73
 OSGi standard services,
 introduction 9
 OSGI-OPT 266
 osgi.framework.Service-
 Exception 501
 Osmorc 526

P

p2 321
 package 5
 export attributes
 implicit 158–160
 and brittle
 dependencies 160
 mandatory 160–161
 exported 10
 importing 155–158
 managing 155
 exported, managing 155–164
 exports
 duplicate 162–164
 filtering 161–162
 imported 10
 imports
 dynamic 165–166
 as bad practice 166
 in class search order 165
 vs. optional 166–167
 logging example 167–171
 loosening 164–171
 optional 164–165
 instances 59
 marking as optional 202
 multiple versions in same
 JVM 294

partial
 avoiding need for 198
 in third-party libraries 198
 private 200
 sharing with contained
 bundles 429
 split 171
 aggregating 173–176
 and completeness 176
 and ordering 176
 and restricted access 176
 best practices 175
 split, mending 519
 split. *See* split package
 superpackages 22
 triggering lazy activation
 with 317
 versioning 43, 293–299
 policy 295
 range 46
 visibility 5
 Package Admin Service 108, 150
 acquiring 131
 and update
 inconsistencies 114
 operations 109
 refreshing bundles 111
 resolving bundle
 dependencies 109
 package private access, and
 fragments 178
 package-level dependency
 47–48
 vs. module-level
 dependency 47
 package-private class 272
 PackageAdmin.refresh-
 Packages() 112, 115
 paint program
 and extender pattern 101
 bundle dependency
 resolution 55
 bundle identification 37–38
 bundle metadata 50
 classes 29
 creating different
 configurations 65
 extending dynamically
 101–107
 finalizing 50–53
 JAR file 28
 launching 52–53
 lifecycle, introducing 73–75
 modular, benefits of 64–67
 modularizing 28–31, 51–52

service
 defining 144
 publishing 144
 tracking 145
 UI elements 29
 using services in 143–146
 Paremus Nimble. *See* Nimble
 parent() 313
 pauseTestThread 242
 Pax Exam 245
 don't overuse 254
 features 246
 JUnit4TestRunner 247
 using for integration
 testing 251–254
 using for management
 testing 254–256
 using for unit testing 250–251
 using to test on multiple
 frameworks 246–250
 Pax Web 487–488
 PermGen 283
 Permission Admin Service 151
 Permission class 441
 permission management 439
 persistence, and component
 framework 348
 persistent identifier. *See* PID
 physical module. *See* module,
 physical
 PID, definition of 300
 Plain Old Java Object (POJO).
 See POJO
 PluginJAR 222
 POJO, publishing as
 service 125, 131
 policy
 as dependency
 characteristic 361
 roles in Declarative
 Services 362
 preference tree, navigating 313
 Preferences interface,
 methods 313
 Preferences object 312
 properties 312
 Preferences Service
 299, 312–314
 data model 312
 getSystemPreferences() 313
 getUserPreferences() 313
 getUsers() 313
 interface 313
 preference tree,
 navigating 313

Preferences Service (*continued*)
 similarity to Java
 Preferences 314
 system root 312
 user root 312
 private attribute 160
 PrivilegedAction 442
 programming
 in the large 26
 in the small 26
 programming to interfaces 124
 property, dynamic typing
 system 309
 provider, substitutability of, in
 component model 348
 proxy logger 167–170
 proxy, delegating service. *See* del-
 egating service proxy
 public key cryptography. *See* bun-
 dle, signing digitally
 public visibility 161
 publish-find-bind model 125
 publish, find, and bind interac-
 tion pattern 11

Q

qualifier 38

R

race condition, exposing 134
 with mock object 237
 with mock objects 243–244
 Red Hat JBoss. *See* JBoss
 refreshBundles() 109
 RegistryEvent 503
 RegistryListener 503
 remote service, exceptions 501
 Remote Services
 specification 493–510
 configuration properties
 498–499
 RemoteRegistry 503
 RemoteServiceReference 503
 removeBundleListener() 98
 removeFrameworkListener() 98
 removeRepository() 327
 RepositoryAdmin 322
 addRepository() 327
 discoverRepositories() 327
 discoverResources() 329
 getResolver() 328
 interface 326

listRepositories() 327
 removeRepository() 327
 UML diagram 326
 Require-Bundle 171, 233
 aggregating split
 packages 173–176
 and consistent bundles
 names 193
 reexport 173
 required bundle. *See* bundle,
 required
 requirement 325
 matching to capability 325
 resolution directive 164, 172
 resolution:=optional 202
 resolveBundles() 108, 110
 Resolver
 add() 328
 deploy() 328
 methods 328
 Resolver.resolve() 328
 resolving, definition 54
 resource 322
 deploying 328
 requirements, satisfying 322
 resource description 322
 resource processor 335–337
 joining to processing 336
 specifying 336
 ResourceBundle, creating 177
 ResourceProcessor 330
 interface 335
 reuse improvement, as benefit of
 modularity 64
 RFC 112. *See* OBR
 @RunWith 247

S

SCA 22, 496
 and OSGi 22
 composite component 22
 configuration 498
 ScalaModules 391
 security 440–444
 domain-based 441
 permissions 441
 privileged calls 442–444
 protection domains
 441–442
 establishing identity 439
 establishing permission
 policies 439
 file permissions,
 ServicePermission 448
 management, key aspects
 of 439
 org.osgi.framework.security
 471
 org.osgi.framework.trust.
 repositories 471
 performing permission
 checks 439
 permissions
 conditional 449–450
 OSGi-specific 444–448
 AdminPermission
 444, 446–447
 BundlePermission
 444–446
 names and actions 444
 PackagePermission
 444–445
 ServicePermission
 444, 447–448
 role-based 441
 starting framework with secu-
 rity enabled 471–475
 whether to use 439–440
 SelfStopThread 101
 semantics, capturing 121
 separation of concerns 25
 in component model 348
 modularity as form of 25
 service 118–119
 all matching, returning 147
 and bundles 221–225
 as configuration target 152
 as substitute for listener
 pattern 123
 best, choosing 128–130
 binding to implementation
 from registry 130
 bundle-specific 147
 can disappear at any
 time 125, 131
 compendium 151–152
 consuming, with Declarative
 Services 359–364
 contract 118, 120
 interface as part of 121
 using to discover services in
 registry 128
 core 150–151
 creating activator for 17
 customizer. *See* customizer
 dangling. *See* dangling service
 defining 126–127
 delegation 119

- service (*continued*)
 - dependency description 354
 - dereferencing 134
 - differentiating local vs. remote 501–502
 - discovering 17, 130
 - discovery vs. dependency injection 122
 - dynamics of 132–143
 - event types 136
 - external, publishing into framework instance 435
 - factory
 - creating instances when needed 147
 - must be thread safe 147
 - factory service 146
 - finding and binding 128–132
 - framework shows those using same version 147
 - highest-ranked, choosing 138
 - interfaces, bundling separately 149
 - introduction 11, 118–124
 - isn't a proxy 131
 - listener. *See* listener
 - listening for 136–141
 - looking up just before using 134
 - management, automating 352–354
 - metadata, updating 127
 - mock, using for early testing 121
 - MODIFIED 136
 - overhead of 124
 - properties 127
 - provider, introduction 11
 - providing, with Declarative Services 357–359
 - pseudo-registration event 139
 - publication of, handling with Declarative Services 355
 - publishing 126–128
 - remotely 496
 - ranking 128
 - reasons to use 119–123
 - REGISTERED 136
 - registered, automatically unregistered when bundle stops 17
 - registering 17
 - registry 11, 17
 - relating to modularity and lifecycle 146–149
 - removing 127
 - tracker
 - closing 141
 - methods 142
 - opening 141
 - tracking 141–143
 - ungetting 131
 - UNREGISTERING 136
 - using 130–132
 - between major components 123
 - in paint program 143–146
 - to choose between implementations 123
 - vs. component 119
 - vs. method call 118
 - when not to use 124
 - when to unget 148
 - when to unregister 148
 - when to use 123–124
 - working with 125–132
- Service Binder project 21
- Service Binder, and improving OSGi component model 351
- Service Component Architecture (SCA). *See* SCA
- service dependency and component lifecycle management 362
- characteristics 361–364
 - cardinality 361
 - policy 362
 - target filter 362
- service framework, managing listeners with 123
- service layer 10–12, 16–18
 - incorporation of service-oriented concepts 11
 - lightweight 11
 - promoting separation of interface and implementation 11
 - service-based dynamism 11
- service provider, discovering at execution time 415
- service registry
 - accessing 125
 - binding to implementation from 130
 - private 127
 - publish-find-bind model 125
 - use of indirect references 128
- service-oriented approach
 - benefits 120
 - plug and play 120
- service-oriented architecture (SOA) 11
- service-oriented component model 349–351
 - execution-time dynamism 351
- service-oriented interaction pattern 11, 351
- service-provider configuration files, META-INF/services directory 415
- service.description 127
- service.exported.interfaces 495
- service.id 127, 129
- service.imported.service
 - property 501
- service.intents.service
 - property 502
- service.pid 127, 300
- service.ranking 127, 129
- service.vendor 127
- ServiceFactory 147
- ServiceListener 136
 - All* extension 147
- ServiceReference 430
- ServiceReference API
 - getUsingBundles() 288
- ServiceTracker 141–143
- ServiceTrackerCustomizer 142
- servlet
 - app lifecycle 71
 - registering under named alias 151
- Servlet.destroy() 71
- Servlet.init() 71
- setBundleStartLevel() 338
- setInitialBundleStartLevel() 338
- setStartLevel() 338
- shell
 - bundle
 - activator 77
 - JAR file manifest 76
 - configuration properties 86
 - creating as bundle 73
 - functionality, implementing 85
- side-file syndrome, and component orientation 349
- Sigil 526
- signature. *See* bundle, signing digitally

sleep, replacing in unit tests 241
 SLF4J, refactored API 198
 SOA in a VM 11
 software, lifecycle 71
 split package
 package-private classes
 from 272
 See also package, split
 Spring Beans, and
 Blueprint 374
 Spring DM test support 245
 features 246
 Spring, Bundlor 527
 SpringSource Enterprise Bundle Repository 193
 Start Level Service 150
 Start Level service 338–339
 interface 338
 using 339–341
 start level, querying and
 setting 150
 Structure101 218
 substitutable export 157
 when to use 158
 Sun
 Jini. *See* Jini
 NetBeans. *See* NetBeans
 Project Fuji 21
 Project Jigsaw 22
 sun.misc.* 203
 superpackage 22
 SynchronousBundleListener 98
 event types 98
 system bundle 82
 lifecycle operations 82

T

TCCL 228, 280–282
 accessing 280
 default 281
 solving class-loading issue 281
 using with nested
 containers 280
 telnet binding 85
 TelnetBinding 78
 @Test 247

test

 bundle test 251
 bundling 232–235
 with application code 232
 coverage 235–236
 gathering 235
 turning JAR into
 bundle 235

 deployment options 232
 combination of inter-bundle
 and intra-bundle 234
 inter-bundle 233
 intra-bundle 234
 integration. *See* integration
 test
 management. *See* manage-
 ment test
 migrating to OSGi 231–236
 multithreaded 241–243
 running on as many contain-
 ers as possible 245
 running on multiple
 frameworks 246–250
 steps to build and deploy 245
 unit *See* unit test
 unit. *See* unit test
 test tools 245–246
 advantages and
 disadvantages 245
 features 245
 testing
 advanced 244–256
 early, with mock services 121
 expected behavior 237–238
 in-container 231–236
 instrumented classes 235
 interactions between
 components 251
 TestNG 231
 third-party software provider,
 establishing identity 439
 thread
 context class loader 431–432
 management 79
 Thread Context Class Loader
 (TCCL). *See* TCCL
 ThreadLocal behavior in
 Java 5 284
 transaction, and component
 framework 348
 Tycho 527
 typing system, dynamic, for
 properties 309

U

 ungetService(), counting calls
 compared to
 getService() 148
 unit test 250–251
 sleep in 241
 untrusted software, reasons to
 run 439

 URL Handlers Service 150, 213
 disabling 214
 URLStreamHandlerFactory
 150, 213
 URLStreamHandlerService 214
 User Admin Service 152
 uses constraint 275–277
 and optionally imported
 packages 197
 uses constraints 59–64
 transitive 63

V

 version number
 component changes 295
 components 38
 default 38
 format 38
 major number 38
 micro number 38
 minor number 38
 qualifier 38
 version range 46, 295
 use by implementers vs.
 users 297
 version verification, as benefit of
 modularity 64
 versioning
 as core task 295
 importance of 298
 meaningful, in OSGi 293–295
 module versioning. *See* mod-
 ule, versioning
 policy 295
 bundle and package
 numbers 297
 consistency 297
 pitfalls 296–297
 visibility directive 173
 visibility, and class loading 278
 VMware, OSGi Blueprint
 specification 21

W

 WAB 489
 creating 489
 WAR files, in OSGi 488–493
 WAR, turning into bundle 207
 web application
 and HTTP Service
 specification 479–488
 configuring 482

- web application (*continued*)
 - HttpContext interface 482–484
 - HttpService.createDefaultHttpContext() 482
 - org.osgi.service.http.port 482
 - registering resources 480–481
 - registering servlets 484–487
- ServletContext 487
 - support from Pax Web 487–488
 - and Web Applications specification 488–491
 - benefits of OSGi 478
 - creating 478–493
- web application bundle (WAB).
 - See* WAB
- web container extender
 - bundle 489
- web service 493–510
 - consuming 499–502
 - providing 494–499
 - using 500–501
- Web URL Handler 492–493
 - parameters 492
- WebSphere, OSGi and 19
- whiteboard pattern 123