

Symbols

-> operator 381
=> operator 381

A

ACID 284, 297, 300
ActiveMQ 4, 27
 dead letter queue 294
 TCP connectors 288
 VM protocol 288
ActiveMQ component 292, 303, 457
ActiveMQConnectionFactory 27
addInterceptStrategy
 method 378
addRoutes 28
aggregate 71
<aggregate> tag 241, 250
AggregationStrategy
 71, 240–241
 example 242
 thread safe 243
 using 242–243
Aggregator EIP 238–255
 aggregated size 246
 aggregation strategy 240–241
 AggregationStrategy 263
 combining message 242–243, 263
 completion condition 240–241, 243–248
 Aggregated 246
 batch consumer
 completion 244
 conditions provided 244
 examples 244
 interval completion 244
 predicate completion 244, 247
 size completion 244–245
 timeout completion 244–246
 using multiple conditions 245–247
 completion size 241
 configuration
 additional option 247
 closeCorrelationKeyOnCompletion 247
 eagerCheckCompletion 247
 groupExchanges 248
 mandatory 240
 correlation expression 241
 correlation group 243
 correlation identifier 240, 243, 246
 invalid 247–248
 Exchange properties 246
 in Splitter EIP 262
 ordering 243
 persistence 248–251
 AggregationRepository 248, 250
 example 250
 file based repository 249
 HawtDB 249
 memory repository 248
 pluggable repository 248
 RecoverableAggregationRepository 249
 setup HawtDB 249
 recovery 249, 251–255
 background task 253, 255
 commit 253
 dead letter channel 255
 HawtDBAggregationRepository 254
 interval 254–255
 maximum redeliveries 254–255
 recover 253
 RecoverableAggregationRepository 255
 redelivered 254
 rollback 253
 setup 254
 similar to JMS Broker 252
 transaction 252–253
 using error handler 251
 reject incoming message 247
 uses cases 239
akka 487
actor 488–489
 actor API 488
 bang operator 489
 pattern match 488
 reply 491
actor model 487
ActorComponent 495–496
akka-camel component 488
asynchronous 488
bangbang operator 492

- akka (*continued*)
 - CamelContextManager 494
 - sending message 491
 - CamelService 490, 495
 - customize 494–495
 - CamelServiceManager 490, 495
 - consume from Camel 489–492
 - disable JMX 494
 - endpointUri 490–492, 497
 - forwarding message 493, 498
 - immutable message 490
 - message exchange
 - pattern 493
 - one-way messaging 490
 - produce message 492–493
 - ProducerTemplate 492
 - receive 490–491, 497
 - receiveAfterProduce 493, 498
 - receiveBeforeProduce 498
 - reply 498
 - request-reply messaging 491
 - route
 - routing example 497–498
 - sending message to actor 496
 - software transactional
 - memory 487
 - some failure 493
 - some message 493
 - timeout 492–493
 - with Spring 494
 - XML response 491
 - alternative languages 380
 - annotation
 - @Attachments 114
 - @Autowired 96
 - @Bean 116
 - @BeanShell 116
 - @Body 107, 113–114
 - @Consume 446–447
 - @DynamicRouter 271
 - @EL 116
 - @EndpointInject 163
 - @Groovy 116
 - @Handler 107–108
 - @Header 107, 114
 - @Headers 114
 - @InOnly 459
 - @InOut 459
 - @JavaScript 116
 - @MVEL 116
 - @ONGL 116
 - @OutHeaders 114
 - @PHP 116
 - @Produce 447–450
 - @Properties 114
 - @Property 114
 - @Python 116
 - @RoutingSlip 269
 - @Ruby 116
 - @Simple 116
 - @XPath 116, 286
 - @XQuery 116
 - Apache ActiveMQ 361, 452, 477
 - Apache Camel website 483
 - Apache CXF 205
 - HTTP transport 434
 - link to website 205
 - Apache Felix 438
 - Apache Karaf 428, 437
 - adding Camel feature 439
 - features 439
 - install bundle 440
 - installing 439
 - listing installed bundles 440
 - log file 441
 - shutdown OSGi
 - container 441
 - starting 439
 - Apache Maven 359, 428
 - archetype plugin 360
 - archetypeArtifactId 362
 - archetypeVersion 362
 - create vs. generate goal 362
 - archetypes 360
 - artifactId 361
 - central repository 12, 364
 - configuring local Maven
 - repository in Eclipse 368
 - convention over
 - configuration 361
 - data directory 11
 - dependency tree 365
 - directories installed 11
 - generating an OSGi
 - bundle 438
 - groupId 361
 - local download cache 12
 - managing projects 360–366
 - Maven Eclipse plugin 366
 - obtaining 11
 - POM 11, 360
 - pom.xml 11, 361
 - quickstart archetype 360
 - recommended reading 12
 - src directory 11
 - transitive dependencies 364
 - using the m2eclipse
 - plugin 368
 - Apache MINA 216
 - Apache OpenJPA 227
 - Apache ServiceMix 353, 414
 - message bus 353
 - scalability 353
 - Apache Tomcat 215, 428
 - deploying Camel 432
 - hot deployment 432, 435
 - leverage servlet transport 434
 - starting 432–433
 - using CXFServlet 434
 - ApplicationContext 34, 429
 - ApplicationContextRegistry
 - registry 99, 101
 - Archetypes 360
 - asynchronous 335
 - messaging 197, 230, 286
 - Atomikos6 299
 - @Attachments 114
 - auditing 56
 - autocomplete 29
 - automating tasks 232
 - AutoStartup, disabling 418
 - @Autowired 96
-
- B**
- BatchConsumer 244
 - @Bean 116
 - bean
 - BeanProcessor 98, 103, 112
 - method
 - adapting to method
 - signature 111
 - AmbiguousMethodCall
 - Exception 107–109
 - BeanInvocation 106
 - beanRef 110
 - CamelBeanMethodName 105
 - @Handler 107–108
 - method selection
 - algorithm 105–107
 - MethodNotFoundException 104, 107, 109–110
 - NoTypeConversionAvailable
 - Exception 109–110
 - selecting bean method 103–111
 - selection example 107, 111
 - understanding bean method selection 104
 - method call expression 268

- bean (*continued*)
 - parameter binding 111–119
 - @Bean 117
 - @Body 107, 114
 - built-in types 113–114
 - exception 148
 - @Header 107, 114
 - multiple parameters 112–119
 - @NamespacePrefix 118
 - single parameter 111
 - table of annotations 114
 - table of build in types 113
 - table of language annotations 116
 - using annotations 114–119
 - using language annotations 115
 - @XPath 116
 - @XPath with namespace 118
 - registry 98–102
 - ApplicationContextRegistry 101–102
 - JndiRegistry 99–101
 - lookup 99
 - lookup in OSGi service registry 102
 - lookup in Spring 101
 - lookupByType 99
 - multiple registries 99
 - OsgiServiceRegistry 99–102
 - SimpleRegistry 100
 - table of registries 99
 - typesafe lookups 99
 - Service Activator EIP 94–98
 - Spring, @Autowired 96
 - using 94–97
 - bean in Java DSL 97
 - <bean> in Spring XML 96
 - beanRef 100
 - from a Processor 94
 - Bean component 98, 106
 - <bean> tag 96
 - BeanInvocation 456
 - @BeanShell 116
 - BeanShell 45
 - bindy 80
 - annotations 81
 - BindyType.Csv 82
 - @CsvRecord 81
 - data types used 81
 - @DataField 81
 - specifying package name 82
 - using CSV 80
 - using FIX 83
 - using fixed length 83
 - @Body 107, 113–114
 - browse 191
 - builder pattern 68
- C**
-
- Camel 4
 - alternative languages 380–384
 - archetypes
 - list of archetypes 361
 - using camel-archetype-component 371
 - using camel-archetype-java 362
 - architecture 15–20
 - CamelContext 16
 - component 18
 - consumer 20
 - domain-specific language (DSL) 17
 - endpoint 18
 - high level 15
 - modular and pluggable 7
 - processor 18
 - producer 19
 - routes 17
 - routing engine 17
 - bridging routes together 231
 - community 8, 483–484
 - component library 7
 - components, understanding 188–236
 - configuration, easy 7
 - ConsumerTemplate 480–482
 - core, lightweight 7
 - creating custom components 371–377
 - creating routes in Java 28–33
 - creating routes with Spring 34–43
 - creating WAR from Camel application 432
 - deploying Camel 428–441
 - developing interceptors 377–380
 - developing projects 359–384
 - directories installed 8
 - EIPs 43–57
 - endpoint 24
 - example
 - first 9–13
 - revisited 20
 - hide Camel APIs 443
 - hiding middleware 451–460
 - higher-level abstractions 4
 - Java DSL
 - bean method 54
 - choice method 44
 - convertBodyTo method 419
 - end method 47, 421
 - endsWith predicate 45
 - executorService method 51
 - from method 30
 - header method 45
 - log method 418
 - multicast method 51
 - noStreamCaching method 415
 - otherwise method 46
 - recipientList method 53
 - regex predicate 46
 - stop method 47
 - stopOnException method 52
 - streamCaching method 414
 - to method 32
 - tokenize method 53
 - using Camel Proxy from Java DSL 460
 - wireTap method 56
 - xpath method 49
 - Maven dependencies 364
 - message model 13–15
 - exchange 14
 - message 13
 - obtaining 8
 - OSGi ready 437
 - payload-agnostic router 7
 - POJO model 7
 - predicates and expressions 45
 - regular expression predicate 46
 - xpath expression 49
 - producer and consumer templates 477–482
 - ProducerTemplate 448, 477–480
 - protocol support 4
 - reasons to use 5
 - remoting and Camel proxy 456
 - route startup order 413
 - routing 22–57

- Camel (*continued*)
 - routing and mediation
 - engine 5
 - routing annotations 444, 450
 - @Consume example 446
 - dependencies 446
 - injecting a proxy
 - producer 449
 - loading annotated classes
 - into Camel 446
 - @Produce example 449
 - receiving messages with
 - @Consume 445
 - referencing a shared
 - endpoint 447
 - sending messages with
 - @Produce 448
 - specifying what Camel-Context to use 447
 - routing engine 4
 - routing with beans 443–451
 - running 410–442
 - Scala DSL 380–381
 - > operator 381
 - ==> operator 381
 - adding Scala routes to the CamelContext 382
 - comparison with Java DSL 380
 - EIPs support 382
 - maven-scala-plugin 383
 - Scala RouteBuilder 381
 - using Java-based Camel-Context with routes 382
- shutting down 424
- Spring DSL 37–40
 - adding a processor 38
 - adding multiple routes 39
 - advanced configuration
 - options 43
 - CamelBeanPostProcessor 447
 - configuring components
 - and endpoints 41
 - consumerTemplate
 - element 479
 - custom camelContext
 - element 412
 - defining a reusable
 - endpoint 42
 - dynamic RouteBuilder
 - loading 40
 - example 38
 - finding route builders 40
 - importing configuration
 - and routes 42
 - importing route defined in
 - another file 43
 - naming a component 41
 - packageScan element 40
 - proxy element 455
 - route id 417
 - routeContext element 43
 - routePolicyRef option 424
 - routerBuilder element 40
 - shutdownRoute option 427
 - starting a Spring Camel-Context manually 429
 - streamCache option 414
 - template element 479
 - when method 44
 - starting 411, 413–419
 - startup options 413
 - scope 415
 - TypeConverter 33
 - using in Eclipse 366–371
 - web console 362, 419
- Camel community
 - articles, blogs, podcasts 486
 - IRC and JIRA 486
 - mailing list and reference
 - card 486
 - website 483
- .camel directory 194
- Camel extensions, JUnit 155
- Camel Extra project 436, 485
- Camel registry 99
- Camel Test Kit 155–166
- camel-core module,
 - components 191
- CamelContext 11, 190, 411
 - accessing via Processor 64
 - createNewProducerTemplate
 - method 478
 - graceful shutdown 425
 - namespace 412
 - setStreamCaching
 - method 414
 - start method 411
 - starting 412
 - startRoute 419
 - stopRoute 419–420
- CamelContext id 448
- <camelContext> tag 137
- CamelContextFactoryBean 412
- CamelInvocationHandler 458
- CamelNamespaceHandler 412
- Channel 125–126, 132, 181, 398
- code-first 205, 215
- codec 218
- commons-net 364
- compensation 309
 - rollback 310
 - using synchronization 310
- Component 18, 374
- component
 - ActiveMQ 303, 457
 - activemq transaction 292
 - actor 495
 - asynchronous
 - processing 353–354
 - akka-camel 488
 - Bean 98, 106
 - bean, using OGNL 468
 - custom component,
 - @ManagedAttribute 407
 - customize output log 396
 - Direct 304, 322
 - @EndpointInject 163
 - fault 138
 - File 310–311
 - file
 - concurrency 332
 - simple language
 - variables 465
 - FTP 310
 - simple language
 - variables 465
 - HawtDB 249, 251
 - HTTP 151
 - Jasypt 164
 - Jetty 83, 142, 175, 387
 - continuations 354
 - JMS 169
 - log 395
 - MINA 73, 142, 175
 - Mock 155, 166–178
 - unit testing 167–178
 - Properties 161–165, 465
 - properties file 165
 - syntax 162
 - using without Spring 163
 - SEDA 169, 273, 304, 322
 - simulating 175–177
 - smtip 88
 - velocity 88
 - XSLT 73
- Component interface 190
- component resolver 191
- components
 - ActiveMQ 198
 - autodiscovering
 - components 190

- components (*continued*)
 - bean 191
 - example using InOut MEP 201
 - how BeanComponent is resolved 191
 - camel-core module 191
 - creating a custom component 371
 - CXF 205
 - code-first
 - development 215–216
 - configuring using endpoint bean 207
 - configuring using URI options 206
 - contract-first
 - development 209–215
 - cxf-codegen-plugin in Maven 211
 - dataFormat option 207
 - dataformats 207
 - document literal 211
 - exposing a route as a web service 208, 213
 - HTTP-based
 - components 214
 - invoking a web service 213
 - Maven dependencies 208
 - modes of development 205
 - operationName header 214
 - portName option 206
 - producer and consumer terminology in web services 208
 - Service Endpoint Interface 206
 - serviceClass option 206
 - serviceName option 206
 - specifying operation to invoke 214
 - transports 209
 - using a WSDL 211–213
 - using Apache Tomcat for
 - HTTP inbound endpoints 434
 - wsdl2java tool 211
 - wsdl2service tool 210
 - wsdl2xml tool 210
 - wsdlURL option 206
 - dataset 191
 - Direct 229, 455
 - how a consumer calls a producer 376
 - File 191
 - default filename 195
 - delay option 193
 - delete option 193–194
 - exclude option 193
 - fileName option 195
 - include option 193–194
 - locking files 194
 - maxMessagesPerPoll option 420
 - move option 193–194
 - noop option 38, 193
 - reading and writing files 193–194
 - recursive option 193
 - simple expression for filename 195
 - writing files example 194
 - File and FTP
 - Components 192–197
 - FTP 24–25
 - binary option 196
 - CamelFileName header 33
 - consuming files 25
 - default FTP port 25
 - disconnect option 196
 - example using embedded FTP server 197
 - Maven dependency 25, 196, 364
 - maximumReconnectAttempts option 196
 - password 25
 - password option 196
 - reconnectDelay option 196
 - remote directory 25
 - transitive
 - dependencies 364
 - URI options 196
 - username 25
 - username option 196
 - using embedded FTP server for testing 197
 - Hibernate 221
 - HTTP 214, 479
 - JBI 414–415
 - JBoss 436
 - JDBC 221–224
 - example 224
 - Maven dependency 221
 - readSize option 222
 - statement option 222
 - URI options 222
 - useJDBC4ColumnNameAndLabelSemantics option 222
 - using a bean to create the SQL statement 223
 - Jetty 214
 - JMS 26–28, 197–204, 457
 - ActiveMQConnectionFactory 197
 - autoStartup options 199
 - clientId options 199
 - concurrentConsumers options 199
 - configure JMS provider 26
 - connecting to ActiveMQ 27
 - over TCP 197
 - disableReplyTo options 199
 - durableSubscriptionName options 199
 - endpoint URI 27
 - exchangePattern option 202
 - how headers are handled 204
 - how to empty a JMS queue 480
 - JMSCorrelationID header 201
 - JMSReplyTo header 201
 - mapJmsMessage option 204
 - Maven dependency 27, 198
 - maxConcurrentConsumers options 199
 - message mappings 202
 - replyTo option 199, 202
 - request/reply
 - messaging 201
 - requestTimeout options 199
 - selector options 199
 - sending and receiving messages 200
 - sending messages with the ProducerTemplate 201
 - TCP as transport 457
 - timeToLive option 199
 - transacted options 199
 - URI options 198
 - using a topic 200
 - using multiple JMS providers 42
 - valid JMS headers 204

- components (*continued*)
 - JPA 224–229
 - connecting Camel JPA to OpenJPA 227
 - consumeDelete option 225
 - consumeLockEntity option 225
 - consumer.delay option 225
 - consumer.initialDelay option 225
 - consumer.namedQuery option 225
 - consumer.nativeQuery option 225
 - consumer.query option 225
 - @Entity 226
 - example 228
 - flushOnSend option 225
 - manually querying the database 228
 - Maven dependencies 224
 - maximumResults option 225
 - maxMessagesPerPoll option 225
 - persistenceUnit option 225
 - persistenceUnitName 227
 - transactionManager option 225
 - URI options 225
 - LGPL/GPL components 436
 - manually adding to the CamelContext 190
 - message headers set 33
 - MINA 216–221
 - codec option 217
 - CumulativeProtocolDecoder 220
 - custom codecs 219, 221
 - encoding option 217
 - example 218
 - filters option 217
 - Maven dependency 216
 - object serialization codec 219
 - setting up a TCP server 218
 - sync option 217
 - textline codec 218
 - textline option 217
 - textlineDelimiter option 217
 - timeout option 217
 - transferExchange option 217
 - transport types 217
 - URI options 216
 - mock 192
 - naming a component in Spring 41
 - no component found error 363
 - number of components 189
 - overview 189–192
 - Quartz 233–235
 - cron option 234
 - cron triggers 235
 - job.name option 235
 - job.propertyName option 234
 - JobDetail 234
 - Maven dependency 234
 - SimpleTrigger 234
 - Trigger 234
 - trigger.propertyName option 234
 - trigger.repeatCount option 234
 - trigger.repeatInterval option 234
 - URI options 234
 - ref 192
 - SEDA 192, 230, 455
 - concurrentConsumers option 230
 - multipleConsumers option 230–231
 - pros and cons 230
 - publish-subscribe 231
 - size option 230
 - timeout option 230
 - URI options 230
 - waitForTaskToComplete option 230
 - Servlet 215, 436
 - SQL 221
 - stream 194
 - timer 232–233
 - VM 192, 230–232
- Composed Message Processor
 - EIP 238
- concurrency 316
 - akka 487
 - asynchronous task 342
 - CPU-bound 319
 - eip
 - aggregate 330
 - multicast 330, 332–334
 - recipient list 331
 - split 331
 - threads 331
 - using 330–335
 - wiretap 331, 334–335, 343
- Future 342
- IO-bound 319
- ordering 334
- parallel processing 319–320
- parallelProcessing 333
- performance 334
 - improving 319
- splitter,
 - parallelProcessing 319
 - transaction limitation 335
- concurrency client API
 - asynchronous task 344
 - Callable 344–345
 - ExecutorService 344
 - Future 344–345, 347
 - future
 - get 345–346
 - get with timeout 346
 - isDone 345–346
 - in Camel 347–350
 - in Java 344–347
 - ProducerTemplate 347
 - asyncCallback 348–350
 - asyncCallbackRequestBody 348–349
 - asyncCallbackSendBody 348
 - asynchronous methods provided 348
 - asyncRequestBody 348
 - asyncSend 348
 - asyncSendBody 348
 - Runnable 344
 - Synchronization 348
 - SynchronizationAdapter 349
 - timeout 346, 349–350
- ConnectionFactory 26
- @Consume 445–447
- consumer 33, 374
 - event-driven consumer 20
 - polling consumer 20
 - ScheduledPollConsumer 124
- Consumers 477
- ConsumerTemplate, list of methods 480
 - receiveBodyNoWait 294
- Content Enricher EIP 71
 - enrich 71, 73
 - pollEnrich 71, 73
- Content-Based Router EIP
 - 44, 295, 304

- <context:property-placeholder>
 - tag 165
 - contextConfigLocation 431
 - convention over configuration 7
 - correlation id 397, 399
 - <correlationExpression>
 - tag 241
 - cron expressions 235
 - CSV 23, 44
 - table of data types used 80
 - transforming using bean 66
 - transforming using Processor 64
 - custom component 371
 - changing endpoint URI scheme 372
 - Component and Endpoint classes 373
 - data marshaling 376
 - disabling producer or consumer creating 374
 - extending from default implementations 374
 - how endpoint URI properties work 373
 - Producers and Consumers 375
 - what makes up a component 373
 - CXFServlet 434
- D**
-
- data format
 - configuring 84
 - CSV 317
 - camel-csv 79
 - configure 84
 - consume CSV files 79
 - DataFormat API 85
 - JSON 83
 - camel-jackson 83
 - camel-xstream 83
 - marshal 75, 77, 85
 - table of data formats 78
 - unmarshal 75, 77, 85
 - using JAXB 77
 - using XStream 75
 - writing custom 85
 - ReverseDataFormat 85
 - using Camel type converters 86
 - data format transformation 62
 - data integration and data mapping 63
 - data mapping, definition 63
 - data transformation 62–63
 - enrich 71
 - in routes 63–73
 - message transformation in
 - component adapter 63
 - using bean 66–67
 - using components 63
 - using data formats 63, 77–86
 - using Freemarker 86
 - using Processor 64
 - using templating 86–88
 - using the Content Enricher EIP 70
 - using the Message Translator EIP 63
 - using transform()
 - from Java 67–69
 - from Spring XML 69–70
 - using Velocity 86–87
 - using XStream 75
 - XSL Transformations (XSLT) 73
 - data type transformation 62
 - using type converter 63
 - database
 - HSQLDB 287
 - SQL 285
 - databases 221
 - Dead Letter Channel EIP 124, 126, 294
 - decouple middleware from business logic 451
 - DefaultComponent 373
 - DefaultConsumer 376
 - DefaultEndpoint 374
 - DefaultProducer 375
 - Delayer EIP 134
 - dependencies 364
 - deploying Camel
 - in a Java application
 - file copy example 428
 - pros and cons 430
 - using Main helper class to load Spring context 429
 - using raw Spring 429
 - in a web application 430–431
 - in Apache Tomcat 432
 - in Jetty 432
 - overriding CamelContext filename 431
 - pros and cons 436
 - using Spring context listener 431
 - in Apache Karaf 437
 - in JBoss Application Server 436
 - pros and cons 437
 - in OSGi
 - installing a bundle 440
 - pros and cons 441
 - standalone 428
 - deployment strategy 428
 - Apache Tomcat 432
 - pros and cons 436
 - Java application 428
 - pros and cons 430
 - Java EE Application Server 437
 - JBoss Application Server 436
 - pros and cons 437
 - Jetty 432
 - OSGi 437
 - pros and cons 441
 - standalone 428
 - pros and cons 430
 - web application 430
 - pros and cons 436
 - developing Camel projects 359
 - Direct component 191, 304
 - distribution, obtaining 8
 - doc directory 9
 - doCatch 144
 - doFinally 144
 - domain-specific language (DSL) 6, 17, 30
 - doTry 144
 - DSL 6
 - which DSL to use? 39
 - Dynamic Router EIP 238–239, 270–272
 - @DynamicRouter 271
- E**
-
- EAI 451
 - Eclipse 29, 210, 359
 - configuring local Maven repository 368
 - import wizard 367
 - M2_REPO variable 368
 - m2eclipse plugin 368
 - Maven Eclipse plugin 366
 - package explorer 367
 - using Camel 366
 - Eclipse Equinox 438
 - EIP 238–239
 - Aggregator 238–255
 - command message 222

- EIP (*continued*)
 - Composed Message
 - Processor 238, 262
 - concurrent consumers 318, 322–323
 - content enricher 242
 - Content-Based Router 295, 304
 - content-based router
 - (CBR) 44
 - example 45
 - routing after a CBR 47
 - Dead Letter Channel 124, 126–128, 135, 254, 294
 - Delayer 134
 - Dynamic Router 238–239, 266, 270–272
 - @DynamicRouter 271
 - Load Balancer 238–239, 272–280
 - log 396
 - message filter 49
 - Message Translator 307
 - multicast 51
 - configuring the thread pool 51
 - implement with a JMS topic 200
 - parallel multicast 51
 - parallelProcessing 51
 - stopping on exception 52
 - multicast, aggregate reply 333
 - patterns supporting
 - concurrency 330
 - Recipient List 147
 - recipient list 52
 - comma-separated recipients 53
 - example 55
 - RecipientList
 - annotation 54
 - Routing Slip 238–239, 266–270
 - @RoutingSlip 269
 - Service Activator EIP 97, 103
 - Splitter 238, 255–266, 317, 416
 - big file 318
 - configuring thread pool 320
 - file example 317
 - low memory usage 317
 - ordering 319–320
 - parallelProcessing 319
 - streaming mode 317
 - thread pool 320
 - threads 328
 - Throttler 134
 - Transactional Client 296–297
 - wiretap 334, 394–395
 - concurrency 334–335
- @EL 116
- EL 45
- embedded FTP server 197
- endpoint 18, 374
 - BrowsableEndpoint 184
 - @EndpointInject 162
 - file 158, 179
 - FTP 179
 - HTTP 179–180, 182
 - mock 167–178, 256
 - SEDA 273, 322
 - TCP 145, 148
 - URI 161, 164–165, 267, 269–270
 - URL 182
- <endpoint> tag 165
- Endpoint URI 24
 - context path 19, 25
 - how options work 373
 - options 19, 25
 - scheme 19, 25
- @EndpointInject 163
- enrich 71
 - Enrich 73
 - vs. pollEnrich 71
- enterprise integration patterns (EIPs) 5
 - implemented as Processor 33
- Enterprise Service Bus (ESB) 353
 - Camel is not an ESB 4
- @Entity 226
- entity manager 227
- error handler
 - HandleFault 415
 - Load Balancer EIP
 - failover 277
 - TransactionErrorHandler 298
- error handlers 124–129
 - DeadLetterChannel 124, 126
 - DefaultErrorHandler 124–125, 132
 - features 128
 - LoggingErrorHandler 124, 128
 - NoErrorHandler 124, 128
- scope 135
- TransactionErrorHandler 124, 128
- error handling
 - caught exception 147
 - context scoped 135, 137
 - continued 148
 - dead letter channel 126–128
 - dead letter endpoint 127
 - dead letter queue 136
 - useOriginalMessage 128
 - using original message 127–128
 - dead letter queue 126–127
 - default error handler 125–126
 - default settings 126
 - detour 126
 - detour message 150–151
 - doTry...doCatch...doFinally 144
 - error handlers provided 124–129
 - errorHandler 127, 135
 - errorHandlerRef 137
 - exception
 - caused exception 147
 - context scope 144
 - continued 146
 - custom exception
 - handling 146–148
 - detour message 138
 - exception hierarchy 139
 - gap detection 140
 - handle 138
 - handled 145–146
 - maximumRedeliveries 141
 - multiple exception 142
 - onException 139, 141–142
 - onWhen 150–151
 - redeliverDelay 141
 - understanding
 - catching 139–142
 - understanding handle 143–146
 - understanding
 - redelivery 142–143
 - using exception
 - policies 138–150
 - using processor 146
 - failover 149
 - failure endpoint 147
 - fault 137–138
 - getException 122

- error handling (*continued*)
 - handle fault 138
 - ignoring exception 148
 - lifecycle 124
 - logging error handler 128
 - no error handler 128
 - onException 143, 149, 178
 - original input message 126
 - PollingConsumerPollStrategy 124
 - redeliver 125
 - redelivery
 - asynchronous 131, 134
 - backOffMultiplier 132
 - custom processing 151
 - exhausted 133, 136
 - exponential backoff 136
 - file rollback 130
 - maximumRedeliveries 132
 - onRedeliver 151
 - recoverable errors 129
 - redeliverDelay 130
 - redelivered 133
 - redelivery counter 133
 - redelivery policy 130, 142
 - redeliveryDelay 132
 - redeliveryPolicy 136
 - retryAttemptedLogLevel 132
 - retryWhile 152–153
 - synchronous 134
 - table of redelivery options 130
 - useExponentialBackOff 132
 - using 129–137
 - RedeliveryErrorHandler 125, 128
 - route scoped 135
 - scope 129, 132, 135–137
 - setException 122, 124
 - setFault 123
 - table of error handlers 124
 - transaction error handler 128
 - try catch 143
 - understanding 121–124
 - irrecoverable 121
 - recoverable 121
 - redelivery 122
 - where error handling applies 123–124
 - <errorHandler> tag 132, 137
 - errors, recoverable and irrecoverable 121, 123
 - event
 - EventNotifier 402
 - EventObject 404
 - filtering event 403
 - notification 402–405
 - using custom
 - EventNotifier 403–405
 - examples directory 9
 - exception policies 138–150
 - <exception> tag 146
 - ExceptionHandler 423
 - exchange 14
 - exchange ID 14
 - in message 15
 - instance, working on via Processor 64
 - MEP 14
 - out message 15
 - properties 15
 - exchange id 397
 - Exchange.CAUSED_EXCEPTION 127
 - expression 471
 - compound 68
 - custom 68
 - evaluate 471–472
 - ExpressionAdapter 473
 - fluent builder, syntax sugar 472
 - Message Translator EIP 471
 - transform 472
 - method call 69, 259, 473, 475
 - syntax sugar 471
 - using 471–472
 - custom expression 472–473
 - in Java DSL 472
 - with Spring XML 472
 - using scripting language as 69
 - expression builder methods 45
 - expression language 461
 - dependency 461
 - evaluate 461
 - expression 461
 - Simple 397, 461
 - external DSL 30
- F**
-
- failed to resolve endpoint 363
 - failover
 - Load Balancer EIP 274–278
 - strategy
 - custom 278
 - priority based 278
 - round robin 277
 - <failover> tag 278
 - File component 310–311
 - file endpoint 158, 179
 - file transfer 192
 - files 192
 - filesystem 192
 - filter 49
 - flow of messages in route 32
 - fluent interface 30
 - <from> tag 165
 - FTP 23–24, 196
 - FTPComponent 25, 196, 310
 - FTPEndpoint 25, 179
 - FTPS 196
- G**
-
- generate an Eclipse project 29
 - GenericFile 396
 - graceful shutdown 424–425
 - example 427
 - @Groovy 116
 - Groovy 45, 380
- H**
-
- HandleFault 415
 - @Handler 107–108
 - HawtDB 249
 - persistence aggregator 249–251
 - setup 249
 - transaction 253
 - HawtDB component 249, 251
 - @Header 107, 114
 - @Headers 114
 - hello world 9
 - Hibernate 224
 - hiding middleware 443, 451
 - high availability 387
 - Hohpe, Gregor 5
 - HTTP 23, 214, 479
 - HttpOperationFailed-Exception 151
 - ping service 387
 - HTTP endpoint 179–180, 182
 - HyperSQL 224
- I**
-
- IDEs 366
 - importing Camel project into Eclipse 366

importing generated Eclipse project 29
 in-flight messages 425
 in-flight registry 420
 in-memory messaging 229
 @InOnly 459
 InOnly MEP 218, 478
 @InOut 459
 InOut MEP 201, 230, 478
 inspecting messages 55
 integration testing 183–187
 interceptors 181, 377

- adding an interceptor to the CamelContext 378
- addInterceptStrategy method 378
- applying more than one interceptor 378
- delay 399
- example 379
- excluding a Processor from interception 379
- how interceptors modify a route 378
- intercept 181
- intercept multiple endpoints 182
- InterceptFromEndpoint 181
- InterceptStrategy interface 377
- wrapping Processors in interceptors 377
- wrapProcessorInInterceptors method 377

 InterceptStrategy 377
 internal DSL 30
 Inversion of Control (IoC) 34

J

Jasypt component 164
 Java 364
 Java API for XML Web Services. *See* JAX-WS
 Java DSL 28, 30–33
 Java EE container 437
 Java Management Extensions. *See* JMX
 Java Message Service. *See* JMS
 Java Persistence Architecture. *See* JPA
 Java Transaction API (JTA) 298
 Java Web Start 441
 java.util.Timer 233
 @JavaScript 116
 JavaScript 45
 javax.jms.Message 203
 javax.jws.WebService 215
 javax.persistence.Entity 226
 javax.sql.DataSource 222
 JAX-WS 205, 215
 JAXB

- annotated bean 76
- contextPath 77
- jaxb.index 77
- using annotations 76
- @XmlAccessorType 76
- @XmlAttribute 76
- @XmlRootElement 76

 JBI 414
 JBoss Application Server 428

- starting 436

 JConsole 388, 390, 401, 405, 408, 419
 JDBC 221, 477
 jdbc, data source 288
 JdbcTemplate 477
 Jetty

- continuations 354
- deploying Camel in 432

 JMS 26, 477

- acknowledge mode, auto 289
- ActiveMQ, embedded broker 288
- browse queue 184
- BytesMessage 203
- consumer 285
- dead letter queue 294, 304
- how to empty a JMS queue 480
- JMS message implementations 202
- MapMessage 203
- ObjectMessage 203
- StreamMessage 203
- TextMessage 203
- transaction 290

 JMS component 169, 457
 JMS destination 26
 JmsTemplate 477
 JMX 386, 388

- DefaultManagementAgent 392
- exposed Camel MBeans 391
- JMX connector 391
- jmxAgent 392
- @ManagedResource 407
- ManagementAgent 391–392
- ManagementAware 407
- managing Camel application 405–408
- managing lifecycle 405–406
- managing Tracer 400–402
- MBean 391
- Spring JMX 389
- using JConsole 390–393
 - remotely 391–393
- using JMX with Camel 389–393

 JNDI

- JndiRegistry 101
 - lookup in WebSphere 101
- JndiRegistry 99, 101, 220

 JPA 221, 224
 JpaTransactionManager 225
 jsch 365
 json

- JSON 83
 - marshaling beans 84
 - selecting json library 83

 JUnit 154–166

- extensions 155

 JXPath 45

L

lib directory 9
 LICENSE.txt 9
 Load Balancer EIP 238–239, 272–280

- across remote service 272
- concept behind 272
- introducing 272–274
- strategy 272, 274–275
 - custom 274
 - failover 277–278
 - failover by exception 276
 - failover inherit error handler 277
 - failover maximum attempts 277
 - failover with round robin 277–278
 - random 274
 - round robin 272, 274
 - sticky 274–275
 - topic 274

 <loadBalance> tag 279
 log 191
 log4j 403

M

-
- M2_REPO variable 368
 - m2eclipse plugin 368
 - creating new Camel projects 369
 - installation instructions 369
 - performance issues 369
 - running a project 370
 - using archetypes 369
 - Main 430
 - main class 430
 - management 407
 - JMX
 - @ManagedAttribute 407
 - ManagementAware 407
 - ManagementStrategy 392, 403
 - managing
 - Camel application 405–408
 - consumer 406
 - custom component 406–408
 - @ManagedResource 407
 - lifecycle 405–406
 - stop route 406
 - Maven archetype plugin 360
 - maven-eclipse-plugin 366
 - maven-scala-plugin 383
 - Maven, Eclipse Maven plugin 29
 - MBean 391
 - mediator service 97
 - message 13
 - body 14
 - transforming from one form to another 62
 - correlate 399
 - exchangeId 133
 - fault message 14, 123–124, 137
 - headers 13
 - in-flight 424–425
 - poison message 294
 - tracing 398
 - message exchange pattern (MEP) 335, 399
 - InOnly 14, 335, 337, 399
 - InOut 14, 335, 338, 341, 399
 - message exchange using akka 488
 - message filter 49
 - message processing
 - asynchronous 336–340, 350
 - InOnly 337, 339
 - Threads EIP 339
 - Wire Tap EIP 342
 - asynchronous API
 - AsyncCallback 356
 - asynchronous message processing 352–353
 - asynchronous processing model 353
 - asynchronously 335
 - blocked thread 351
 - concurrently 319
 - in parallel 333
 - in sequence 333
 - InOnly 336
 - multiple threads 335
 - returning early reply to caller 342–344
 - synchronous 340–342, 351
 - InOut 338–340
 - synchronously 335
 - message transformation using custom expression 473
 - Message Translator EIP 307
 - messaging annotations 451
 - message processing
 - asynchronous API 354–356
 - AsyncCallback 355–356
 - AsyncProcessor 355, 357
 - callback 357
 - DefaultAsyncProducer 356–357
 - process 356–357
 - using AsyncProcessor rules 355
 - asynchronous message processing
 - callback 354–355
 - using AsyncProcessor rules 357
 - method signature naming 67
 - <method> tag 260
 - middleware 451
 - mina 73
 - Mock component 155, 166–178
 - unit testing 167, 178
 - mock endpoint 167–178, 256
 - mocked service 454
 - monitoring
 - application activity 393–405
 - audit 395
 - audit log 394
 - correlation 397
 - correlation id 397
 - event 403
 - lifecycle event 394
 - log EIP 396
 - log file scanning 393
 - trace logs 398
 - using custom log 394–398
 - using log 393–398
 - using notification 402–405
 - using Tracer 398–401
 - Camel application 386–389
 - health
 - checking at application level 388–389
 - checking at JVM level 388
 - checking network level 386–388
 - load balancer 387
 - Java Management Extensions (JMX) 386, 388
 - Nagios 386
 - periodic health check 386
 - ping service 386
 - Simple Network Management Protocol (SNMP) 386, 388
 - using JMX 389–393

- multicast 50
- @MVEL 116
- Mvel 45

N

-
- Nagios 386
 - NamespaceHandler 412
 - networking 216
 - noAutoStartup 419
 - NOTICE.txt 9
 - notification
 - configuring
 - EventNotifier 403
 - EventNotifier 403
 - filtering event 403
 - LoggingEventNotifier 402–403
 - PublishEventNotifier 402
 - setEventNotifier 403
 - using 402–405
 - using custom
 - EventNotifier 403–405
 - NotifyBuilder 185–187

O

-
- Object/Relational Mapping (ORM) 224
 - Odersky, Martin 381
 - OGNL 45
 - accessing bean 468
 - accessing List 469

OGNL (*continued*)
 accessing Map 469
 bean parameter binding 469
 null safe operator 469
 onCompletion 420–421
 different from
 synchronization 313
 onCompleteOnly option 422
 onFailureOnly option 422
 OnWhen 422
 scope 313
 using 312–313
 onException 139
 and exceptions 143
 and redelivery 142–146
 onExchangeBegin method 422
 onExchangeDone method 422
 @ONGL 116
 onRedeliver 151
 onWhen 150
 Open eHealth Integration Platform (IPF) 380
 OpenJPA 224
 ordering routes 416
 OSGi 411, 428
 hot deployment 437
 import and export 438
 Maven bundle plugin 438
 OSGi bundle 438
 OSGi compliant 437
 osgi service 102
 OsgiServiceRegistry 102
 Spring Dynamic Modules 102
 with Maven 438
OSGi in Action 437–438
 OsgiServiceRegistry registry
 99, 102
 @OutHeaders 114

P

parameter binding, in
 beans 111–119
 parameter-binding
 annotations 446
 payload conversion 32
 performance 316
 impact 321
 improving 319, 323, 333
 persistence.xml 228
 @PHP 116
 PHP 45
 ping service 386
 pipeline 32

Plain Old Java Object (POJO)
 7, 23, 205
 messaging 444
 pollEnrich 71
 vs. enrich 71
 POM 11
 predicate 473
 combining predicates 475
 compound predicate 475
 and 476
 or 476
 PredicateBuilder 476
 custom predicate 474–475
 using in Java DSL 475
 using in Spring XML 475
 definition 474
 Filter EIP 474
 matches 474
 syntax sugar 474
 fluent builder 474
 using
 with Java DSL 474
 with Spring XML 474
 predicate interface 45
 Processor 18, 64, 94
 inlined in route 95
 interface 33
 introduction 64
 invoking a bean 94
 process 95
 using to translate custom format to CSV 64
 @Produce 447–450
 Producer 19, 374
 producers 33, 477
 ProducerTemplate 100, 229
 example 478
 request-reply example 479
 requesting a response
 202, 214
Programming in Scala 381
 Project Object Model. *See* POM
 project templates 360
 @Properties 114
 properties 192
 Properties component 161–165
 @Property 114
 property placeholders 164–166
 <proxy> tag 458
 ProxyBuilder 460
 ProxyHelper 458
 proxying Camel route as a
 interface 455
 publish/subscribe 26

@Python 116
 Python 45

Q

QName 206
 Quartz 233
 scheduling job with 65
 queue 26

R

reading files 193
 README.txt 9
 Recipient List EIP 147
 @RecipientList 54
 recovery, Aggregator EIP 251
 redelivery 130–135
 asynchronous 134
 attempts 122
 policy 130
 policy options 130
 registry 25
 ApplicationContextRegistry
 99, 101
 Camel 99
 JndiRegistry 99, 101
 OsgiServiceRegistry 99, 102
 SimpleRegistry 99–100
 request-reply messaging 459
 resource
 CPU-bound 316, 350
 IO-bound 316, 350, 352
 REST 362
 retryWhile 152
 Rider Auto Parts 217, 222, 224,
 416, 418, 434
 introduction 23
 inventory update 444
 inventory updates from
 suppliers 411
 starter kit 451
 web service for order
 submission 206
 route
 additional routing using
 OnCompletion 420
 adviceWith 182
 assign id 182
 AutoStartup 416, 418
 defining in Spring XML
 file 96
 defining with
 RouteBuilder 95

- route (*continued*)
 - difference between stop and suspend 428
 - flip routes being active 422
 - lookup by id 182
 - OnCompletion 420
 - ordering 416–418
 - RouteDefinition 182
 - RoutePolicy 422
 - start route 422
 - stop route 422
 - shared route 427
 - StartupOrder 416
 - stop graceful 428
 - RouteBuilder 21, 28–30, 95
 - add to CamelContext 29
 - anonymous RouteBuilder class 29
 - configure method 29
 - RouteBuilder interface 28
 - RouteDefinition 30
 - routeId 417
 - RoutePolicy 422
 - method 423
 - RoutePolicySupport 422–423
 - router, payload agnostic 7
 - routes 17
 - introduction 11
 - ManagedRoute MBean 419
 - ordering example 416
 - starting and stopping
 - programmatically 419
 - starting and stopping via JMX 419
 - StartupOrder 418
 - option 417
 - stopping and starting with CamelContext 420
 - stopping and starting with RoutePolicy 422
 - streamCache option 415
 - routing
 - slow processing down 399
 - routing engine 17, 350, 353
 - asynchronous 350, 353–354, 356–358, 488, 498
 - advantage 354
 - disadvantage 354
 - synchronous 353, 355
 - Routing Slip EIP 238, 266–270
 - @RoutingSlip 269
 - using a bean as slip 267
 - using expression as slip 268
 - using header as slip 267
 - @RoutingSlip 269
 - @Ruby 116
 - Ruby 45, 380
- S**
-
- Scala 380–381
 - maven-scala-plugin 383
 - Scala plugin for Eclipse 381
 - Scala DSL 361, 380
 - scala, akka 488
 - scalability 350
 - Apache SerivecMix 353
 - asynchronous
 - message processing 350
 - processing model 352–353
 - routing engine 350
 - writing custom component 356–358
 - blocked thread 351, 355
 - error handling 134
 - high scalability 353–354
 - in Camel 352–353
 - increasing load 351
 - jetty
 - consumer 350
 - continuations 354
 - thread pool 351
 - limit 352
 - scalability goal 352
 - scalability problem 350–352
 - thread blocked 352
 - schedule tasks 232
 - ScheduledPollConsumer 376
 - scripting language, using as expression 69
 - SEDA component 169, 273, 304, 455
 - sending to multiple destinations 50
 - Service Activator EIP 97–98, 103
 - service level agreement 23, 284
 - service provider interface, registry 99
 - service-oriented architecture. *See* SOA
 - ServiceMix 4
 - SFTP 196
 - shutdown 424
 - defer 425, 427
 - graceful 424
 - timeout 425
 - reliable 424–425, 427
 - ShutdownRoute option 427
 - suspend 425
 - shutdownRoute method 427
 - ShutdownRoute.Defer 427
 - ShutdownStrategy 425
 - @Simple 116
 - Simple
 - built-in file variables 465–466
 - file extension 466
 - filename 465–466
 - built-in functions 463–465
 - bean 464
 - bodyAs 463
 - date 464
 - properties 464–465
 - built-in operators 466–468
 - automatic type coerce 467
 - combining two expressions 468
 - contains 466
 - equals 466–467
 - greater than 466
 - in 466
 - less than 466–467
 - null safe operator 468
 - range 467
 - regex 467–468
 - syntax 467
 - with built-in functions 468
 - built-in variables 462–463
 - body 463
 - date 466
 - header 463
 - property 463
 - expression 461, 470
 - OGNL 468–469
 - operator 462
 - predicate 461–462, 470
 - SimpleBuilder 469–470
 - splitting message body 80
 - syntax 462
 - using as expression 69
 - using from Processor 469
 - using with
 - Content Based Router EIP 461
 - using with custom Java code 469
 - variables binding to Exchange 462
 - simple expression 195
 - simple language 233
 - Simple Network Management Protocol (SNMP) 386, 388
 - Simple Object Access Protocol. *See* SOAP
 - <simple> tag 260

- simple, invoke method on message body 260
 - SimpleDateFormat 195
 - SimpleRegistry registry 99–100
 - SOA 205
 - SOAP 138, 205, 211
 - SoapUI 433
 - SonicMQ 42
 - split big file 318
 - <split> tag 261, 264
 - Splitter EIP 238, 255–266
 - aggregate 258–264
 - AggregationStrategy 262
 - big message 260–262
 - streaming mode 261–262
 - using stream 260
 - combine message 262
 - Exchange properties 258
 - Expression 258
 - handling error 264–266
 - by AggregationStrategy 265
 - by stopping 264
 - in AggregationStrategy 264
 - stopOnException 264
 - iterate 257–258
 - message body 256
 - overview 255
 - split 79–80
 - split complete 258
 - split index 258
 - split size 258
 - splitting message body 79
 - tokenizer 261
 - using 256–257
 - beans for splitting 258–260
 - Expression 256
 - using java.util.Scanner 261
 - Spring 34, 361–362, 411
 - bean wiring example 34
 - BeanPostProcessor 447
 - ClassPathXmlApplicationContext 35, 160
 - configuring Camel
 - components 36
 - context listener 431
 - defining bean in Spring
 - XML 95
 - invoking a bean 95
 - JMX 407
 - @ManagedAttribute 407
 - @ManagedResource 407
 - loading CamelContext in
 - Spring XML 36
 - property placeholder 161
 - referencing a
 - RouteBuilder 37
 - RouteBuilder 95
 - separate wiring into several
 - XML files 42
 - Spring JMX 389
 - Spring Remoting 456
 - Spring transaction 290–291
 - wiring 35
 - Spring in Action* 34
 - Spring property
 - placeholders 164–166
 - spring.handlers 412
 - SpringCamelContext 447
 - SQL 222, 228
 - SSL 196
 - staged event-driven architecture
 - (SEDA) 322, 340, 342
 - concurrency 322–323
 - concurrent consumer 323
 - starting
 - Camel 411–419
 - autostartup 412
 - startup diagram 412
 - reliable startup 411
 - route startup order 416
 - StartupOrder 416
 - StreamCaching 344, 414
 - synchronicity 335–344
 - eip threads 339
 - Synchronization 311
 - using 310–312
 - synchronous 335
 - synchronous messaging 229
-
- T**
- template, transforming using
 - JAXB 76
 - Test Kit 8
 - testing
 - Camel test kit 155, 166
 - CamelSpringTestSupport 156, 304
 - CamelTestSupport 156
 - createRouteBuilder 157
 - existing RouteBuilder
 - class 159
 - Maven dependencies 156
 - RouteBuilder 157
 - SpringCamelTestSupport 159
 - @Test 157
 - TestSupport 156
 - unit testing route 156
 - using SpringCamelTestSupport 159–161
 - camel-test.jar 155
 - externalize dynamic parts 161
 - file copy example 157–159
 - in multiple
 - environments 161–166
 - reusable unit test 162
 - integration test 182–185
 - NotifyBuilder 185–187
 - message ordering 174
 - expects 175
 - expectsAscending 174
 - expectsDescending 174
 - gap detection 174
 - using custom
 - expression 174–175
 - MockEndpoint 167
 - NotifyBuilder 184
 - from a specific
 - endpoint 185
 - matches 186
 - table of methods 185
 - understanding message
 - complete 186
 - using predicate 185
 - whenAnyDoneMatches 185
 - whenDone 186
 - whenFailed 186
 - replacing JMS with SEDA 169
 - simulating error 178
 - communication error 180
 - connection error 288
 - interceptSendToEndpoint 181, 288
 - skipSendToOriginalEndpoint 181
 - using interceptor 180–183
 - using mock 180
 - using processor 178–180
 - transaction 293–295
 - database 293
 - JMS broker 293
 - unit test 155
 - using interceptor
 - adviceWith 182, 289
 - wildcard 182
 - using mock 166
 - AssertionError 168
 - assertIsNotSatisfied 175
 - assertIsSatisfied 168, 175
 - assertMockEndpointsIsSatisfied 171

- testing (*continued*)
 - expectation 167
 - expectedBodiesReceived 168, 170
 - expectedMessageCount 168–169
 - expectedMinimumMessageCount 168
 - expression-based methods 171
 - getReceivedExchanges 171
 - message 172
 - regular expression 173
 - setResultWaitTime 169
 - simulating real component 175–177
 - table of predicate methods 172
 - verify message content 169
 - whenAnyExchangeReceived 176–177, 180
 - whenExchangeReceived 176
 - with expression 170–174
 - using ProducerTemplate 158, 162
 - using Spring property placeholders 164
 - verify message content 169
 - without mock 183
- thread pool 319
 - benefits by Camel 325
 - blocked thread 351–352
 - cached thread pool 321
 - concurrency 323
 - creating 320
 - custom pool 334
 - Java DSL 328
 - Spring XML 328
 - ThreadPoolBuilder 328
 - using Executors 325
 - custom profile 326
 - configuring 327–328
 - idle time 327
 - ThreadPoolProfileSupport 327
 - custom thread pool 320–322
 - default profile 326–327
 - configuring 326–327
 - getDefaultThreadPoolProfile 326
 - settings 326
 - eip
 - splitter 320
 - threads 331–332
 - execute task 323
 - ExecutorService 320, 324
 - executorServiceRef 327–328, 334
 - ExecutorServiceStrategy 325–326, 329
 - custom 329
 - using in custom component 329–330
 - exhausted 324, 326
 - fixed thread pool 321
 - idle time 324, 326, 328
 - JMX 325
 - list of options 324
 - lookup 327
 - executorServiceRef 327–328
 - maximum pool size 324, 326–328, 334
 - newScheduledThreadPool 329
 - parallelProcessing 331
 - pool size 324, 326, 328, 331, 334
 - profile 327–328
 - rejected policy 326
 - schedule task 329–330
 - scheduled background task 329
 - ScheduledExecutorService 329
 - shutdown 325
 - sizing 322
 - task queue 323, 325
 - task queue size 326
 - thread blocking 353–354
 - thread factory 325
 - thread name 320
 - human understandable 325
 - ThreadPoolBuilder 334
 - ThreadPoolExecutor 324
 - understanding 323–326
 - unique thread names 325
 - using 323–330
 - WorkManager 323
 - threading model 335
 - Throttler EIP 134
 - timer 192
 - TLS 196
 - <to> tag 165
 - tooling 8
 - topic 26
 - TopLink 224
 - tracer 398–401
 - customizing 399
 - enable 398
 - managing using JMX 400–401
 - noTracing 400
 - per route 400
 - trace logs 398
 - traceFormatter 399
 - tracing 400
 - <transacted> tag 302
 - transaction
 - ActiveMQXACConnectionFactory 299
 - begin 289, 291
 - boundary 296
 - commit 289, 291, 296–297, 304
 - compensation 309
 - on completion 309
 - rollback 309
 - synchronization 310
 - configuring 301–303
 - Camel route 292
 - convention over configuration 302
 - default configurations 293, 302
 - Spring XML 291–292
 - declarative 290
 - EIP, transactional client 296
 - example 291–295
 - configuring 291
 - global transaction 298–301
 - JMS, transacted acknowledge mode 289, 292
 - JmsTransactionManager 291, 297
 - timeout 292
 - JPA
 - EntityManager 289
 - EntityTransaction 289
 - JtaTransactionManager 298–299
 - local transaction 295, 297
 - locally managed 289
 - losing message 288–289, 291, 297
 - markRollbackOnly 306–307
 - markRollbackOnlyLast 306
 - multiple resource 297–298
 - multiple transaction 305
 - multiple transaction manager 292
 - propagation 337, 340, 342
 - redelivery 294
 - JMSRedelivered 295

transaction (*continued*)
 rollback 289, 291, 298, 300, 304, 306–307
 return custom
 response 306
 route
 convention over
 configuration 292
 transacted 292
 runtime environment
 agnostic 290
 single resource 297
 spring transaction
 JmsTransactionManager 291
 TransactionManager 290
 transacted 292–293, 301–303, 305–306
 transaction manager,
 suspension 306
 transaction propagation 301
 required 301–302, 305
 requires new 301, 305
 SpringTransactionPolicy 302
 with multiple routes 303
 XA standard protocol 299
 transaction, error handling
 onException 307
 Transactional Client EIP 296
 transform
 XML to SQL 285–286
 XML, @XPath 286
 transform, losing message 293
 transform()
 from Java 67
 from Spring XML 69
 Transmission Control Protocol (TCP) 216
 endpoint 145, 148
 type converter 7, 62
 AnnotationTypeConverter-
 Loader 89
 convertBodyTo 90

@Converter 89
 convertTo 89, 159
 data type transformation
 using 88–91
 encoding 90
 getBody 88
 IOConverter example 89
 loading 89
 TypeConverter 89
 TypeConverterRegistry 88
 understanding 88
 using 90
 writing custom 90
 type converters 446

U

unit of work 309
 boundaries 310
 introducing 309–310
 synchronization 309
 adding 311
 UnitOfWorkProcessor 310
 User Datagram Protocol (UDP) 216

V

Velocity
 endpoint 88
 generating email content 87
 loading templates 88
 table of information in
 VelocityContext 87
 template 87

W

Walls, Craig 34
 WAR 362
 web console 362
 web services
 contract 205

service endpoint interface (SEI) 205
 with transaction 306
 Web Services Description Language (WSDL) 205, 307
 WEB-INF directory 431
 @WebService 215
 website links for
 components 189
 wire tap 55
 Woolf, Bobby 5
 writing files 194
 WSDL 205, 209
 binding 211
 example 209
 message 211
 other bindings and
 encodings 211
 portType 211
 service 211
 tools that help generate
 WSDLs 210
 WSDL 2.0 support 210

X

XML
 object marshaling 75
 serializing objects to and from
 XML 75–76
 transforming 73–77
 transforming using XSLT 73
 XPath 116
 with namespace 118
 XML schema 211
 @XPath 54, 116, 223, 286
 XPath 45, 50, 223
 @XQuery 116
 XQuery 45
 XSLT, loading stylesheets 73