

NUMERICS

- 1:N (one-to-many) relationships, 411, 416–420
- 1:1 (one-to-one) relationships, 411–416
- 2PC (two-phase commit) protocol, 340–342

A

- Aardvark Knowledge Builder legacy data integration tool, 472
 - aborting transactions, 329
 - abstract entities, 39
 - abstract persistence schema (CMP entity beans), 186–187
 - accessing databases, 13
 - `AccessLocalException` exception, 794
 - ACID properties (transactions), 304–306
 - activation
 - entity beans, 129
 - stateful session beans, 83–84, 86–88
 - stateless session beans, 84
 - Active Directory (Microsoft), 69
 - administration of EJB servers, 616
 - aggregate functions (EJB-QL), 751–752
 - aggregation relationship, 434–436
 - agreement between application server and components, 7
 - Ambler, Scott W., *The Unified Process Inception Phase*, 606
 - Anderson, Ross, *Security Engineering*, 350
 - Ant build tool (Apache group), 607
 - AOP (Aspect-Oriented Programming), 284–287
 - application assembler, 16–17
 - application integration
 - defined, 478
 - infrastructure services problem, 482–483
 - J2EE Connector Architecture, 479–480, 542
 - Java Message Service (JMS), 479, 542
 - M x N integration problem, 480–481
 - message-driven beans, 479, 542
 - proprietary solutions, 480–481
 - Web Services, 479, 543
 - application logic components, 122
 - application servers
 - agreement with components, 7
 - choosing, 603
 - component architecture, 7
 - deploying EJB applications, 288–289
 - J2EE application servers, 18
 - market for, 7
 - middleware, 5
 - application-level exceptions, 58–59
 - APS (arrivals per second), 573
 - architectures
 - collocated, 573–577
 - component, 7, 9
 - distributed, 573–577
 - J2EE Connector Architecture, 479–480, 483, 542
 - Service-Oriented Architecture (SOA), 8–9, 103–105
 - arrivals per second (APS), 573
 - Ascential Software DataStage legacy data integration tool, 472
 - Aspect-Oriented Programming (AOP), 284–287
 - assembly descriptor, 732–737
 - asynchronous method invocation, 254
 - asynchronous programming, 218
 - atomic operations, 300–301
 - attribute-oriented programming, 288
 - auditing, 6
 - authentication
 - defined, 352
 - Java Authentication and Authorization Service (JAAS), 357–368
 - Web applications, 354–355
 - authorization
 - declarative, 352, 368, 373–376
 - defined, 352
 - instance-level, 376
 - programmatic, 352, 368–373, 376–377
 - security roles, 368
 - Web applications, 355
 - availability
 - EJB servers, 613
 - large business systems, 571
 - AVG aggregate function (EJB-QL), 751
- ## B
- back-end integration, 5
 - bank account example
 - `AccountBean.java`, 158–170
 - `AccountException.java`, 170
 - `AccountHome.java` home interface, 153
 - `Account.java` remote interface, 151–152
 - `AccountLocalHome.java` local home interface, 155–156
 - `AccountLocal.java` local interface, 152–155
 - `AccountPK.java`, 156–157
 - class diagram, 150
 - client program, 175–177
 - `Client.java`, 171–173
 - container-specific deployment descriptor, 175
 - deployment descriptor, 173–174
 - Batch utility (Unix), 392
 - BEA WebLogic
 - J2EE application server, 18
 - Workshop, 19

- bean class
 - defined, 35–36
 - entity beans, 124–125
 - Hello World example, 61–62
 - message-driven beans, 234–236
 - bean failure, 549
 - bean instance pooling, 42
 - bean instances, 46
 - bean provider, 16
 - bean-independent resources, 553–554
 - bean-managed persistence (BMP), 131
 - bean-specific resources, 553–554
 - begin()* method, 326, 799
 - benchmarks for SPECjAppServer, 567
 - best practices
 - Aspect-Oriented Programming (AOP), 284–287
 - client-side callbacks, 282–283
 - code reuse, 292–293
 - debugging, 290–291
 - deploying EJB applications, 288–289
 - Extreme Programming (XP), 277–279
 - integration, 541
 - method invocation, 287–288
 - Middle Driven Development (MDD), 275–276
 - RMI-IIOP versus messaging, 294–297
 - servlets, 284
 - singletons, 293
 - unit testing, 279–282
 - Web application frameworks, 272–274
 - when to use EJB, 270–271
 - XML, 293–294
 - BMP (bean-managed persistence), 131
 - BMP entity beans
 - bank account example
 - AccountBean.java*, 158–170
 - AccountException.java*, 170
 - AccountHome.java* home interface, 153
 - Account.java* remote interface, 151
 - AccountLocalHome.java* local home interface, 155–156
 - AccountLocal.java* local interface, 152–153
 - AccountPK.java*, 156–157
 - class diagram, 150
 - client program, 175–177
 - Client.java*, 171–173
 - container-specific deployment descriptor, 175
 - deployment descriptor, 173–174
 - bPassivate()* method, 147
 - bugs, 451
 - code reduction, 450
 - control, 451
 - cost, 452
 - differences from CMP entity beans, 181–186, 410
 - directionality of relationships, 429–430
 - ejbActivate()* method, 146
 - ejbCreate()* method, 145
 - ejbFind()* method, 144
 - ejbHome()* method, 145
 - ejbLoad()* method, 146
 - ejbPostCreate()* method, 146
 - ejbRemove()* method, 147
 - ejbStore()* method, 147
 - fake many-to-many (M:N) relationships, 423–424
 - learning curve, 452
 - life cycle, 177–180
 - one-to-many (1:N) relationships, 418–419
 - one-to-one (1:1) relationships, 413–414
 - performance, 450
 - referential integrity, 440
 - relationships, 452
 - setEntityContext()* method, 144, 779
 - true many-to-many (M:N) relationships, 426–427
 - unsetEntityContext()* method, 147
- Borland JBuilder, 19
- bPassivate()* method, 147
- Brose, Vogel, and Duddy, *Java Programming With CORBA*, 688
- bugs in CMP/BMP entity beans, 451
- business interface, 74–75
- business logic, 13
- business logic tier, 630–631
- business process integration, 479
- business requirements, 593–594
- ## C
- caching
 - large business systems, 6
 - persistence, 448
 - callbacks (client-side), 282–283
 - calling beans from other beans
 - EJB references, 257–259
 - environment properties, 262–263
 - handles, 263–265
 - JNDI lookups, 256–257
 - layers, 255–256
 - resource factories, 259–262
 - capacity planning, 549–550
 - cardinality, 411
 - CCI (Common Client Interfaces), 483
 - centralized naming servers, 589–590
 - certified message delivery, 220
 - chained transactions, 310
 - circular relationships, 438–439
 - CLASSPATH directory, 566
 - clean shutdown, 6, 617
 - CleanDayLimitOrdersEJB* timer example
 - bean class, 401–403
 - clients, 404–406
 - deployment descriptor, 403–404
 - features, 399
 - home interface, 403
 - remote interface, 400
 - client code
 - product line example, 212–214
 - transactions, 330–331
 - client-initiated transactions, 313–316
 - clients
 - CleanDayLimitOrdersEJB* timer example, 404–406
 - CORBA clients, 68
 - distributed objects, 30–31
 - dynamically generated Web pages, 14
 - Java RMI-IIOP-based, 68
 - thick clients, 14
 - Web Services, 14–15, 114–116
 - client-side callbacks, 282–283
 - client-side output, 73
 - clustering
 - arrivals per second (APS), 573
 - clusters (defined), 572
 - EJB servers, 616
 - Enterprise JavaBeans (EJB), 578–580
 - entity beans, 584–588
 - fail-overs, 572
 - highly available systems, 572
 - idempotent, 579–580
 - invocations per second (IPS), 573
 - large business systems, 5, 569–571
 - load balancing, 572
 - logic, 578
 - message-driven beans, 243–244, 588–590
 - naming servers, 589–590
 - node, 572
 - partitioning clusters, 573–577
 - redundancy, 571
 - requests per second (RPS), 573
 - session beans, 581–584
 - single access point simplicity, 572
 - throughput, 573
 - transactions per second (TPS), 573
 - CMP (container-managed persistence), 132
 - CMP entity beans
 - abstract persistence schema, 186–187
 - application server and database independence, 451–452
 - bugs, 451
 - code reduction, 450
 - control, 451
 - cost, 452
 - dependent value classes, 453–454
 - differences from BMP entity beans, 181–186, 410
 - directionality of relationships, 430–431
 - ejbActivate()* method, 194
 - ejbCreate()* method, 193
 - ejbFind()* method, 192
 - ejbHome()* method, 193
 - ejbLoad()* method, 194

- ejbPassivate()* method, 195
 - ejbPostCreate()* method, 194
 - ejbRemove()* method, 195
 - ejbSelect()* method, 189–190, 192
 - ejbStore()* method, 195
 - fake many-to-many (M:N) relationships, 424–426
 - get/set methods, 184–185
 - learning curve, 452
 - life cycle, 214–215
 - one-to-many (1:N) relationships, 419–420
 - one-to-one (1:1) relationships, 414–416
 - performance, 450
 - product line example, 196–214
 - query language, 187–188
 - rapid application development, 450
 - referential integrity, 440
 - relationships, 452
 - setEntityContext()* method, 191, 779
 - subclassing, 181–183
 - true many-to-many (M:N) relationships, 427–428
 - unsetEntityContext()* method, 195
 - code
 - glue-code tools, 39
 - thread-safe, 226
 - code reuse, 292–293, 601
 - code testing, 279–282
 - collocated architecture, 573–577
 - commit()* method, 326, 799
 - Common Client Interfaces (CCI), 483
 - Common Object Request Broker Architecture (CORBA). *See* CORBA (Common Object Request Broker Architecture)
 - Common Secure Interoperability version 2 (CSIv2), 379–381
 - component architecture, 7, 9
 - component interfaces
 - implementing, 73–75
 - message-driven beans, 228
 - components
 - application logic, 122
 - persistent data, 122–123
 - composition relationship, 434–436
 - concurrency control (transactions)
 - dirty reads, 334–335
 - need for, 332–333
 - optimistic, 339
 - pessimistic, 339
 - phantom problem, 336–337
 - unrepeatable reads, 336–337
 - Connection interface, 225
 - connection management contract, 484, 495–498
 - connection pooling, 260
 - ConnectionFactory instance (JMS), 222
 - ConnectionFactory interface, 225
 - connections (JMS), 222
 - Connector Architecture. *See* J2EE Connector Architecture
 - connectors. *See* J2EE Connector Architecture
 - Constantine, Larry L.
 - Constantine on Peopeware*, 605
 - The Unified Process Inception Phase*, 606
 - consumers (JMS), 223
 - container-managed persistence (CMP), 132
 - containers
 - built-in thread support, 38
 - debug logs, 73
 - defined, 18
 - EJB timer service, 394–395
 - glue-code tools, 39
 - layer of indirection, 38
 - life cycle of deployed beans, 42
 - management methods, 39
 - monitoring, 38
 - resource management, 42
 - transactions, 37, 310, 317–318, 330
 - container-specific deployment descriptor
 - bank account example, 175
 - product line example, 210–211
 - contexts
 - defined, 62–64
 - entity beans, 137–138
 - message-driven contexts, 786
 - security contexts, 377–378
 - session contexts, 92, 787
 - transactional contexts, 342
 - controls (security), 351–353
 - conversational state
 - Java object serialization, 84
 - session beans, 84–85
 - conversations, defined, 80
 - conversion tools, 610
 - CORBA (Common Object Request Broker Architecture)
 - advantages, 684–685
 - basis of EJB, 684
 - clients, 68
 - CORBA Component Model (CCM), 691
 - CORBA Naming Service (COS Naming), 691, 699–700
 - CORBA-EJB interoperability, 700–703
 - differences from RMI, 695–696
 - disadvantages, 685
 - distributed objects, 31
 - Internet Inter-ORB Protocol (IIOP), 683–684, 697
 - interoperability with RMI, 692–694, 698–699
 - invocation process, 689–690
 - Java-to-IDL Mapping, 696–697
 - middleware, 684
 - object adapter, 690
 - Object Request Broker (ORB), 686–687
 - Object Transaction Service (OTS), 324–325
 - OMG interface definition language (OMG IDL), 687–689
 - OMG Web site, 688
 - Portable Object Adapter (POA), 690
 - services, 691
 - uses, 684–685
 - Core J2EE Patterns, John Crupi, et al., 600
 - CosTransactions interfaces, 324
 - CosTSPortability interface, 325
 - COUNT aggregate function (EJB-QL), 751
 - count bean
 - client code, 94–97
 - client-side output, 97
 - CountBean.java code, 90–92
 - defined, 88
 - deployment descriptor, 93–94
 - Ejb-jar file, 94
 - home interface, 92–93
 - proprietary descriptor, 94
 - remote interface, 90
 - server-side output, 97–98
 - create()* method, 60
 - CreateException exception, 794
 - createTimer()* method, 395, 794
 - Cron jobs, 392
 - CRUD operations (session beans), 551
 - Crupi, John, *Core J2EE Patterns*, 600
 - cryptography, 383–386
 - CSIv2 (Common Secure Interoperability version 2), 379–381
- D**
- data confidentiality protection
 - defined, 353
 - Web applications, 356
 - data integrity protection
 - defined, 352
 - Web applications, 356
 - data model, 459
 - data tier, 631
 - database access, 13
 - databases
 - impact of updates, 461
 - large result sets, 474–475
 - legacy databases
 - common design problems, 463–471
 - integration tools, 472
 - migration strategies, 472–474
 - DataStage (Ascential Software) legacy data integration tool, 472
 - Date, C.J., *An Introduction to Database Systems, 7th Edition*, 458
 - DCOM (Microsoft), 31
 - deadlocks, 333
 - debug logs, 73

- debugging, 290–291, 451
 - declarative authorization, 352, 368, 373–376
 - declarative (implicit) middleware, 33–35
 - declarative transactions, 312–314, 328
 - decompilers, 291
 - deferred database writes, 611
 - Demarco, Tom, *Peopleware: Productive Project and Teams, 2nd Edition*, 605
 - denormalization, 457–459
 - dependent value classes, 453–454
 - deploying
 - EJB applications, 288–289
 - Enterprise JavaBeans (EJB), 66–67
 - Web services, 113–114
 - deployment descriptor
 - assembly descriptor, 732–737
 - assembly-descriptor* element, 733–734
 - autogenerating, 49
 - bank account example, 173–174
 - bean management, 48
 - CleanDayLimitOrdersEJB* timer example, 403–404
 - cmp-field* element, 713–714
 - cmr-field* element, 731–732
 - container-transaction* element, 735
 - count bean, 93–94
 - defined, 48–49, 51
 - EJB references, 257–259, 722–725
 - Ejb-client* jar file, 67–68
 - ejb-jar* element, 707–708
 - ejb-local-ref* element, 724–725
 - ejb-ref* element, 723–724
 - ejb-relation* element, 730
 - ejb-relationship-role* element, 730–731
 - entity beans, 710–715
 - entity* element, 710–714
 - env-entry* element, 722
 - environment properties, 262–263, 721–722
 - exclude-list* element, 735–736
 - header element, 706–707
 - Hello World example, 64–65
 - jaxrpc-mapping-file* element, 720
 - lifecycle requirements, 48
 - message-destination-type* element, 718
 - message-driven beans, 236–241, 716–718
 - message-driven* element, 716–717
 - messaging-type* element, 718
 - method* element, 736
 - method-params* element, 715, 737
 - method-permission* element, 734–735
 - OutboundLoanRA example, 534–535
 - persistence requirements, 48
 - port-component* element, 721
 - product line example, 207–210
 - query* element, 714
 - query-method* element, 715
 - relationship-role-source* element, 731
 - relationships, 729–732
 - relationships* element, 730
 - resource factories, 260–261, 727–728
 - resource-env-ref* element, 728
 - resource-ref* element, 727–728
 - root element, 706–707
 - run-as* element, 726
 - security, 725–726
 - security requirements, 48–49
 - security-identity* element, 726
 - security-role* element, 734
 - security-role-ref* element, 725
 - session beans, 708–710
 - session* element, 708–710
 - timers, 718
 - transaction requirements, 48
 - Web Services, 719–721
 - webservice-description* element, 719–720
 - webservices* element, 719
 - writing, 49
 - XML Schema definitions (XSD), 705–706
 - designing J2EE object model, 600
 - Destination* interface, 225
 - destinations (JMS), 222
 - developing EJBs
 - bean class, 61–62
 - deployment descriptor, 64–65
 - Ejb-jar* file, 65–66
 - Hello World object model, 55
 - home interface, 57–58
 - local home interface, 59–60
 - local interface, 56–57
 - remote interface, 55–56
 - step-by-step, 54
 - vendor-specific files, 65
 - diagrams
 - BMP entity beans, 178
 - CMP entity beans, 215
 - entity beans, 765–769
 - message-driven beans, 770–771
 - session beans, 758–764
 - directionality of relationships, 428–433
 - Directory Server (iPlanet), 69
 - directory services. *See* naming and directory services
 - dirty reads, 334–335
 - distributed architecture, 573–577
 - distributed objects
 - clients, 30–31
 - CORBA (OMG), 31
 - DCOM (Microsoft), 31
 - defined, 30
 - Java RMI-IIOP (Sun), 31
 - middleware, 32–35
 - stubs, 30–31
 - distributed systems, 4
 - distributed transactions, 340–342, 618
 - distribution transparency, 31–32
 - DNS load balancing, 590
 - document-style SOAP, 8
 - domains (messaging), 220–222
 - dooming transactions, 329
 - Duddy, Vogel, and Brose, *Java Programming With CORBA*, 688
 - DuplicateKey* exception, 795
 - durability of transactions, 340–341
 - dynamic proxy invocation, 287–288
 - dynamic redeployment, 5
 - dynamically generated Web pages, 14
- ## E
- Eclipse IDE, 19
 - Ecosystem
 - application assembler, 16–17
 - bean provider, 16
 - business models, 15
 - container, 18
 - EJB deployer, 17
 - integrated development environments (IDEs), 18–19
 - roles of participants, 19–20
 - server provider, 18
 - system administrator, 17–18
 - EIS (enterprise information system), 480
 - EIS-specific client interfaces, 483
 - EJB (Enterprise JavaBeans)
 - clustering, 578–580
 - CORBA-EJB interoperability, 684, 700–703
 - defined, 3, 27–28
 - deployment, 66–67
 - development
 - bean class, 61–62
 - deployment descriptor, 64–65
 - Ejb-jar* file, 65–66
 - Hello World object model, 55
 - home interface, 57–58
 - local home interface, 59–60
 - local interface, 56–57
 - remote interface, 55–56
 - step-by-step, 54
 - vendor-specific files, 65
 - entity beans, 28–29
 - industry agreement, 11
 - Java interfaces, 12
 - JMS-EJB integration, 226–227
 - message-driven beans, 29
 - portability, 11
 - rapid application development, 12

- security, 37, 353
- session beans, 28
- specification, 12
- when to use, 270–271
- EJB containers
 - built-in thread support, 38
 - debug logs, 73
 - defined, 18
 - EJB timer service, 394–395
 - glue-code tools, 39
 - layer of indirection, 38
 - life cycle of deployed beans, 42
 - management methods, 39
 - monitoring, 38
 - resource management, 42
 - transactions, 37, 310, 317–318, 330
- EJB context object, 62–64
- EJB deployer, 17
- EJB Design Patterns*, Floyd Marinescu, 269, 545, 600
- EJB Ecosystem
 - application assembler, 16–17
 - bean provider, 16
 - business models, 15
 - container, 18
 - EJB deployer, 17
 - integrated development environments (IDEs), 18–19
 - roles of participants, 19–20
 - server provider, 18
 - system administrator, 17–18
- EJB home handles, 264–265
- EJB object
 - bean instances, 46
 - container-specific code, 38
 - create()* method, 60
 - defined, 38–39, 51
 - home interface, 43–46
 - home objects, 42–43
 - remote interface, 39–40
 - remove()* method, 60
 - request interceptor role, 38
- EJB object factory, 43
- EJB object handles, 263–264
- EJB object model, 459
- EJB Query Language (EJB-QL)
 - aggregate functions, 751–752
 - CMP entity beans, 188
 - conditional expressions, 745–747
 - defined, 739–740
 - example, 740–741
 - FROM clause, 742–744
 - functions, 745
 - ORDER BY clause, 752–753
 - performance optimization, 754–755
 - relationships, 436–437, 741–742
 - SELECT clause, 748–751
 - syntax, 742–753
 - truth tables, 753–754
 - WHERE clause, 744, 748
- EJB references
 - calling beans from other beans, 257–259
 - deployment descriptor, 257–259, 722–725
- EJB servers
 - administration, 616
 - automatic EJB generation, 617
 - availability, 613
 - clean shutdown, 617
 - clustering, 616
 - compatibility, 610
 - complex mappings, 611
 - conversion tools, 610
 - deferred database writes, 611
 - distributed transactions, 618
 - hot deployment, 617
 - IDE integration, 614
 - in-memory data cache, 612
 - instance pooling, 617
 - integrated tier support, 612
 - Java Management Extension (JMX), 616
 - Java Runtime Environment (JRE), 610
 - JDBC driver support, 611
 - lazy loading, 611
 - load balancing, 615
 - messaging, 618
 - nontechnical criteria, 621
 - open source, 620
 - performance optimization, 567
 - pluggable persistence providers, 611–612
 - provided EJB components, 619
 - real-time deployment, 618
 - scalability, 612–613
 - security, 613–614
 - special features, 620–621
 - SPECjAppServer benchmarks, 567
 - technical support, 621
 - training, 621
 - transparent fail-over, 58–59, 615
 - UML editor integration, 615
 - Web Services, 619
 - workflow engines, 619–620
- EJB timer service
 - strengths, 406–407
 - timer service APIs, 394–398
 - weaknesses, 407
- ejbActivate()* method
 - BMP entity beans, 146, 782
 - CMP entity beans, 194, 782
 - entity beans, 129
 - session beans, 86–89, 788
- Ejb-client jar file, 67–68
- EJBContext* object, 772–773
- ejbCreate()* method
 - BMP entity beans, 145, 780
 - CMP entity beans, 193, 780
 - entity beans, 132–134
 - message-driven beans, 232, 786, 788
 - session beans, 89, 788
- EJBException exception, 795
- ejbFind()* method, 144, 192, 779
- ejbHome()* method, 145, 193, 780
- EJBHome* object, 773–774
- Ejb-jar file, 49–51, 65–66, 94
- ejbLoad()* method
 - BMP entity beans, 146, 782
 - CMP entity beans, 194, 782
 - entity beans, 126–127, 139
- EJBLocalHome* object, 774
- EJBLocalObject* object, 775
- EJBMetaData* object, 775–776
- EJBObject* object, 776–777
- ejbPassivate()* method
 - BMP entity beans, 781
 - CMP entity beans, 195, 781
 - entity beans, 86–87, 89
 - session beans, 130–131, 788
- ejbPostCreate()* method, 146, 194, 781
- EJB-QL (EJB Query Language). *See* EJB Query Language (EJB-QL)
- ejbRemove()* method
 - BMP entity beans, 147, 783
 - CMP entity beans, 195, 783
 - entity beans, 89, 133–135, 139
 - message-driven beans, 232, 245, 786
 - session beans, 101, 788
- ejbSelect()* method, 189–190, 192, 779
- ejbStore()* method
 - BMP entity beans, 147, 781
 - CMP entity beans, 195, 781
 - entity beans, 126–127
- ejbTimeout()* method, 397, 399
- encryption, 383–386
- end-to-end security, 382–383
- enterprise bean class. *See* bean class
- enterprise bean instance, 50–51
- enterprise information system (EIS), 480
- Enterprise JavaBeans (EJB)
 - clustering, 578–580
 - CORBA-EJB interoperability, 684, 700–703
 - defined, 3, 27–28
 - deployment, 66–67
 - development
 - bean class, 61–62
 - deployment descriptor, 64–65

- Enterprise JavaBeans (EJB), development (*continued*)
 - Ejb-jar file, 65–66
 - Hello World object model, 55
 - home interface, 57–58
 - local home interface, 59–60
 - local interface, 56–57
 - remote interface, 55–56
 - step-by-step, 54
 - vendor-specific files, 65
- entity beans, 28–29
- industry agreement, 11
- Java interfaces, 12
- JMS-EJB integration, 226–227
- message-driven beans, 29
- portability, 11
- rapid application development, 12
- security, 37, 353
- session beans, 28
- specification, 12
- when to use, 270–271
- Enterprise Security with EJB and CORBA*, Bret Hartman et al., 350
- EnterpriseBean* object, 777
- entity beans
 - bean-managed persistence, 131
 - BMP entity beans
 - bank account example, 150–177
 - bPassivate()* method, 147
 - bugs, 451
 - code reduction, 450
 - control, 451
 - cost, 452
 - differences from CMP entity beans, 181–186
 - directionality of relationships, 429–430
 - ejbActivate()* method, 146
 - ejbCreate()* method, 145
 - ejbFind()* method, 144
 - ejbHome()* method, 145
 - ejbLoad()* method, 146
 - ejbPostCreate()* method, 146
 - ejbRemove()* method, 147
 - ejbStore()* method, 147
 - fake many-to-many (M:N) relationships, 423–424
 - learning curve, 452
 - life cycle, 177–180
 - one-to-many (1:N) relationships, 418–419
 - one-to-one (1:1) relationships, 413–414
 - performance, 450
 - referential integrity, 440
 - relationships, 410, 452
 - setEntityContext()* method, 144, 779
 - true many-to-many (M:N) relationships, 426–427
 - unsetEntityContext()* method, 147
 - characteristics, 125–126
 - client interaction, 30
 - clustering, 584–588
 - CMP entity beans
 - abstract persistence schema, 186–187
 - application server and database independence, 451–452
 - bugs, 451
 - code reduction, 450
 - control, 451
 - cost, 452
 - dependent value classes, 453–454
 - differences from BMP entity beans, 181–186
 - directionality of relationships, 430–431
 - ejbActivate()* method, 194
 - ejbCreate()* method, 193
 - ejbFind()* method, 192
 - ejbHome()* method, 193
 - ejbLoad()* method, 194
 - ejbPassivate()* method, 195
 - ejbPostCreate()* method, 194
 - ejbRemove()* method, 195
 - ejbSelect()* method, 189–190, 192
 - ejbStore()* method, 195
 - fake many-to-many (M:N) relationships, 424–426
 - get/set methods, 184–185
 - learning curve, 452
 - life cycle, 214–215
 - one-to-many (1:N) relationships, 419–420
 - one-to-one (1:1) relationships, 414–416
 - performance, 450
 - product line example, 196–214
 - query language, 187–188
 - rapid application development, 450
 - referential integrity, 440
 - relationships, 410, 452
 - setEntityContext()* method, 191, 779
 - subclassing, 181–183
 - true many-to-many (M:N) relationships, 427–428
 - unsetEntityContext()* method, 195
 - contexts, 137–138
 - data instances, 124
 - defined, 28–29, 119, 124
 - deployment descriptor, 710–715
 - diagrams, 765–769
 - ejbActivate()* method, 129, 782
 - ejbCreate()* method, 132–134, 780
 - ejbFind()* method, 779
 - ejbHome()* method, 780
 - ejbLoad()* method, 126–127, 139, 782
 - ejbPassivate()* method, 130–131, 781
 - ejbPostCreate()* method, 781
 - ejbRemove()* method, 133–135, 139, 783
 - ejbSelect()* method, 779
 - ejbStore()* method, 126–127, 781
 - enterprise bean class, 36
 - entity bean class, 124–125
 - failures, 125
 - finder methods, 136, 143–150
 - getEJBLocalObject()* method, 138, 784
 - getEJBObject()* method, 138, 784
 - getPrimaryKey()* method, 138–139, 784
 - granularity, 453–454
 - instances, 124, 126–130
 - javax.ejb.EntityBean* interface, 36, 141–143
 - large result sets, 474–475
 - life cycle, 125
 - lifetime, 123
 - modifying data, 136–137
 - persistence
 - bean-managed, 131
 - container-managed, 132
 - defined, 123–124
 - pooling, 128–130
 - primary key class, 125
 - session beans, 29, 123–124
 - SetEntityContext (EntityContext ctx)* method, 779
 - transactions, 315
 - tuning, 556–562
 - unsetEntityContext()* method, 783
- EntityBean* object, 777–778
- EntityContext* object, 778–784
- environment properties
 - calling beans from other beans, 262–263
 - deployment descriptor, 262–263, 721–722
- ETI*Extract legacy data integration tool, 472
- exceptions
 - AccessLocalException*, 794
 - application-level, 58–59
 - CreateException*, 794
 - DuplicateKey*, 795
 - EJBException*, 795
 - FinderException*, 795
 - NoSuchEntityException*, 795
 - NoSuchObjectLocalException*, 795
 - ObjectNotFoundException*, 795
 - remote, 58
 - RemoveException*, 795
 - system-level, 58–59
 - throwing, 58–59
 - TransactionRequiredLocalException*, 796
 - TransactionRolledBackLocalException*, 796
 - unchecked, 59
- explicit middleware, 32–33
- Extreme Programming (XP), 277–279
- ExtremeProgramming.Org Web site, 606

F

fail-overs, 572
 failures
 entity beans, 125
 session beans, 549
 transactions, 301–302
 fat key pattern, 450
 files
 deployment descriptor, 49, 51, 64–65
 Ejb-client jar, 67–68
 Ejb-jar, 49–51, 65–66
 vendor-specific, 49, 51, 65
 Web Services Definition Language (WSDL) file, 104
 finder methods (entity beans), 136, 143–150
 FinderException exception, 795
 firewalls, 577
 flag for switching between local and remote access to
 beans, 47–48
 flat transactions, 306–308
 Forte for Java, 19
 Fowler, Martin, *Refactoring: Improving the Design of Existing Code*, 278
 frameworks
 choosing, 272–273
 Hibernate, 122, 272
 integration with EJB, 273
 open source versus closed source, 274
 small device support, 273
 Spring, 272
 standards support, 274
 StrutsEJB project, 273
 tools support, 273
 unit testing, 280–281
 Wafer project, 274
 XDoclet, 289–290

G

getEJBLocalObject() method, 138, 784, 789
 getEJBObject() method, 138, 784, 789
 getHandle() method, 264
 getInfo() method, 396
 getMessageContext() method, 789
 getPrimaryKey() method, 138–139, 784
 getStatus() method, 326–327, 799
 getTimers() method, 395, 794
 getTimerService() method, 398
 glue-code tools, 39
 granularity of entity beans, 453–454
 guaranteed message delivery, 220

H

handles, 263–265, 784
 hard-coded SQL, 454–455
 hardware proxies, 590
 Harrison, Neil B., *Organizational Patterns for Teams*, 606
 Hartman, Bret, *Enterprise Security with EJB and CORBA*, 350
 helper code, 39
 Hibernate framework, 122, 272
 highly available systems, 572
 home handles, 264–265, 784–785
 home interface
 bank account example, 153
 CleanDayLimitOrdersEJB timer example, 403
 count bean, 92–93
 defined, 51
 Hello World example, 57–58
 problems, 44–46
 rules, 43–44
 home objects, 43, 69–72
 hot deployment, 617
 HTTP servlets, 637–639

I

IBM
 Lotus Notes Domino Server, 69
 WebSphere application server, 18
 WebSphere Studio Application Developer, 19
 IDE (integrated development environment), 18–19, 614
 idempotent, 579–580
 IDL (interface definition language), 687–689
 IIOP (Internet Inter-ORB Protocol), 683–684, 697

impedance mismatch, 454
 implementing
 component interfaces, 73–75
 Web services, 110–111
Implementing Enterprise Web Services JSR 921
 specification, 110
 implicit middleware, 33–35
 infinite block problem, 316
 Informatica PowerCenter legacy data integration tool, 472
 infrastructure services problem, 482–483
 inheritance, 291–293
 initial vertical slice, 601–602
 instance pooling, 42, 617
 instance-level authorization, 376
 integrated development environment (IDE), 18–19, 614
 integration
 application integration
 defined, 478
 infrastructure services problem, 482–483
 J2EE Connector Architecture, 479–480, 542
 Java Message Service (JMS), 479, 542
 M x N integration problem, 480–481
 message-driven beans, 479, 542
 proprietary solutions, 480–481
 Web Services, 479, 543
 benefits, 478
 best practices, 541–542
 business process integration, 479
 existing applications, 14
 importance of, 477–478
 JMS-EJB, 226–227
 RMI-IIOP and JNDI, 679–681
 interface definition language (IDL), 687–689
 interfaces
 business, 74–75
 Common Client Interfaces (CCI), 483
 component
 implementing, 73–75
 message-driven beans, 228
 Connection, 225
 ConnectionFactory, 225
 CosTransactions, 324
 CosTSPortability, 325
 Destination, 225
 EIS-specific client interfaces, 483
 home interface
 bank account example, 153
 CleanDayLimitOrdersEJB timer example, 403
 count bean, 92–93
 defined, 51
 Hello World example, 57–58
 problems, 44–45
 rules, 43–46, 51
 java.io.Serializable, 84–85
 java.rmi.Remote, 41
 java.rmi.RemoteException, 41
 javax.coordination.ServiceManager, 346
 javax.ejb.EJBContext, 63, 772–773
 javax.ejb.EJBHome, 44, 773–774
 javax.ejb.EJBLocalHome, 47, 774
 javax.ejb.EJBLocalObject, 47, 775
 javax.ejb.EJBMetaData, 775–776
 javax.ejb.EJBObject, 39–41, 776–777
 javax.ejb.EnterpriseBean, 36, 777
 javax.ejb.EntityBean, 36, 141–143, 777–778
 javax.ejb.EntityContext, 137–138, 778–784
 javax.ejb.Handle, 784
 javax.ejb.HomeHandle, 785
 javax.ejb.MessageDrivenBean, 36, 231, 785
 javax.ejb.MessageDrivenContext, 786
 javax.ejb.SessionBean, 36, 786–787
 javax.ejb.SessionContext, 92, 787–789
 javax.ejb.SessionSynchronization, 789–790
 javax.ejb.TimedObject, 397, 790
 javax.ejb.Timer, 396–397, 791–792
 javax.ejb.TimerHandle, 397, 792
 javax.ejb.TimerService, 395–396, 793–794
 javax.jms.MessageListener, 231
 javax.transaction.UserTransaction, 325–327
 local home interface
 bank account example, 155–156
 defined, 51

- interfaces, local home interface (*continued*)
 - Hello World example, 59–60
 - writing, 46–47
 - local interface
 - bank account example, 152–155
 - defined, 51
 - Hello World example, 56–57
 - local objects, 45
 - performance optimization, 552–553
 - relationships, 433
 - writing, 46–47
 - marker interface, 36, 662
 - MessageConsumer*, 225
 - MessageProducer*, 225
 - remote interface
 - bank account example, 151–152
 - CleanDayLimitOrdersEJB* timer example, 400
 - counter bean, 90
 - defined, 51
 - Hello World example, 55–56
 - javax.ejb.EJBObject* interface, 39–40
 - performance optimization, 552–553
 - RMI-IIOP, 657–658
 - rules, 39–40
 - Service Endpoint Interface (SEI), 111–112
 - service interfaces, 104
 - Session*, 225
 - Internet Inter-ORB Protocol (IIOP), 683–684, 697
 - interoperability
 - CORBA-EJB, 700–703
 - CORBA-RMI, 692–694, 698–699
 - secure interoperability, 378–381
 - Web services, 105
 - An Introduction to Database Systems, 7th Edition, C.J. Date*, 458
 - invocation models
 - dynamic proxy invocation, 287–288
 - reflective invocation, 287–288
 - static invocation, 287
 - invocations per second (IPS), 573
 - iPlanet Directory Server, 69
 - isolation levels
 - READ COMMITTED*, 335, 797
 - READ UNCOMMITTED*, 334–335, 797
 - REPEATABLE READ*, 336, 797
 - SERIALIZABLE*, 337, 797
 - isolation of transactions, 331–333, 338
- J**
- JAAS (Java Authentication and Authorization Service), 26, 357–368
 - Jad decompiler, 291
 - Java 2 Platform, Enterprise Edition (J2EE), 21–23
 - Java 2 Platform, Micro Edition (J2ME), 21
 - Java 2 Platform, Standard Edition (J2SE), 21–22
 - Java API for XML Parsing (JAXP), 25
 - Java API for XML RPC (JAX-RPC), 23, 112
 - Java Authentication and Authorization Service (JAAS), 26, 357–368
 - Java Community Process (JCP), 21, 272
 - Java Data Objects (JDO) specification, 122
 - Java Database Connectivity (JDBC), 13, 24, 142
 - Java IDL, 25
 - Java language
 - cross-platform functionality, 13
 - interface/implementation separation, 12
 - security, 12
 - Java Management Extension (JMX), 616
 - Java Message Service (JMS)
 - application integration, 479, 542
 - client-side callbacks, 282–283
 - connection, 222
 - Connection* interface, 225
 - ConnectionFactory* instance, 222
 - ConnectionFactory* interface, 225
 - consumer, 223
 - defined, 24, 220
 - destination, 222
 - Destination* interface, 225
 - JMS-EJB integration, 226–227
 - MessageConsumer* interface, 225
 - message-driven beans, 228–229, 231
 - MessageProducer* interface, 225
 - messaging domains, 220–222
 - non-durable subscriptions, 231
 - producer, 223
 - sending/receiving messages, 223–225
 - Service Provider Interface (SPI), 220
 - session, 222
 - Session* interface, 225
 - Java Naming and Directory Interface (JNDI)
 - architecture, 671–672
 - atomic name, 672
 - benefits, 670–671
 - bindings, 672–673
 - client API, 671
 - code example, 677–678
 - composite name, 674
 - composite namespaces, 674–676
 - compound name, 672
 - context operations, 678
 - contexts, 673
 - defined, 24
 - home objects, 69–72
 - initial context factories, 674–675
 - initial contexts, 674–677
 - initialization parameters, 256–257
 - integrating with RMI-IIOP, 679–681
 - mixing implementations, 654
 - namespaces, 674
 - naming systems, 674
 - provider URL, 677
 - Service Provider Interface (SPI), 671
 - specification, 653
 - subcontexts, 673
 - supplying environment information, 72–73
 - tutorial, 653
 - Java Native Interfaces (JNI), 480
 - Java platforms
 - Java 2 Platform, Enterprise Edition (J2EE), 21–23
 - Java 2 Platform, Micro Edition (J2ME), 21
 - Java 2 Platform, Standard Edition (J2SE), 21–22
 - Java Programming With CORBA*, Brose, Vogel, and Duddy, 688
 - Java RMI-IIOP-based clients, 68
 - Java Runtime Environment (JRE), 610
 - Java Server Pages (JSP), 24–25, 639
 - Java servlets, 24
 - Java Timer APIs, 393
 - Java Transaction API (JTA), 24, 325–327
 - Java Transaction Service (JTS), 24, 325
 - Java Virtual Machine (JVM)
 - crashes, 660
 - tuning, 563–565
 - JavaBeans, 16
 - java.io.Serializable* interface, 84–85
 - JavaMail service, 25
 - java.rmi.Remote* interface, 41
 - java.rmi.RemoteException* interface, 41
 - Java-to-IDL Mapping, 696–697
 - javax.coordination.ServiceManager* interface, 346
 - javax.ejb.EJBContext* interface, 63, 772–773
 - javax.ejb.EJBHome* interface, 44, 773–774
 - javax.ejb.EJBLocalHome* interface, 47, 774
 - javax.ejb.EJBLocalObject* interface, 47, 775
 - javax.ejb.EJBMetaData* interface, 775–776
 - javax.ejb.EJBObject* interface, 39–41, 776–777
 - javax.ejb.EnterpriseBean* interface, 36, 777
 - javax.ejb.EntityBean* interface, 36, 141–143, 777–778
 - javax.ejb.EntityContext* interface, 137–138, 778–784
 - javax.ejb.Handle* interface, 784
 - javax.ejb.HomeHandle* interface, 785
 - javax.ejb.MessageDrivenBean* interface, 36, 231, 785
 - javax.ejb.MessageDrivenContext* interface, 786
 - javax.ejb.SessionBean* interface, 36, 777
 - javax.ejb.Session-Bean* interface, 786–787
 - javax.ejb.SessionContext* interface, 92, 787–789
 - javax.ejb.SessionSynchronization* interface, 789–790
 - javax.ejb.TimedObject* interface, 397, 790
 - javax.ejb.Timer* interface, 396–397, 791–792
 - javax.ejb.TimerHandle* interface, 397, 792
 - javax.ejb.TimerService* interface, 395–396, 793–794
 - javax.jms.MessageListener* interface, 231

- javax.resource* Package (J2EE Connector API), 486
- javax.resource.cci* Package (J2EE Connector API), 486–489
- javax.resource.spi* Package (J2EE Connector API), 490–492
- javax.resource.spi.endpoint* Package (J2EE Connector API), 492
- javax.resource.spi.security* Package (J2EE Connector API), 493
- javax.resource.spi.work* Package (J2EE Connector API), 493–494
- javax.transaction.UserTransaction* interface, 325–327
- JAXP (Java API for XML Parsing), 25
- JAX-RPC (Java API for XML RPC), 23, 112
- JBoss open source application server, 18
- JBUILDER (Borland), 19
- JCA (J2EE Connector Architecture). *See* J2EE Connector Architecture
- JCP (Java Community Process), 21, 272
- JDBC connection pool tuning, 565–566
- JDBC driver support (EJB servers), 611
- JDBC (Java Database Connectivity), 13, 24, 142
- JDeveloper (Oracle), 19
- JDO (Java Data Objects) specification, 122
- JMS (Java Message Service)
 - application integration, 479, 542
 - client-side callbacks, 282–283
 - connection, 222
 - ConnectionFactory* interface, 225
 - ConnectionFactory* instance, 222
 - ConnectionFactory* interface, 225
 - consumer, 223
 - defined, 24, 220
 - destination, 222
 - Destination* interface, 225
 - durable subscriptions, 231
 - JMS-EJB integration, 226–227
 - MessageConsumer* interface, 225
 - message-driven beans, 228–229, 231
 - MessageProducer* interface, 225
 - messaging domains, 220–222
 - non-durable subscriptions, 231
 - producer, 223
 - sending/receiving messages, 223–225
 - Service Provider Interface (SPI), 220
 - session, 222
 - Session* interface, 225
- JMS message-driven beans. *See* message-driven beans
- JMX (Java Management Extension), 616
- JNDI (Java Naming and Directory Interface)
 - architecture, 671–672
 - atomic name, 672
 - benefits, 670–671
 - bindings, 672–673
 - client API, 671
 - code example, 677–678
 - composite name, 674
 - composite namespaces, 674–676
 - compound name, 672
 - context operations, 678
 - contexts, 673
 - defined, 24
 - home objects, 69–72
 - initial context factories, 674
 - initial contexts, 674–677
 - initialization parameters, 256–257
 - integrating with RMI-IIOP, 679–681
 - mixing implementations, 654
 - namespaces, 674
 - naming systems, 674
 - provider URL, 677
 - Service Provider Interface (SPI), 671
 - specification, 653
 - subcontexts, 673
 - supplying environment information, 72–73
 - tutorial, 653
- JNI (Java Native Interfaces), 480
- JProf performance-profiling tool, 565
- JRE (Java Runtime Environment), 610
- JSP (Java Server Pages), 24–25, 639
- JTA (Java Transaction API), 24, 325–327
- JTS (Java Transaction Service), 24, 325
- J2EE Activity Service, 346–347
- “J2EE and .NET” article, Rima Patel Sriganesh, 598
- J2EE application servers, 18
- J2EE Blueprints best practices guide, 600
- J2EE Connector API
 - javax.resource* Package, 486
 - javax.resource.cci* Package, 486–489
 - javax.resource.spi* Package, 490–492
 - javax.resource.spi.endpoint* Package, 492
 - javax.resource.spi.security* Package, 493
 - javax.resource.spi.work* Package, 493–494
- J2EE Connector Architecture
 - defined, 479–480
 - message-driven beans, 228
 - non-managed environments, 483
- OutboundLoanRA example
 - architecture, 508–509
 - client contracts, 511
 - ConnectionFactoryImpl.java*, 512–514
 - ConnectionImpl.java*, 514–516
 - ConnectionMetaDataImpl.java*, 516–517
 - ConnectionRequestInfoImpl.java*, 532–533
 - ConnectionSpecImpl.java*, 517
 - deployment, 533–534
 - deployment descriptor, 534–535
 - extending, 541
 - InteractionImpl.java*, 517–519
 - JavaLoanApp.java*, 509–510
 - LoanApp.dll*, 510–511
 - LoanRatesClient* standalone Java application, 538–541
 - LoanRatesEJB* stateless session bean, 535–538
 - ManagedConnectionFactoryImpl.java*, 525–528
 - ManagedConnectionImpl.java*, 528–532
 - ManagedConnectionMetaDataImpl.java*, 533
 - MappedRecordImpl.java*, 520–522
 - RecordFactoryImpl.java*, 522
 - ResourceAdapterMetaDataImpl.java*, 523–525
 - system contracts, 525
- system contracts
 - connection management, 484, 495–498
 - life cycle management, 484, 494–495
 - message in-flow, 486, 506–508
 - security management, 484, 498–500
 - transaction inflow, 485
 - transaction management, 484, 501–504
 - work management, 485, 504–506
- transactions, 319
- uses, 14, 25
- when to use, 542
- J2EE Deployment API (JSR-88), 289
- J2EE (Java 2 Platform, Enterprise Edition), 21–23
- J2EE object model, 600
- “J2EE vs. Microsoft.NET” whitepaper, Roger Sessions, 598
- J2ME (Java 2 Platform, Micro Edition), 21
- J2SE (Java 2 Platform, Standard Edition), 21–22
- JVM (Java Virtual Machine)
 - crashes, 660
 - tuning, 563–565
- L**
- large result sets, 474–475
- lazy loading, 433–435, 558, 611
- LDAP (Lightweight Directory Access Protocol), 670
- legacy databases
 - common design problems, 463–471
 - integration tools, 472
 - migration strategies, 472–474
- life cycle
 - BMP entity beans, 177–180
 - CMP entity beans, 214–215
 - defined, 6
 - entity beans, 123, 125
 - message-driven beans, 233–234
 - role of containers, 42
 - session beans, 79–80, 98–101
- life cycle management contract, 484, 494–495
- Lightweight Directory Access Protocol (LDAP), 670
- Lister, Timothy, *Peopleware: Productive Project and Teams, 2nd Edition*, 605

- load balancing
 - clustering, 572
 - DNS, 590
 - EJB servers, 615
 - large business systems, 5
 - message-driven beans, 242
- local home interface
 - bank account example, 155–156
 - defined, 51
 - Hello World example, 59–60
 - writing, 46–47
- local interface
 - bank account example, 152–155
 - defined, 51
 - Hello World example, 56–57
 - local object, 45
 - performance optimization, 552–553
 - relationships, 433
 - writing, 46–47
- local object (defined), 51
- local/remote transparency, 660
- location transparency, 32, 38, 43
- locking transactions, 333
- logging
 - debug logs, 73
 - large business systems, 6
 - tuning, 565
- lost update problem, 332
- Lotus Notes Domino Server (IBM), 69
- M**
- M x N integration problem, 480–481
- Mandatory transaction attribute, 321–322, 324
- many-to-many (M:N) relationships, 411, 421–428
- mapping objects to relational data, 120–122
- Marinescu, Floyd, *EJB Design Patterns*, 269, 545
- marker interfaces, 36, 662
- MAX aggregate function (EJB-QL), 752
- MDD (Middle Driven Development), 275–276
- message in-flow contract, 486, 506–508
- MessageConsumer* interface, 225
- message-driven beans
 - application integration, 479, 542
 - bean implementation class, 234–236
 - characteristics, 229–230
 - client interaction, 30
 - client program, 241
 - clustering, 243–244, 588–590
 - component interfaces, 228
 - defined, 29, 217, 227–228
 - deployment descriptor, 236–241, 716–718
 - diagrams, 770–771
 - ejbCreate()* method, 232, 786
 - ejbRemove()* method, 232, 245, 786
 - enterprise bean class, 36
 - Java Message Service (JMS), 228–229, 231
 - javax.ejb.MessageDrivenBean* interface, 36, 231
 - javax.jms.MessageListener* interface, 231
 - J2EE Connector Architecture, 228
 - life cycle, 233–234
 - load-balancing, 242
 - message ordering, 245
 - onMessage(Message)* method, 232, 786
 - poison messages, 246–249
 - pull model, 242
 - push model, 242
 - queues, 246
 - request()* method, 253
 - request/response paradigm, 249–253
 - security, 242
 - setMessageDrivenContext(MessageDrivenContext)* method, 233, 785
 - timers, 236
 - transactions, 241–242, 316
 - tuning, 563
- message-driven contexts, 786
- MessageDrivenBean* object, 785
- MessageDrivenContext* object, 786
- message-oriented middleware (MOM), 6, 219–220
- MessageProducer* interface, 225

- messaging
 - certified message delivery, 220
 - consumers, 218
 - defined, 217–218
 - domains, 220–222
 - EJB servers, 618
 - guaranteed message delivery, 220
 - Java Message Service (JMS), 220
 - producers, 218
 - remote method invocations, 218
 - RMI-IIOP, 219, 294–297
 - store and forward, 220
- method invocation models
 - dynamic proxy invocation, 287–288
 - reflective invocation, 287–288
 - static invocation, 287
- methods
 - begin()*, 326, 799
 - bPassivate()*, 147
 - commit()*, 326, 799
 - create()*, 60
 - createTimer()*, 395, 794
 - ejbActivate()*
 - BMP entity beans, 146, 782
 - CMP entity beans, 194, 782
 - entity beans (general), 129
 - session beans, 86–89, 788
 - ejbCreate()*
 - BMP entity beans, 145, 780
 - CMP entity beans, 193, 780
 - entity beans, 132–134, 780
 - message-driven beans, 786
 - session beans, 89, 788
 - ejbFind()*, 144, 192, 779
 - ejbHome()*, 145, 193, 780
 - ejbLoad()*
 - BMP entity beans, 146, 782
 - CMP entity beans, 194, 782
 - entity beans, 126–127, 139
 - ejbPassivate()*
 - BMP entity beans, 781
 - CMP entity beans, 195, 781
 - entity beans, 86–87, 89, 788
 - session beans, 130–131, 788
 - ejbPostCreate()*, 146, 194, 781
 - ejbRemove()*
 - BMP entity beans, 147, 783
 - CMP entity beans, 195, 783
 - entity beans, 89, 101, 133–135, 139
 - message-driven beans, 786
 - session beans, 101, 788
 - ejbSelect()*, 189–190, 192, 779
 - ejbStore()*
 - BMP entity beans, 147, 781
 - CMP entity beans, 195, 781
 - entity beans, 126–127
 - ejbTimeout()*, 397, 399
 - getEJBLocalObject()*, 138, 784, 789
 - getEJBObject()*, 138, 784, 789
 - getHandle()*, 264
 - getInfo()*, 396
 - getMessageContext()*, 789
 - getPrimaryKey()*, 139, 784
 - getStatus()*, 326–327, 799
 - getTimers()*, 395, 794
 - getTimerService()*, 398
 - onMessage()*, 232, 786
 - remove()*, 60
 - request()*, 253
 - rollback()*, 326, 799
 - setEntityContext()*
 - BMP entity beans, 144, 779
 - CMP entity beans, 191, 779
 - setMessageDrivenContext(MessageDrivenContext)*, 233, 785
 - setRollbackOnly()*, 326, 799
 - setSessionContext()*, 62, 89, 92, 788
 - setTransactionTimeout(int)*, 326, 799
 - unsetEntityContext()*, 147, 195, 783

- Microsoft
 - Active Directory, 69
 - DCOM distributed objects, 31
 - .NET, 13
- Middle Driven Development (MDD), 275–276
- middleware
 - application servers, 5
 - building versus buying, 5
 - CORBA, 684
 - defined, 5–6
 - deployment descriptors, 48–49
 - distributed objects, 32–35
 - explicit, 32–33
 - implicit, 33–35
 - message-oriented middleware (MOM), 6, 219–220
- Middleware Company, 599
- Middleware Dark Matter* article, Steve Vinoski, 109
- migration of legacy data, 449, 472–474
- MIN aggregate function (EJB-QL), 752
- M:N (many-to-many) relationships, 411, 421–428
- mock object code generation, 282
- mock object unit testing, 281–282
- modifying entity bean data, 136–137
- MOM (message-oriented middleware), 6, 219–220
- monitoring EJB containers, 38
- multithreaded beans, 226
- N**
- naming and directory services
 - Active Directory (Microsoft), 69
 - CORBA Naming Service (COS Naming), 691, 699–700
 - defined, 68–69, 667–669
 - Directory Server (iPlanet), 69
 - Java Naming and Directory Interface (JNDI), 670–677
 - Lightweight Directory Access Protocol (LDAP), 670
 - Lotus Notes Domino Server (IBM), 69
 - Network Directory System (NDS), 670
 - Network Information System (NIS), 670
- naming servers
 - centralized, 589–590
 - clustering, 589–590
 - shared, replicated, 589
- natural keys, 460–461
- nested transactions, 308–310
- .NET (Microsoft), 13
- NetBeans IDE, 19
- Network Directory System (NDS), 670
- Network Information System (NIS), 670
- network or machine failures during transactions, 301–302
- Never* transaction attribute, 321–322, 324
- nonpersistent objects, 80
- normalization, 457–459
- `NoSuchEntityException` exception, 795
- `NoSuchObjectLocalException` exception, 795
- NotSupported* transaction attribute, 321–322, 324
- Novell Network Directory System (NDS), 670
- O**
- object factory, 43
- object handles, 263–264
- object life cycle, 6
- Object Request Broker (ORB), 686–687
- object serialization, 84, 662–667
- Object Transaction Service (OTS), 324–325
- `ObjectNotFoundException` exception, 795
- Object-Oriented Programming (OOP), 286–287
- object-relational mapping
 - data model, 459
 - defined, 120–121
 - denormalization, 457–459
 - design process, 459–460
 - EJB object model, 459
 - hard-coded versus soft-coded SQL, 454–455
 - impedance mismatch, 454
 - normalization, 457–459
 - stored procedures, 455–457
 - tools, 122
- object. *See* EJB object
- OMG CORBA. *See* CORBA (Common Object Request Broker Architecture)
- OMG interface definition language (OMG IDL), 687–689
- OMG Web site, 688
- one-to-many (1:N) relationships, 411, 416–420
- one-to-one (1:1) relationships, 411–416
- `onMessage(Message)` method, 232, 786
- OOP (Object-Oriented Programming), 286–287
- open source EJB servers, 620
- optimistic concurrency control (transactions), 339
- Optimizelt performance-profiling tool, 565
- optimizing performance. *See* performance optimization
- O/R mapping frameworks, 272
- Oracle Application Server, 18
- Oracle JDeveloper, 19
- Oracle TopLink, 122
- ORB (Object Request Broker), 686–687
- Organizational Patterns for Teams*, Neil B. Harrison, 606
- OTS (Object Transaction Service), 324–325
- OutboundLoanRA example
 - architecture, 508–509
 - client contracts, 511
 - `ConnectionFactoryImpl.java`, 512–514
 - `ConnectionImpl.java`, 514–516
 - `ConnectionMetaDataImpl.java`, 516–517
 - `ConnectionRequestInfoImpl.java`, 532–533
 - `ConnectionSpecImpl.java`, 517
 - deployment, 533–534
 - deployment descriptor, 534–535
 - extending, 541
 - `InteractionImpl.java`, 517–519
 - `JavaLoanApp.java`, 509–510
 - `LoanApp.dll`, 510–511
 - `LoanRatesClient` standalone Java application, 538–541
 - `LoanRatesEJB` stateless session bean, 535–538
 - `ManagedConnectionFactoryImpl.java`, 525–528
 - `ManagedConnectionImpl.java`, 528–532
 - `ManagedConnectionMetaDataImpl.java`, 533
 - `MappedRecordImpl.java`, 520–522
 - `RecordFactoryImpl.java`, 522
 - `ResourceAdapterMetaDataImpl.java`, 523–525
 - system contracts, 525
- P**
- packaging Web services, 113–114
- partitioning
 - clusters, 573–577
 - resources, 553–554
- pass-by-reference, 655–656, 662
- pass-by-value, 655–656, 661–662
- passivation
 - entity beans, 130–131
 - stateful session beans, 83–84, 86–87
 - stateless session beans, 84
- Peopleware: Productive Project and Teams, 2nd Edition*, Tom Demarco and Timothy Lister, 605
- performance
 - BMP entity beans, 450
 - CMP entity beans, 450
- performance optimization
 - capacity planning, 549–550
 - CLASSPATH directory, 566
 - defining performance requirements, 545–546
 - EJB Query Language (EJB-QL), 754–755
 - EJB server, 567
 - local interface, 552–553
 - partitioning resources, 553–554
 - remote interface, 552–553
 - RMI-IIOP, 218, 566
 - session façade, 550–552
 - statelessness, 547–549
 - tuning
 - entity beans, 556–562
 - Java Virtual Machine (JVM), 563–565
 - JDBC connection pool, 565–566
 - logging, 565
 - message-driven beans, 563
 - performance-profiling tools, 565
 - session beans, 554–556
 - Web applications, 566
 - tuning stateless session beans, 555
- performance-profiling tools, 565
- persistence
 - bean-managed, 131
 - caching, 448
 - container-managed, 132

- persistence (*continued*)
 - defined, 37, 80
 - migration, 449
 - object-relational mapping, 120–122, 454–460
 - pluggable persistence providers, 611–612
 - rapid application development, 449
 - schema independence, 448–449
 - session beans, 446–449
- persistent data components, 122–123
- pessimistic concurrency control (transactions), 339
- phantom problem, 336–337
- Plain Old Java Object (POJO), 272
- platform independence of Web services, 109–110
- platform selection, 594–598
- platforms
 - Java 2 Platform, Enterprise Edition (J2EE), 21–23
 - Java 2 Platform, Micro Edition (J2ME), 21
 - Java 2 Platform, Standard Edition (J2SE), 21–22
- pluggable persistence providers, 611–612
- POA (Portable Object Adapter), 690
- point-to-point (PTP) messaging domain, 221
- poison messages, 246–249
- POJO (Plain Old Java Object), 272
- pooling
 - connection pooling, 260
 - entity beans, 128–130
 - stateful session beans, 83–84
- port component, 110
- portability
 - Enterprise JavaBeans (EJB), 11
 - Web services, 105
- Portable Object Adapter (POA), 690
- presentation tier, 630, 637
- primary key class, 125
- primary key generation, 659–660
- primary keys, 460–461
- Principles of Databases Systems*, Jeffrey D. Ullman, 333
- producers (Java Message Service), 223
- product line example
 - class diagram, 196
 - client code, 212–214
 - container-specific deployment descriptor, 210–211
 - deployment descriptor, 207–210
 - ProductBean.java, 203–207
 - ProductHome.java, 198–200
 - Product.java, 197
 - ProductLocalHome.java, 200–201
 - ProductLocal.java, 198
 - ProductPK.java, 201–202
 - running the client program, 214
- programmatically authorization, 352, 368–373, 376–377
- programmatic transactions
 - benefits, 314
 - CORBA Object Transaction Service (OTS), 324
 - declarative versus programmatic, 328
 - defined, 311
 - spaghetti code, 330
- project team, 603–606
- proprietary descriptor, 94
- PTP (point-to-point) messaging domain, 221
- publish/subscribe messaging domain, 220–221
- pull model in message-driven beans, 242
- push model in message-driven beans, 242
- Q**
- Quartz scheduler, 394
- queues (message-driven beans), 246
- R**
- rapid application development (RAD), 12, 449–450
- RAS properties, 570–572
- READ COMMITTED isolation level, 335, 797
- read locks, 333
- READ UNCOMMITTED isolation level, 334–335, 797
- real-time deployment, 618
- recursive relationships, 437–438
- redundancy, 571
- refactoring, 278–279
- Refactoring: Improving the Design of Existing Code*, Martin Fowler, 278
- references. *See* EJB references
- referential integrity, 439–444
- reflective invocation, 287–288
- relationships
 - aggregation, 434–436
 - BMP entity beans, 410, 452
 - cardinality, 411
 - circular, 438–439
 - CMP entity beans, 410, 452
 - composition, 434–436
 - deployment descriptor, 729–732
 - directionality, 428–433
 - EJB Query Language (EJB-QL), 436–437
 - lazy loading, 433–435
 - local interfaces, 433
 - many-to-many (M:N), 411, 421–428
 - one-to-many (1:N), 411, 416–420
 - one-to-one (1:1), 411–416
 - recursive, 437–438
 - referential integrity, 439–444
 - session beans, 412
- reliability
 - large business systems, 570–571
 - RMI-IIOP, 218
- remote accessibility, 37–38
- remote exceptions, 41, 58
- remote interface
 - bank account example, 151–152
 - CleanDayLimitOrdersEJB* timer example, 400
 - counter bean, 90
 - defined, 51
 - Hello World example, 55–56
 - javax.ejb.EJBObject* interface, 39–40
 - performance optimization, 552–553
 - RMI-IIOP, 657–658
 - rules, 39–40
- Remote Method Invocation over the Internet Inter-ORB Protocol (RMI-IIOP). *See* RMI-IIOP (Remote Method Invocation over the Internet Inter-ORB Protocol)
- Remote Method Invocation (RMI)
 - client-side callbacks, 283
 - defined, 23
 - differences from CORBA, 695–696
 - features, 654
 - Internet Inter-ORB Protocol (IIOP), 697
 - interoperability with CORBA, 692–694, 698–699
 - Java-to-IDL Mapping, 696–697
- remote method invocations
 - defined, 654
 - large business systems, 5
 - marshalling/unmarshalling, 655
 - messaging, 218
 - network or machine instability, 655
 - parameter passing conventions, 655
- remote procedure call (RPC), 654
- remove()* method, 60
- RemoveException* exception, 795
- REPEATABLE READ isolation level, 336, 797
- request()* method, 253
- request/reply messaging domain, 222
- requests per second (RPS), 573
- Required* transaction attribute, 319–320, 322, 324
- RequiresNew* transaction attribute, 320, 322, 324
- resource factories
 - calling beans from other beans, 259–262
 - deployment descriptor, 260–261, 727–728
- resource managers, 304
- resource pooling, 6
- resources
 - bean-independent, 553–554
 - bean-specific, 553–554
 - partitioning, 553–554
- reusable services, 9–11
- reusing code, 292–293, 601
- risk assessment (security), 351–352
- RMI (Remote Method Invocation)
 - client-side callbacks, 283
 - defined, 23
 - differences from CORBA, 695–696
 - features, 654
 - Internet Inter-ORB Protocol (IIOP), 697
 - interoperability with CORBA, 692–694, 698–699
 - Java-to-IDL Mapping, 696–697

- rmic (RMI compiler) tool, 661
- RMI-IIOP (Remote Method Invocation over the Internet Inter-ORB Protocol)
 - clients, 68
 - defined, 23–24, 654
 - distributed objects, 31
 - integrating with JNDI, 679–681
 - integration with other MOM systems, 218
 - java.rmi.Remote interface, 41
 - JVM crashes, 660
 - local/remote transparency, 660
 - messaging, 219, 294–297
 - mixing implementations, 654
 - object serialization, 662–667
 - parameter-passing conventions, 41
 - pass-by-reference, 655–656, 662
 - pass-by-value, 655–656, 661–662
 - performance optimization, 218, 566
 - primary key generation, 659–660
 - reliability, 218
 - remote interface, 657–658
 - remote method invocations, 654–655
 - remote object implementation, 658–659
 - RMI-IIOP/CORBA combinations, 694–695
 - skeletons, 660–661
 - specification, 653
 - stubs, 660–661
 - support, 218
 - threading, 660
 - tutorial, 653
- rollback()* method, 326, 799
- Ronin International, 599
- RPC (remote procedure call), 654
- RPC-style SOAP, 8
- RPS (requests per second), 573
- S**
- sagas, 310
- Salesforce.com enterprise software, 10
- SAML (Security Assertion Markup Language), 386–387, 389
- scalability of EJB servers, 612–613
- scheduling
 - Batch utility (Unix), 392
 - code execution, 391–392
 - Cron jobs, 392
 - EJB support, 394
 - EJB timer service, 394–398, 406–407
 - Java Timer APIs, 393
 - Quartz scheduler, 394
 - Sims Computing Flux Scheduler, 394
 - workflows, 391–392
- schema independence, 448–449
- secure interoperability, 378–381
- security
 - authentication, 352, 354–355
 - authorization, 352, 355, 368–377
 - Common Secure Interoperability version 2 (CSlv2), 379–381
 - controls, 351–353
 - cryptography, 383–386
 - data confidentiality protection, 353, 356
 - data integrity protection, 352, 356
 - deployment descriptor, 725–726
 - EJB servers, 613–614
 - Enterprise JavaBeans (EJB), 37, 353
 - firewalls, 577
 - Java Authentication and Authorization Service (JAAS), 357–368
 - Java language, 12
 - large business systems, 6
 - message-driven beans, 242
 - risk assessment, 351–352
 - Security Assertion Markup Language (SAML), 386–387, 389
 - security roles, 368
 - social engineering, 350
 - SSL/TLS, 379
 - violations (security breaches), 351
 - vulnerabilities, 351
 - Web applications, 353–356
 - Web Services, 381–383
 - WS-Security, 387–389
 - XML Digital Signature, 383–386, 389
 - XML Encryption, 383–386
 - Security Assertion Markup Language (SAML), 386–387, 389
 - security contexts, 377–378
 - Security Engineering*, Ross Anderson, 350
 - security management contract, 484, 498–500
 - SEI (Service Endpoint Interface), 111–112
 - serializability of transactions, 333
 - SERIALIZABLE isolation level, 337, 797
 - serialization, 84, 662–667
 - server provider, 18
 - server-side output, 73
 - TheServerSide.com Web site, 598, 600, 606
 - Service Endpoint Interface (SEI), 111–112
 - service interfaces, 104
 - Service Provider Interface (SPI), 220
 - serviceability, 571
 - Service-Oriented Architecture (SOA), 8–9, 103–105
 - services
 - CORBA, 691
 - defined, 8
 - directory services, 68–69
 - JavaMail service, 25
 - naming services, 68–69
 - reusable services, 9–11
 - Web Services, 8–9, 103
 - servlets, 24, 284, 637–639
 - session beans
 - client interaction, 30
 - clustering, 581–584
 - conversational state, 84–85
 - count bean example, 88, 90–98
 - CRUD operations, 551
 - defined, 28, 79
 - deployment descriptor, 708–710
 - diagrams, 758–764
 - ejbActivate()* method, 86–89, 788
 - ejbCreate()* method, 89, 788
 - ejbPassivate()* method, 86–87, 89, 788
 - ejbRemove()* method, 89, 101, 788
 - enterprise bean class, 36
 - entity beans, 29, 123–124
 - failures, 549
 - getEJBLocalObject()* method, 789
 - getEJBObject()* method, 789
 - getMessageContext()* method, 789
 - javax.ejb.SessionBean interface, 36
 - life cycle, 98–101
 - lifetime, 79–80
 - LoanRatesEJB* example, 535–538
 - persistence, 446–449
 - relationships, 412
 - setSessionContext()* method, 89, 92, 788
 - stateful (defined), 80–81, 83–84
 - stateless (defined), 81–82, 84
 - tuning, 554–556
 - Web Service endpoints, 284
 - session contexts, 92, 787
 - session façade, 550–552
 - Session interface, 225
 - SessionBean object, 786–787
 - SessionContext object, 787–789
 - sessions (Java Message Service), 222
 - Sessions, Roger, “J2EE vs. Microsoft.NET”
 - whitepaper, 598
 - SessionSynchronization object, 789–790
 - setEntityContext()* method
 - BMP entity beans, 144, 779
 - CMP entity beans, 191, 779
 - setMessageDrivenContext (MessageDrivenContext)*
 - method, 233, 785
 - setRollbackOnly()* method, 326, 799
 - setSessionContext()* method, 62, 89, 92, 788
 - setTransactionTimeout(int)* method, 326, 799
 - shared, replicated naming servers, 589–590
 - shutdown (clean), 6, 617
 - Siebel enterprise software, 10
 - Simple Object Access Protocol (SOAP). *See* SOAP protocol
 - Sims Computing Flux Scheduler, 394
 - single access point simplicity, 572

- single-threaded beans, 226
 - singletons, 293
 - skeletons (RMI-IIOP), 660–661
 - SOA (Service-Oriented Architecture), 8–9, 103–105
 - SOAP protocol
 - defined, 8
 - document-style, 8
 - RPC-style, 8
 - Web services, 108–109
 - social engineering, 350
 - soft-coded SQL, 454–455
 - software proxies, 590
 - specifications
 - Enterprise JavaBeans (EJB), 12
 - Implementing Enterprise Web Services* JSR 921, 110
 - J2EE Activity Service and the Extended Transactions (JSR 095), 346
 - Java Data Objects (JDO), 122
 - RMI-IIOP, 653
 - Sun Java Data Objects (JDO) specification, 122
 - SPECjAppServer benchmarks, 567
 - SPI (Service Provider Interface), 220
 - Spring open source framework, 13, 272
 - Sriganesh, Rima Patel, “J2EE and .NET” article, 598
 - SSL/TLS, 379
 - staffing, 598–599
 - standard build process, 607
 - stateful session beans
 - activation, 83–84, 86–88
 - clustering, 583–584
 - count bean example, 88, 90–98
 - defined, 80–81
 - diagrams, 762–764
 - ejbActivate()* method, 89
 - ejbCreate()* method, 89
 - ejbPassivate()* method, 89
 - ejbRemove()* method, 89
 - life cycle, 100–101
 - passivation, 83–84, 86–87
 - pooling, 83–84
 - relationships, 412
 - setSessionContext()* method, 89, 92
 - tuning, 555–556
 - stateless session beans
 - activation, 84
 - clustering, 581–582
 - defined, 81–82
 - diagrams, 759–761
 - ejbActivate()* method, 89
 - ejbCreate()* method, 89
 - ejbPassivate()* method, 89
 - ejbRemove()* method, 89
 - life cycle, 98–100
 - LoanRatesEJB* example, 535–538
 - passivation, 84
 - relationships, 412
 - setSessionContext()* method, 89
 - tuning, 554–555
 - Web Service endpoints, 284
 - statelessness, 547–549
 - static invocation, 287
 - status of transactions, 327, 798
 - store and forward, 220
 - stored procedures, 455–457
 - StrutsEJB project, 273
 - stubs, 30–31, 660–661
 - SUM aggregate function (EJB-QL), 752
 - Sun Microsystems
 - Java Studio, 19
 - Java System Application Server, 18
 - ONE Application Server, 18
 - Web site, 653
 - Supports* transaction attribute, 320, 322, 324
 - surrogate keys, 460–461
 - system administrator, 17–18
 - system contracts
 - connection management, 484, 495–498
 - lifecycle management, 484, 494–495
 - message in-flow, 486, 506–508
 - security management, 484, 498–500
 - transaction inflow, 485
 - transaction management, 484, 501–504
 - work management, 485, 504–506
 - system-level exceptions, 58–59
 - systems management, 6
- ## T
- team organization strategies, 603–606
 - testing code, 279–282
 - TheServerSide.com Web site, 598, 600, 606
 - thick clients, 14
 - thread support, 38
 - threading, 6, 660
 - thread-safe code, 226
 - throughput, 573
 - throwing exceptions, 58–59
 - TimedObject* object, 790
 - Timer* object, 791–792
 - timer service API, 395–398
 - TimerHandle* object, 792
 - timers
 - CleanDayLimitOrdersEJB* example, 399–406
 - deployment descriptor, 718
 - EJB timer service, 394–398, 406–407
 - Java Timer APIs, 393
 - message-driven beans, 236
 - transactions, 399
 - TimerService* object, 793–794
 - tools
 - Apache group’s Ant build tool, 607
 - conversion tools, 610
 - glue-code tools, 39
 - legacy data integration tools, 472
 - mock object code generation, 282
 - product reviews, 606
 - rmic (RMI compiler) tool, 661
 - TopLink (Oracle), 122
 - TPS (transactions per second), 573
 - training programs, 599, 621
 - transaction inflow contract, 485
 - transaction management contract, 484, 501–504
 - transactional communications protocol, 342–343
 - transactional contexts, 342
 - TransactionRequiredLocalException*
 - exception, 796
 - TransactionRolledBackLocalException*
 - exception, 796
 - transactions
 - aborting, 329
 - ACID properties, 304–306
 - atomic operations, 300–301
 - attributes, 317–324, 796–797
 - begin()* method, 326
 - benefits, 303–304
 - chained transactions, 310
 - client code, 330–331
 - client-initiated, 313–316
 - commit()* method, 326
 - concurrency control
 - dirty reads, 334–335
 - need for, 332–333
 - optimistic, 339
 - pessimistic, 339
 - phantom problem, 336–337
 - unrepeatable reads, 336–337
 - container-managed, 37, 310, 317–318, 330
 - deadlocks, 333
 - declarative, 312–314, 328
 - defined, 303–304
 - designing transactional conversations, 343–345
 - distributed, 340–342, 618
 - dooming, 329
 - durability, 340–341
 - entity beans, 315
 - failures, 301–302
 - flat transactions, 306–308
 - getStatus()* method, 326–327
 - isolation, 331–333

- isolation levels, 334–338, 797
 - J2EE Activity Service, 346–347
 - J2EE connectors, 319
 - Java Transaction API (JTA), 24, 325–327
 - Java Transaction Service (JTS), 24, 325
 - javax.coordination.ServiceManager* interface, 346
 - javax.transaction.UserTransaction* interface, 325–327
 - large business systems, 5
 - locking, 333
 - lost update, 332
 - message-driven beans, 241–242, 316
 - methods, 799
 - multiple users, 302–303
 - nested transactions, 308–310
 - network or machine failures, 301–302
 - Object Transaction Service (OTS), 324–325
 - programmatic, 311, 314, 324, 328, 330
 - read locks, 333
 - resource managers, 304
 - resources, 304
 - rollback()* method, 326
 - sagas, 310
 - serializability, 333
 - setRollbackOnly()* method, 326
 - setTransactionTimeout(int)* method, 326
 - status, 327, 798
 - timeouts, 799
 - timers, 399
 - transaction managers, 304
 - transactional objects, 304
 - two-phase commit protocol, 340–342
 - write locks, 333
 - transactions per second (TPS), 573
 - transparent fail-over
 - EJB servers, 58–59, 615
 - large business systems, 5
 - Trillium Control Center legacy data integration tool, 472
 - truth tables (EJB-QL), 753–754
 - tuning
 - entity beans, 556–562
 - Java Virtual Machine (JVM), 563–565
 - JDBC connection pool, 565–566
 - logging, 565
 - message-driven beans, 563
 - performance-profiling tools, 565
 - session beans, 554–556
 - Web applications, 566
 - tutorials for RMI-IIOP and JDNI, 653
 - two-phase (2PC) commit protocol of transactions, 340–342
- U**
- UDDI (Universal Description, Discovery, and Integration), 104
 - Ullman, Jeffrey D, *Principles of Databases Systems*, 333
 - UML editor, 615
 - unchecked exceptions, 59
 - The Unified Process Inception Phase*, Scott W. Ambler and Larry L. Constantine, 606
 - unit testing, 279–282
 - Universal Description, Discovery, and Integration (UDDI), 104
 - unrepeatable reads, 336–337
 - unsetEntityContext()* method, 147, 195, 783
- V**
- vendor-specific files, 49, 51, 65
 - versioning, 461–463
 - vertical slice, 601–602
 - Vinoski, Steve, *Middleware Dark Matter* article, 109
 - violations (security breaches), 351
 - Vogel, Brose, and Duddy, *Java Programming With CORBA*, 688
 - vulnerabilities in security, 351
- W**
- Wafer project, 274
 - Web application frameworks
 - choosing, 272–273
 - Hibernate, 122, 272
 - integration with EJB, 273
 - open source versus closed source, 274
 - small device support, 273
 - Spring, 272
 - standards support, 274
 - StrutsEJB project, 273
 - tools support, 273
 - Wafer project, 274
 - XDoclet, 289–290
 - Web applications
 - security, 353–356
 - tuning, 566
 - Web Service Description Language (WSDL), 8–9, 106–107
 - Web Services
 - application integration, 479, 543
 - clients, 14–15, 114–116
 - client-side callbacks, 283
 - defined, 8–9, 103
 - deploying, 113–114
 - deployment descriptor, 719–721
 - EJB servers, 619
 - implementing, 110–111
 - interoperability, 105
 - Java APIs for XML RPC (JAX-RPC), 112
 - packaging, 113–114
 - platform independence, 109–110
 - port component, 110
 - portability, 105
 - security, 381–383
 - Service Endpoint Interface (SEI), 111–112
 - service interfaces, 104
 - Service-Oriented Architecture (SOA), 103–105
 - servlets, 284
 - SOAP protocol, 108–109
 - stateless session beans, 284
 - Universal Description, Discovery, and Integration (UDDI), 104
 - Web Services Definition Language (WSDL) file, 104
 - XML artifacts, 109–110
 - Web sites
 - Aardvark Knowledge Builder, 472
 - Ascential Software DataStage, 472
 - ETI*Extract, 472
 - ExtremeProgramming.Org, 606
 - Informatica PowerCenter, 472
 - Middleware Company, 599
 - OMG, 688
 - Ronin International, 599
 - StrutsEJB project, 273
 - Sun Microsystems, 653
 - TheServerSide.com, 598, 600, 606
 - Trillium Control Center, 472
 - Wafer project, 274
 - work management contract, 485, 504–506
 - workflow engines (EJB servers), 619–620
 - write locks, 333
 - WSDL (Web Service Description Language), 8–9, 106–107
 - WS-Security, 387–389
- X**
- XDoclet framework, 289–290
 - XML, 293–294
 - XML artifacts, 109–110
 - XML Digital Signature, 383–386, 389
 - XML Encryption, 383–386
 - XP (Extreme Programming), 277–279