

Symbols

- # placeholders
 - using inline parameters with 89–91
- \$ placeholders
 - using inline parameters with 91–92
- @@identity value 111

A

- abbreviations 82
- abstraction layer 19, 258
- accessor methods 82
- account 304
- Account objects 248
- account.accountId property 134
- Account.getIdValue mapped statement 90
- Account.xml file 236
- AccountDao interface 230, 241, 264
- AccountDao variable 230
- AccountDaoImpl class 230
- accountId parameter 123
- accountId property 86, 116
- accountId value 90, 93
- AccountManufacturerProduct class 141
- Action 306
 - class 262
- ActionContext 306–308

- ActionForm 306–307
- ActionForward 307
- ActionMapping 308
- ActionServlet 311
- active transactions 154–155
- address fields 31
- advanced query techniques.
 - advanced
 - See query techniques, advanced
- advantages of iBatis
 - encapsulated Structured Query Language (SQL) 13
 - externalized Structured Query Language (SQL) 11–12
 - overview 10–11
- Adventure Deus 176
- aging static data
 - caching 213–216
- Ajax 16
- Apache issue tracking system
 - JIRA 61
- Apache license 50
- Apache Software Foundation 50–51
- application architecture layers
 - business logic layer 17
 - business object model 15
 - overview 14
 - persistence layer
 - abstraction layer 18–19
 - driver or interface 19
 - overview 18
 - persistence framework 19
 - presentation layer 15–17
- relational database
 - integrity 20
 - overview 19–20
 - performance 20–21
 - security 21–22
- application database 22–24
- application server 159
- application source code 4
- Atlassian 51
- atomicity 148–149
- attributes function 166
- attributes interface 256
- autocommit mode 151
- auto-generated keys 110–113
- automatic mapping 94
- automatic result maps 93–95
- automatic transactions 147, 151–152

B

- BaseSqlMapDao class 327
- bash shell script for Linux 59
- BasicAttributes class 256
- batch file for Windows 59
- batch updates, running 115–117
- Bean 103
- bean property names 93
- BeanAction 307
 - ActionContext 307–308
 - BeanAction 307
 - BeanBase 307

- BeanAction (*continued*)
 - overview 306
 - style behavior 317
 - BeanBase 307
 - beans. *See* JavaBeans
 - <behaviorName> () method,
 - public String 317
 - best practices
 - managing iBATIS configuration files 295–298
 - naming conventions
 - overview 298
 - parameter maps 298
 - result maps 299
 - statements 298
 - XML files 299–300
 - overview 288
 - unit testing with iBATIS
 - overview 288
 - unit testing Data Access Objects (DAO) 291–293
 - consumer layers 293, 295
 - unit testing mapping layer 288–291
 - whether to use beans, maps or XML
 - JavaBeans 300
 - maps 300
 - overview 300
 - primitives 301
 - XML 301
 - binary distribution 58
 - binary tags 169–170
 - BLOBs 269
 - boolean values 270, 272
 - booleanToYesNo() method 272
 - build 310
 - build directory 60
 - build.bat process 60
 - build.sh process 60
 - business logic layer 17
 - demarcating transactions at 160–161
 - business object model 15
 - classes 15
 - business-to-business
 - transactions 147
 - bytecode enhancement for lazy loading 62–63
- C**
-
- c:forEach tag 319
 - c:set tags 319
 - cache implementation 201
 - Cache Model configuration 198
 - CacheController 269
 - creating 277
 - interface 199, 276
 - overview 276–277
 - putting, getting and flushing 277–278
 - registering for use 278–279
 - cached object 200
 - CacheKey instance 278
 - cacheModel 325–326
 - instance 278
 - tag 198
 - type attribute 212
 - cacheModelsEnabled 72
 - caching 72
 - aging static data 213–216
 - Cache Model
 - combining read only and serialize 200–201
 - default cache implementation types 198–199
 - overview 198
 - readOnly attribute 199
 - serialize attribute 199
 - types 204–208
 - using tags inside 201–204
 - data 64
 - determining caching
 - strategy 208–209
 - distributed 64
 - iBATIS caching
 - philosophy 197–198
 - overview 196
 - read-only long term data 209–211
 - read-write data 212–213
 - simple example 196–197
 - CallableStatement object 219
 - calling thread 200
 - camel-case pattern 82
 - cart 305
 - catalog 304
 - CatalogBean 315, 319
 - CatalogService 315
 - class 317, 327
 - Category
 - caching 210
 - codes 27
 - object 164
 - parameter class 165
 - table 180
 - categoryCache 210
 - #categoryId# parameter 12
 - categoryId 327
 - name 176
 - property 164, 184
 - CategorySearchCriteria 184
 - CDATA 98
 - CGLIB 62, 131
 - optimized classes 72
 - child records, updating or deleting 115
 - class loaders 65, 71
 - class path 64
 - ClassCastException 65
 - ClassLoader 295
 - Classpath 234, 295–297
 - environment variable 64
 - CLOBs 269
 - close attribute
 - Binary Tag 169
 - Dynamic Tag 168
 - Iterate Tag 174
 - Parameter Tag 173
 - Unary Tag 171
 - close() method 283
 - closeStatement(s) 250
 - coarse grained service
 - methods 17
 - Cocoon 124
 - code coverage 61
 - column 100
 - columnIndex attribute 100
 - com.iBATIS.sql-
 - map.engine.cache.CacheController interface 208
 - commit() method 283
 - commitRequired attribute 76
 - commitTransaction()
 - method 159
 - compareProperty attribute 169
 - Binary Tag 169
 - compareValue attribute 169
 - Binary Tag 169
 - complex keys and relationships 28
 - composite key 28

concurrent updates 114
 Concurrent Version System (CVS) 59
 conditional tags
 simplified 190
 configuration files 43, 50
 managing 295–298
 configure() method 277
 configuring iBatis 48–49
 See also installing and configuring iBatis
 configuring web.xml 311–312
 conjunction attribute 173
 Iterate Tag 174
 Connection instance 156
 Connection object 68
 consistency 149
 constraints 20
 Contact class 256
 control of databases 26–27
 createDao() method 266
 CRUD operations 234
 css directory 309
 culling process 211
 custom transactions
 148, 156–158
 custom type handlers 77
 creating TypeHandlerCallback
 getting results 273
 nulls 273–275
 overview 271
 setting parameters 272
 implementing 270–271
 overview 269–270
 registering TypeHandlerCallback for use 275–276
 CVS 289

D

dao tag 322
 DAO. *See* Data Access Object (DAO)
 dao.xml
 configuration file 221, 248
 configuring 321–322
 file 223, 230, 243
 DaoException 250, 257

DaoFactory interface 264
 DaoManager
 class 221
 creating 221–223
 instance 222
 daoManager.commitTransaction() method 323
 daoManager.endTransaction() method 323
 daoManager.startTransaction() method 323
 DaoManagerBuilder
 instance 222, 235
 DaoService class 221
 DaoTransactionManager 229
 implementation 224
 daoXmlResource variable 222
 data access layer 197, 209, 326
 data access mechanism 220
 Data Access Object (DAO)
 class 241
 configuration 229
 configuring
 context element 223
 Data Access Objects (DAO) elements 229–230
 overview 223
 properties element 223
 tips for 230–233
 transaction manager element 224–229
 creating own DAO layer
 decoupling and creating factory 263–266
 overview 263
 separating interface from implementation 263
 framework 262
 Hibernate DAO implementation
 actual DAO
 implementation 245–248
 defining DAO context 243–244
 mapping Account table 244–245
 overview 243
 hiding implementation details
 overview 218

 reasons for separating data access from data access interface 219–220
 simple example 220–223
 JDBC DAO
 implementation 248–253
 layer 166, 262
 testing 288
 manager 232
 overview 218, 243
 pattern 19, 218, 243
 SQL Map DAO implementation example
 coding DAO
 implementation 239–241
 configuring DAO in iBatis 234–235
 creating DaoManager instance 235
 defining transaction manager 235–236
 loading maps 236–239
 overview 233–234
 unit testing
 DAO consumer layers 293–295
 overview 291–292
 with mock objects 292–293
 using DAO pattern with other data sources
 examples 253–260
 overview 253
 using Spring DAO
 overview 260
 reasons for using instead of iBatis 262
 writing code 260–261
 writing
 interface and implementation 326–327
 overview 323–324
 SQLMap 325–326
 configuration 324–325
 data access object factory 321
 Data Definition Language (DDL) 5, 137
 DDL scripts 189
 data manipulation 16

- Data Manipulation Language (DML) 6
- data source 290
- data transfer objects 15
- database administration teams 27
- database challenges
 - access by multiple disparate systems 27–28
 - complex keys and relationships 28
 - denormalized or overnormalized models 29–30
 - overview 26
 - ownership and control 26–27
 - skinny data models 30–32
- database management system 20
- database percent (%)
 - syntax 179
- database server 46
- database tables 20, 197
- database teams 27
- database transaction 73
- database types
 - application databases 22–23
 - enterprise databases 23–25
 - legacy databases 25–26
 - overview 22
 - proprietary databases 25
- database vendors 111
- database.properties file 324
- databases, age of 22
- DataSource 324
 - connection pool 324
 - DataSourceFactory 269
 - implementation 76
 - interface 279–280
 - element 76–77
 - instance 290
 - interface 219
 - property 225
 - unsupported, configuring
 - DataSourceFactory
 - interface 279–280
 - overview 279
- db.properties properties file 71
- DBCP data source 226–227
- DBCP data source factory 77
- DBJndiContext property 228
- deep copy 199
- default constructor 85
- default keyword 199
- default settings 73
- <delete> statement type 87, 107
- delete method 107
- deleteOrder method 117
- deleting data. *See* updating and deleting data
- demarcating transactions
 - at business logic layer 160–161
 - at presentation layer 159–160
 - overview 158–159
- denormalization 29
- denormalized models 29–30
- dependencies
 - bytecode enhancement for lazy loading 63
 - distributed caching 64
 - Jakarta Commons Database Connection Pool (DBCP) 63–64
 - overview 62
- design concept
 - account 304
 - cart 305
 - catalog 304
 - order 305
 - overview 304
- design layers 268
- dev-javadoc.zip file 62
- devlib 310
 - directory 60
- devsrc directory 60
- Distinguished Name (DN) 254
- distributed caching 64
- distribution of iBATIS, obtaining
 - binary distribution 59
 - building from source
 - digging into repository 59–60
 - overview 59
 - running build 60–61
 - overview 58–59
- .do extension 312
- doc directory 60
- Document Object Model 301
 - DOM object 123
- Document Type Definition (DTD) 50
- domain classes 15
- domain package 309
- dot notation 84
- duplication 89
- durability 151
- <dynamic> parent 181
- <dynamic> tag 165
- dynamic fields 31
- dynamic result mapping 94
- dynamic Structured Query Language (SQL) 8–9, 11, 118
 - advanced techniques
 - applying dynamic SQL tags to static SQL 181–183
 - defining what input is required 179–180
 - defining what resulting data will be 178–179
 - overview 178
 - writing out SQL in static format 180–181
 - alternative approaches to comparing to iBATIS 189–190
 - overview 183–184
 - using Java code 184–187
 - using stored procedures 187–189
 - dynamic tags
 - binary tags 169–170
 - dynamic tag 168
 - iterate tag 174
 - overview 166–167
 - parameter tags 172–173
 - unary tags 171–172
 - dynamic WHERE clause
 - criteria 164–166
 - future of
 - expression language 191
 - overview 190
 - simplified conditional tags 190
 - overview 164
 - simple example
 - apply SQL tags to static SQL 177–178
 - define how to retrieve and display data 176

dynamic Structured Query Language (SQL) (*continued*)
 determine which database structures are involved 176
 overview 176
 write out SQL in static format 176–177
 dynamic tags 166

E

EMMA 61
 enabled property 272
 encapsulated Structured Query Language (SQL) 13
 encapsulation 13
 endTransaction() method 159
 enhancementEnabled 72
 attribute 131
 enterprise databases 23–25
 equal sign (=) 164
 example application
 building application 49
 configuring iBATIS 48–49
 overview 45
 running application 49–50
 setting up database 46
 writing code 46–47
 execute delete statements 107
 executeBatch() method 116
 explicit result maps
 JavaBean and Map results 102–103
 overview 100–101
 primitive results 101–102
 expression language 191
 J2EE 191
 extending iBATIS
 CacheController
 creating 277
 overview 276–277
 putting, getting and flushing 277–278
 registering for use 278–279
 configuring unsupported DataSource
 DataSourceFactory
 interface 279–280

overview 279
 custom type handlers
 creating
 TypeHandlerCallback 271–275
 implementing 270–271
 overview 269–270
 registering TypeHandler-Callback for use 275–276
 customizing transaction management
 overview 280–281
 Transaction interface 282–283
 TransactionConfig interface 281–282
 overview 268
 pluggable component
 design 268–269
 eXtensible Markup Language (XML). *See* XML
 extensions 51
 EXTERNAL 76
 external mapping 95
 external parameter
 mapping 108
 external parameter maps 97
 inserting data using 110
 EXTERNAL transaction 155
 External transaction manager 225
 ExternalDaoTransactionManager
 EXTERNAL type 224
 externalization 11
 externalized Structured Query Language (SQL) 11–12
 ExternalTransactionConfig implementation 280
 eXtreme Programming 288

F

FIFO type, cacheModel 199
 FifoCacheController class 276
 finer grained business logic 17
 First in, first out (FIFO)
 cache 214

cache model 207
 strategy 207
 firstLetter property 184
 flush tags 201
 flush() method 278
 <flushInterval> tag
 201, 203–204
 flushInterval tag 210
 <flushOnExecute> tag 201–203
 flushOnExecute 197, 209
 tag 203, 215
 FOREIGN KEY constraint 20
 FROM clause 12
 future of iBATIS
 additional platforms and languages 51–52
 Apache Software Foundation 50–51
 more extensions and plugins 51
 overview 50
 simpler, smaller, with fewer dependencies 51

G

garbage collection process 66
 Garbage Collector 212
 Gateway pattern 253
 generated keys 111, 113
 generation dynamic SQL tags 190
 getAccountInfoList mapped statement 129
 getAccountInfoListN mapped statement 134
 getAttributeValue method 256
 getChildCategories query mapped statement 211
 getConnection() method 283
 getDao() method 265
 getDataSource() method 279, 282
 getIdDescriptionListByExample mapped statement 239
 getInstance() method 265
 getOrderItem() methods 83
 getOrderItemList mapped statement 130

getPerson() method 292–293
 getProductById query mapped statement 213
 getProductListByCategory 326–327
 method 327
 getProperty 82
 getResourceAsReader() method 296
 getResult() method 273
 getString() method 77, 273
 getValidatedPerson() method 294
 global transactions 148, 156
 context 154
 overview 153
 starting, committing, and ending transaction 155–156
 using active or passive transactions 154–155
 Google 259
 groupBy attribute 132, 140
 GROUENAME parameter 48
 GuiSqlMapConfig.xml 299

H

handleRow method 138
 hash (#) syntax 90
 HashMap 48
 Hibernate 44, 220, 223
 Hibernate DAO implementation
 actual DAO
 implementation 245–248
 defining DAO context 243–244
 mapping Account table 244–245
 overview 243
 Hibernate session factory 225
 Hibernate transaction manager 225, 244
 HIBERNATE type
 HibernateDaoTransactionManager 224
 hibernate.properties file 244
 HibernateDaoTransactionManager
 HIBERNATE type 224
 hotProductsCache tag 215

hours attribute, flushInterval tag 203
 HSQLDB memory database 289
 html:link 319
 HttpServletRequest 307
 HttpServletResponse 307

I

iBATIS

advantages of
 encapsulated Structured Query Language (SQL) 13
 externalized Structured Query Language (SQL) 11–12
 overview 10–11
 example application
 building application 49
 configuring iBATIS 48–49
 overview 45
 running application 49–50
 setting up database 46
 writing code 46–47
 future of
 additional platforms and languages 51–52
 Apache Software Foundation 50–51
 more extensions and plugins 51
 overview 50
 simpler, smaller, with fewer dependencies 51
 how handles common database challenges
 access by multiple disparate systems 27–28
 complex keys and relationships 28
 denormalized or overnormalized models 29–30
 overview 26
 ownership and control 26–27
 skinny data models 30–32
 overview 4–5
 reasons for using
 division of labor 42

open source and
 honesty 43
 portability 42–43
 relationship with application architecture layers
 business logic layer 17
 business object model 15
 overview 14
 persistence layer 18–19
 presentation layer 15–17
 relational database 19–22
 roots of
 dynamic Structured Query Language (SQL) 8–9
 inline Structured Query Language (SQL) 7–8
 modern stored procedures 7
 object relational mapping (ORM) 10
 old-school stored procedures 6
 overview 5
 Structured Query Language (SQL) 5–6
 when not to use
 overview 43
 when application requires fully dynamic Structured Query Language (SQL) 44
 when doesn't work 45
 when have full control 43–44
 when not using relational database 44–45
 working with different database types
 application databases 22–23
 enterprise databases 23–25
 legacy databases 25–26
 overview 22
 proprietary databases 25
See also best practices
See also extending iBATIS
 iBATIS configuration file 290
 ibatis-common-2.jar file 62
 ibatis-dao-2.jar file 62

- ibatis-sqlmap-2.jar file 62
- ibatis-src.zip file 62
- iBSqlMapConfig.xml 290
- id attribute 210
 - cacheModel Tag 198
- IdDescription objects 248
- images directory 310
- implementation 320
- IN parameters 119
- IN statement 164, 187
- <include> statement type
 - 88, 108
- include element 88
- includePassword property 95
- index method 314
- index.jsp 313
- index.tiles definition 314
- indexed properties 83
- InitialContext constructor 228
- initialize() method 279, 281
- inline mapping 95
- inline parameter mapping
 - 97, 108
 - inserting data using 108–109
 - revisited 97–99
- inline parameters
 - using with # placeholders 89–91
 - using with \$ placeholders 91–92
- inline result maps
 - JavaBean and Map results 102–103
 - overview 100–101
 - primitive results 101–102
- Inline SQL 11
- inline Structured Query Language (SQL) 7–8
- inner transaction scope 155
- INOUT parameters 120
- <insert> statement type 87, 107
- insert method 106, 112–113
- INSERT statement 298
- inserting data
 - auto-generated keys 110–113
 - overview 108
 - using external parameter map 110
 - using inline parameter mapping 108–109
- installing and configuring iBATIS
 - adding iBATIS to your application
 - overview 64
 - using iBATIS with stand-alone application 64–65
 - using iBATIS with web application 65
 - dependencies
 - bytecode enhancement for lazy loading 63
 - distributed caching 64
 - Jakarta Commons Database Connection Pool (DBCP) 63–64
 - overview 62
 - distribution contents 62
 - getting iBATIS distribution
 - binary distribution 59
 - building from source 59–61
 - overview 58–59
 - iBATIS and JDBC
 - overview 65–66
 - reducing complexity 67–68
 - releasing JDBC resources 66
 - Structured Query Language (SQL) injection 66–67
 - overview 58, 68
 - properties element 70–71
 - settings element
 - cacheModelsEnabled 72
 - enhancementEnabled 72
 - lazyLoadingEnabled 71–72
 - maxRequests (Deprecated) 73
 - maxSessions (Deprecated) 73
 - maxTransactions (Deprecated) 73
 - overview 71–72
 - useStatementNamespaces 72–73
 - sqlMap element 78
 - Structured Query Language (SQL) Map Configuration file 69–70
 - transactionManager element
 - dataSource element 76–77
 - overview 75–76
 - property elements 76
 - typeAlias elements 73–75
 - typeHandler element 77–78
- int parameter 281
- int property 102
- int typed property 273
- Integer objects 93
- integration database 24
- integrity
 - relational database 20
- interface attribute 322
- interface property 229
- Introspector class, Java 84
- IS keyword 164
- <isEmpty> unary tag 172
- <isEqual> tag 181–182
 - binary dynamic tag 170
- <isGreaterEqual> binary dynamic tag 170
- <isGreaterThan> binary dynamic tag 170
- <isLessEqual> binary dynamic tag 170
- <isLessThan> binary dynamic tag 170
- <isNotEmpty> tag 167, 182
 - unary tag 172
- <isNotEqual> binary dynamic tag 170
- <isNotNull> tag 165, 167
 - unary tag 171
- <isNotParameterPresent> parameter tag 173
- <isNotPropertyAvailable> unary tag 171
- <isNull> tag 165, 167
 - unary tag 171
- isolation-overview 149–150
- isolation-read
 - committed 150
 - uncommitted 150
- isolation-repeatable read 150–151
- isolation-serializable 151
- <isParameterPresent> parameter tag 173
- isProperty 82

<isPropertyAvailable> unary tag 171
 Item table 180
 <iterate> tag 167, 182
 iterate tag 174, 178

J

J2EE 191
 Jakarta Commons Database Connection Pool (DBCP) 63–64
 project 64, 226
 JAR file 62
 Java 42–43
 mapping to LDAP 254–258
 Java code 187
 using as alternative approach to dynamic SQL 184–187
 Java Runtime Environment (JRE) 64
 Java Transaction API (JTA) transaction manager 228
 java.net.URL class 71
 JavaBeans 81, 300
 creating
 bean navigation 84
 overview 81
 what makes a bean 82–84
 JavaBean and Map parameters 99
 javadoc directory 60
 javaType
 attribute 100
 mapping attribute 96
 JDBC 76
 and iBATIS
 overview 65–66
 reducing complexity 67–68
 releasing JDBC resources 66
 Structured Query Language (SQL) injection 66–67
 JDBC DAO 291
 implementation 248–253
 JDBC driver
 implementation 219
 JDBC transaction manager
 DBCP data source 226–227
 JNDI data source 227–228

 overview 225
 SIMPLE data source 225–226
 JDBC type
 JdbcDaoTransactionManager 224
 JDBC.ConnectionURL property
 DBCP data source 227
 SIMPLE data source 225
 JDBC.DefaultAutoCommit property, SIMPLE data source 225
 JDBC.Driver property
 DBCP data source 227
 SIMPLE data source 225
 JDBC.Password property
 DBCP data source 227
 SIMPLE data source 225
 JDBC.Username property
 DBCP data source 227
 SIMPLE data source 225
 JdbcDaoTemplate class 249
 JdbcDaoTransactionManager
 JDBC type 224
 JdbcTransactionConfig implementation 280
 jdbcType
 attribute 100
 mapping attribute 96
 JGameStore application 175
 JIRA 61
 Atlassian's 51
 JMock object 293
 JNDI context 155
 JNDI data source 227–228
 JNDI data source factory 77
 joining related data 95
 JSP directory 309
 JSTL (Java Standard Tag Library) 190
 core tags 319
 JTA 76, 155
 JTA type
 JtaDaoTransactionManager 224
 JtaDaoTransactionManager JTA type 224
 JtaTransactionConfig implementation 280
 JUnit report 61

K

key
 complex 28
 generation technique 111
 parameter 86
 value 111
 keyProperty attribute 112

L

languages 51–52
 layered architecture 159
 layering strategy 14
 layout.catalog definition 314
 layout.main definition 314
 lazy loading 62, 130–132
 bytecode enhancement for 63
 data 10
 lazyLoadingEnabled 71–72
 attribute 131
 LDAP
 using Data Access Object (DAO) with
 mapping from Java to LDAP 254–258
 overview 253–254
 understanding LDAP terminology 254
 LDAP DN attribute 256
 Least Recently Used (LRU) cache model 206–207
 cacheModel 199
 strategy 206, 210
 legacy databases 25–26
 lib 310–311
 List of Map objects 248
 List of Strings 177
 local transactions 147, 151–153
 LruCacheController class 276

M

magic numbers 98, 273
 Main.jsp 313
 managed transactions 156
 manufacturer objects 143
 Map 103
 Map configuration, SQL 198
 Map interface, Java 255

- Map keys 93
- Map objects 140
- Map parameters 99
- mapped statements
 - 73, 85–86, 125
 - creating JavaBeans
 - bean navigation 84
 - overview 81
 - what makes a bean 82–84
 - inline and explicit result maps
 - JavaBean and Map
 - results 102–103
 - overview 100–101
 - primitive results 101–102
 - mapped statement types
 - 87–89
 - mapping parameters
 - external parameter
 - maps 97
 - inline parameter mapping
 - revisited 97–99
 - JavaBean and Map
 - parameters 99
 - overview 95
 - primitive parameters 99
 - overview 81
 - relating objects with
 - avoiding N+1 Selects
 - problem 132–134
 - complex collections
 - 129–130
 - lazy loading 131–132
 - overview 128
- SELECT statements
 - automatic result maps
 - 93–95
 - joining related data 95
 - overview 89
 - Structured Query Language
 - (SQL) injection 92–93
 - using inline parameters
 - with #
 - placeholders 89–91
 - using inline parameters
 - with \$
 - placeholders 91–92
- SqlMap API
 - overview 85
 - queryForList() methods 85

- queryForMap()
 - methods 86
- queryForObject()
 - methods 85
- mapping 133
- mapping parameters
 - external parameter maps 97
 - inline parameter mapping
 - revisited 97–99
 - JavaBean and Map
 - parameters 99
 - overview 95
 - primitive parameters 99
- maps 300
- master configuration file 299
- mathematical addition 148
- maxRequests 73
 - (Deprecated) 73
- maxSessions 73
 - (Deprecated) 73
- maxTransactions 73
 - (Deprecated) 73
- memberSince property 110
- MEMORY cacheModel
 - 199, 205–206
- MemoryCacheController
 - class 276
- methodName() method, public
 - String 307
- milliseconds attribute, flush-
 - Interval tag 203
- minutes attribute, flushInterval
 - tag 203
- mock objects
 - unit testing Data Access
 - Object (DAO) with
 - 292–293
- MockObjects 308
- mode mapping attribute 97
- modern stored procedures 7
- modularization 13
- multiple databases 25
- multiple sub-queries 118
- multiple user interfaces 160
- multi-select dropdown 179
- multi-table select
 - mapping a 95
- MyNiftyTransactionManager
 - class 229
- MySQL 44

N

- N+1 Selects problem
 - 130, 132–134
- name attribute 320
 - property tag 204
- name property 319
- namespace attribute 202
- naming conventions
 - overview 298
 - parameter maps 298
 - result maps 299
 - statements 298
 - XML files
 - master configuration
 - file 299
 - overview 299
 - Structured Query Language
 - (SQL) mapping
 - files 299–300
- NamingException 257
- native file system drivers 19
- natural key 28
- .NET 42–43
- .NET framework 291
- newTransaction() method
 - 281–282
- non-query statements
 - building blocks for updating
 - data
 - overview 106
 - SqlMap API for non-query
 - Structured Query Language (SQL)
 - statements 106–107
 - executing 109
 - executing building blocks for
 - updating data
 - non-query mapped
 - statements 108
 - inserting data
 - auto-generated keys
 - 110–113
 - overview 108
 - using external parameter
 - map 110
 - using inline parameter
 - mapping 108–109
- overview 106
- running batch updates
 - 115–117

non-query statements (*continued*)
 stored procedures
 IN parameters 119
 INOUT parameters 120
 OUT parameters 120–121
 overview 117
 pros and cons 117–119
 updating and deleting data
 handling concurrent
 updates 114
 overview 113–114
 updating or deleting child
 records 115
 normalization 29
 noun-verb separation 17
 null state, parentCategoryId
 property 166
 nullable columns 273
 NullPointerException 102
 nullValue
 attribute 101
 mapping attribute 97
 NUnit 291

O

O/RM tools 198
 Object / Relational Mapping
 (O/RM) 10–11
 solution 44
 tools 81, 114
 object identification (OID) 198
 object identity 143
 ObjectRelationalBridge
 (OJB) 228
 transaction manager 228
 ODBC API, Microsoft's 66
 OjbBrokerTransactionManager
 OJB type 224
 old-school stored procedures 6
 open attribute
 Binary Tag 169
 Dynamic Tag 168
 Iterate Tag 174
 Parameter Tag 172
 Unary Tag 171
 value 173
 OpenSymphony cache
 (OSCache) 64
 OraclePerson.xml 300
 order 305

Order object 114
 OrderItem objects 83, 114, 128
 orderItemList property 130, 134
 orderList property 129
 org.apache.ibatis.jgamestore.domain
 package 316
 org.apache.ibatis.jgamestore.persis-
 tence.sqlmapdao
 package 327
 org.apache.ibatis.jgamestore.ser-
 vice package 317
 OSCACHE
 cache model 199, 208
 OSCache 2.0 208
 OSCache jars 208
 oscache.properties file 208
 OSCacheController class 276
 OUT parameters 120–121
 overcorrection 7
 overnormalize a database 29
 overnormalized models 29–30
 overview 306
 ownership of databases 26–27

P

packages 25
 pages directory 312
 pageSize parameter 178
 PaginatedList 327
 parameter attribute 307, 314
 parameter class type 182
 parameter element 123
 parameter maps 121
 naming 298
 parameter object 166
 parameter property 169
 parameter tags 172–173
 parameterClass attribute 326
 parameters 177
 ParameterSetter interface 272
 parent tag 165
 parentCategoryId 210
 property 164
 passive transactions 154–155
 \$PATH variable, Linux 64
 %PATH% variable, Windows 64
 performance, relational
 database 20–21
 performance-to-effort ratio 197
 persistence 306
 persistence layer 16, 160
 abstraction layer 18–19
 driver or interface 19
 framework 14
 overview 18
 persistence framework 19
 persistence package 309
 Person.xml mapping file 298
 PersonDao interface 294
 plain old Java object
 (POJO) 123
 platforms 51–52
 pluggable component
 design 268–269
 pluggable interfaces 45
 plug-ins 51
 polar extremes 117
 Pool.MaximumActiveCon-
 nections property
 DBCP data source 227
 SIMPLE data source 226
 Pool.MaximumCheckoutTime
 property, SIMPLE data
 source 226
 Pool.MaximumIdleConnections
 property
 DBCP data source 227
 SIMPLE data source 226
 Pool.MaximumWait property,
 DBCP data source 227
 Pool.PingConnectionsNotUsed-
 For property, SIMPLE data
 source 226
 Pool.PingConnectionsOlder-
 Than property, SIMPLE
 data source 226
 Pool.PingEnabled property,
 SIMPLE data source 226
 Pool.PingQuery property, SIM-
 PLE data source 226
 Pool.TimeToWait property, SIM-
 PLE data source 226
 Pool.ValidationQuery property,
 DBCP data source 227
 portability 42–43
 PostgreSQL 220
 database 137
 PreparedStatement 66, 187, 251
 objects 116, 219

- prepend attribute 165, 182
 - Binary Tag 169
 - Dynamic Tag 168
 - Iterate Tag 174
 - Parameter Tag 172
 - Unary Tag 171
 - prepend value 182
 - presentation 305
 - setting up 312, 320
 - presentation layer 15–17, 160
 - demarcating transactions at 159–160
 - presentation package 309
 - presentation, setting up 312–319
 - primary key 28
 - values 110
 - primitive integer value 113
 - primitive parameters 99
 - primitive wrapper classes 99
 - primitives 301
 - <procedure> statement type 87, 107
 - product objects 143
 - Product table 176, 180
 - Product.java 316
 - Product.xml 325
 - productCache 325
 - cache model 326
 - ProductDao interface 326–327
 - ProductionWebSqlMap-Config.xml 299
 - ProductSqlMapDao class 327
 - productType parameter 173
 - productType property 172
 - properties element 70–71
 - properties file 70
 - Properties object 204
 - property attribute 100, 171, 319
 - Binary Tag 169
 - Iterate Tag 174
 - Unary Tag 171
 - property elements 76
 - property mapping attribute 96
 - property tag 197, 204, 321
 - PropertyDescriptor objects 84
 - proprietary databases 25
 - public Category getCategory(Integer categoryId) 317
 - public List getProductListByCategory(Integer categoryId) 317
 - public String 307, 317
 - putObject() method 278
- Q**
-
- query mapping tool 81
 - query techniques, advanced
 - overview 123
 - processing extremely large data sets
 - overview 138
 - RowHandler example 141
 - RowHandler interface 138–143
 - relating objects with mapped statements
 - avoiding N+1 selects
 - problem 132–134
 - complex collections 129–131
 - lazy loading 131–132
 - overview 128
 - using statement type and data definition language (DDL) 137
 - using XML with iBATIS
 - overview 123
 - XML parameters 123–125
 - XML results 125–127
 - queryForList() methods 85, 263
 - queryForMap() methods 86, 140
 - queryForObject() methods 85, 102
 - queryForPaginatedList class 327
 - method 178, 183
 - queryWithRowHandler method 139
- R**
-
- range clause 150
 - range lock 150
 - RDBMS software 21
 - readOnly attribute 199, 213
 - cacheModel Tag 198
 - read-only long term data
 - caching 209–211
 - read-write data
 - caching 212–213
 - reasons for using iBATIS
 - division of labor 42
 - open source and honesty 43
 - portability 42–43
 - redundancy 29
 - reference-type 205
 - property, MEMORY cache 206
 - relational database
 - integrity 20
 - overview 19–20
 - performance 20–21
 - security 21–22
 - systems 21
 - relational database management systems (RDBMS) 4
 - relationships, complex 28
 - remapResults attribute 95
 - removeFirstPrepend attribute 167, 190
 - Binary Tag 169
 - Iterate Tag 174
 - Parameter Tag 173
 - Unary Tag 171
 - reporting database 24
 - resource attribute 71, 78
 - Resources class 222
 - result class type 182
 - result maps 103, 132
 - automatic 93–95
 - inline and explicit
 - JavaBean and Map results 102–103
 - overview 100–101
 - primitive results 101–102
 - naming 299
 - ResultAccountInfoMap 129
 - result map 129
 - resultClass 326
 - ResultGetter interface 273
 - ResultOrderInfoMap 129
 - ResultOrderInfoNMap
 - result map 134
 - ResultOrderItemMap 129
 - ResultSet handlers 51

- ResultSet object 68
 - return value 85
 - returnValue variable 112
 - rollback() method 283
 - roots of iBATIS
 - dynamic Structured Query Language (SQL) 8–9
 - inline Structured Query Language (SQL) 7–8
 - modern stored procedures 7
 - object relational mapping (ORM) 10
 - old-school stored procedures 6
 - overview 5
 - Structured Query Language (SQL) 5–6
 - RowHandler
 - example 141
 - interface 138–143
- S**
-
- scope attribute 320
 - SCOPE_IDENTITY
 - function 111
 - searchProductList mapped statement 178
 - seconds attribute, flushInterval tag 203
 - security, relational database 21–22
 - <select> element 13
 - <select> statement type 87
 - mapped statement 87
 - SELECT ... FROM Products 177
 - select attribute 101
 - SELECT clause 177
 - SELECT statements
 - automatic result maps 93–95
 - joining related data 95
 - overview 89
 - Structured Query Language (SQL) injection 92–93
 - using inline parameters with # placeholders 89–91
 - using inline parameters with \$ placeholders 91–92
 - selectKey element 111–112
 - selectKey statement 116
 - serialize attribute, cacheModel Tag 198
 - server.properties file 231
 - service 305–306
 - service classes 17
 - service interface 320
 - service layer 320, 326
 - service package 309
 - service, writing
 - configuring dao.xml 321–322
 - overview 320–321
 - transaction demarcation 322–323
 - Servlet filter 159
 - Servlet specification 67
 - servlet tag 311
 - servlet-class tag 311
 - servlet-mapping tag 312
 - Servlets 67
 - session.name property 228
 - setDataSource() method 282
 - setDate() method 272
 - setInt() method 272
 - setObject 187
 - setParameter() method 272
 - setString() method 272
 - settings element 72
 - cacheModelsEnabled 72
 - enhancementEnabled 72
 - lazyLoadingEnabled 71–72
 - maxRequests
 - (Deprecated) 73
 - maxSessions (Deprecated) 73
 - maxTransactions
 - (Deprecated) 73
 - overview 71–72
 - useStatementNamespaces 72–73
 - shared key 260
 - .shtml extension 315
 - SIMPLE data source 225–226
 - SIMPLE data source factory 77
 - SIMPLE DataSource 158
 - SimpleDataSource, iBATIS 225
 - simplified conditional tags 190
 - single-table select, mapping a 95
 - size property 216
 - skinny data models 30–32
 - SOFT reference, MEMORY
 - cache 205
 - software tuning 21
 - spaghetti code 14
 - specific key 260
 - Spring class 260
 - Spring DAO
 - overview 260
 - reasons for using instead of iBATIS 262
 - writing code 260–261
 - Spring framework
 - 159–160, 256, 260
 - <sql> statement type 88, 108
 - sql element 88
 - SQL execution engines 51
 - SQL fragment 239
 - SQL Map 72, 223, 262
 - Config file 156
 - configuration file 68
 - version 243
 - SQL parser 8
 - SQL statements 42
 - mappings 288
 - SQL string substitution
 - syntax 67
 - SQLException method 250
 - SqlExecutor class 117
 - SQLJ, in Java 8
 - SQLMAP
 - SqlMapDaoTransactionManager 224
 - transaction manager 228
 - type 321
 - SQLMap 68, 325–326
 - configuration 324–325
 - SqlMap API
 - for non-query Structured Query Language (SQL) statements
 - delete method 107
 - insert method 106
 - overview 106
 - update method 106–107
 - overview 85
 - queryForList() methods 85
 - queryForMap() methods 86
 - queryForObject()
 - methods 85
 - sqlMap configuration 203
 - sqlMap elements 78
 - sqlmap files 324
 - SqlMap.xml file 46
 - SqlMapBuilder interface 296

- SqlMapClient interface
 - 85, 268, 292–293
 - SqlMapClient object 263
 - SqlMapClient unit test 290–291
 - SqlMapConfig file 68, 321
 - SqlMapConfig.xml 46–47, 78, 131, 234, 296, 299, 324, 326
 - SqlMapConfigResource
 - property 228, 235, 321
 - SqlMapConfigURL
 - property 228
 - SqlMapDaoTemplate class
 - 241, 327
 - SqlMapDaoTransactionManager
 - SQLMAP type 224
 - SqlMapExecutor instance 241
 - SqlMapPersonDao
 - implementation 292
 - SqlMaps framework 189
 - src 308–309
 - src directory 60, 309
 - standard web application 305
 - startTransaction() method 159
 - startup script 64
 - Stateless Session Beans 159–160
 - <statement> statement type 88
 - statement attribute 202
 - Statement object 68
 - statements, naming 298
 - static data, aging, caching
 - 213–216
 - static SQL 164
 - stored procedures 11, 117, 188
 - IN parameters 119
 - INOUT parameters 120
 - modern 7
 - old-school 6
 - OUT parameters 120–121
 - overview 117
 - pros and cons 117–119
 - using as alternative approach
 - to dynamic SQL 187–189
 - String 77, 301
 - string buffer 127
 - String data type 12
 - String parameter 48
 - String value 123
 - StringTypeHandler 77
 - Strong Memory Cache 211
 - STRONG reference, MEMORY cache 205
 - Structured Query Language (SQL) 4–6
 - dynamic 8–9
 - encapsulated 13
 - externalized 11–12
 - fully dynamic 44
 - injection 66–67, 92–93
 - inline 7–8
 - mapping files 299–300
 - See also* dynamic Structured Query Language (SQL)
 - Struts 67, 262, 305, 312
 - Struts Action classes 308
 - Struts html tags 319
 - Struts taglibs 190
 - struts-config.xml 313, 319
 - location 311
 - StrutsTestCase 308
 - substitution (\$) syntax 91, 93
 - Subversion (SVN) 289
 - repository 58–59
 - source control 51
 - Sun 65
- T**
-
- tag attributes 168–169, 172
 - technologies, choosing
 - overview 305
 - persistence 306
 - presentation 305
 - service 305–306
 - TellerService class 17
 - template pattern 260
 - test 309
 - test directory 60
 - TestWebSqlMapConfig.xml 299
 - tickets 117
 - tiles-defs.xml 313, 315
 - TIMESTAMP database type 110
 - Tomcat 65
 - tools directory 60
 - TOPLINK
 - transaction manger 228
 - type, ToplinkDaoTransactionManager 224
 - transaction demarcation 152
 - transaction interface 281–283
 - transaction management 75
 - transaction manager 48, 321
 - configuration 152, 290
 - transaction manager element
 - External transaction manager 225
 - Hibernate transaction manager 225
 - Java Transaction API (JTA) transaction manager 228
 - JDBC transaction manager 225
 - DBCP data source 226–227
 - JNDI data source 227–228
 - overview 225
 - SIMPLE data source 225–226
 - ObjectRelationalBridge (OBJ) transaction manager 228
 - SQLMAP transaction manger 228
 - TOPLINK transaction manger 228
 - TransactionConfig 269
 - TransactionConfig class 282
 - TransactionConfig instance 282
 - TransactionConfig
 - interface 281–282
 - transactionManager 324
 - dataSource element 76–77
 - element 235
 - overview 75–76
 - property elements 76
 - tag 321
 - transactions
 - automatic 151–152
 - custom 156–158
 - demarcating
 - at business logic layer 160–161
 - at presentation layer 159–160
 - overview 158–159
 - global
 - overview 153
 - starting, committing, and ending
 - transaction 155–156
 - using active or passive transactions 154–155
 - local 152–153
 - overview 146

transactions (*continued*)

- properties
 - atomicity 148–149
 - consistency 149
 - durability 151
 - isolation-overview 149–150
 - isolation-read
 - committed 150
 - isolation-read
 - uncommitted 150
 - isolation-repeatable
 - read 150–151
 - isolation-serializable 151
 - overview 148
 - simple banking example 146–148
- two-tier applications 6
- type attribute 75, 199, 320
 - cacheModel Tag 198
- type handler callback 77
- typeAlias 325
 - elements 73–75
- <typeHandler> element 275
- typeHandler element 77–78
- TypeHandler interface 270
- typeHandler mapping
 - attribute 97
- TypeHandlerCallback 269
 - class 275
 - creating
 - getting results 273
 - nulls 273–275
 - overview 271
 - setting parameters 272
 - interface 270, 272
 - registering for use 275–276

U

unary tags 171–172

UNIQUE constraint 20

unit testing with iBATIS

- Data Access Object (DAO)
 - consumer layers 293–295
- Data Access Objects (DAO)
 - overview 291–292
 - unit testing DAO with mock objects 292–293
- mapping layer
 - database scripts 289–290

- iBATIS configuration file
 - (i.e. SqlMapConfig.xml) 290
- iBATIS SqlMapClient unit
 - test 290–291
- overview 288
- test database instance 288–289
- overview 288
- <update> statement type 87, 107
- update method 106–107
- UPDATE statement 298
- update statement 114
- updating and deleting data
 - handling concurrent updates 114
 - overview 113–114
 - updating or deleting child records 115
- updating data
 - building blocks for
 - non-query mapped statements 108
 - overview 106
 - SqlMap API for non-query Structured Query Language (SQL) statements 106–107
- url attribute 71, 78
- User class 272
- user-javadoc.zip file 62
- UserTransaction instance 154, 281
- UserTransaction property 228
- useStatementNamespaces 72–73

V

validate attribute 320

ValidatorActionForm 307

#value# string 90

value attribute, property tag 204

value parameter 86

value property 82

valueOf() method 274

view category 323

viewCategory behavior 317

W

Weak Memory cache 212

WEAK reference, MEMORY

- cache 205

web 309–310

web service, using Data Access Object (DAO) with 258–260

web.xml

- configuring 311–312

WEB-INF

- /classes directory 235
- /lib directory 65, 235
- directory 310

WebServiceGateway

- interface 253

WebSqlMapConfig.xml 299

WebWork 306

WHERE clause 177

wide tables 29

wrapper class 101

X

XML 13, 301

- documents 50
- using with iBATIS
 - overview 123
 - XML parameters 123–125
 - XML results 125–127

XML files 299

- naming
 - master configuration file 299
 - overview 299
 - Structured Query Language (SQL) mapping files 299–300

Y

YesNoTypeHandlerCallback

- interface 274

Z

ZIP file 62