iBATIS in Action

CLINTON BEGIN
BRANDON GOODIN
LARRY MEADORS

MANNING
Greenwich
(74° w. long.)
contents

preface xvii
acknowledgments xix
about this book xxiii
about the authors xxvii
about the title xxviii
about the cover illustration xxix

PART 1 INTRODUCTION ..................................................1

1 The iBATIS philosophy 3

1.1 A hybrid solution: combining the best of the best 4

   Exploring the roots of iBATIS 5
   Understanding the iBATIS advantage 10

1.2 Where iBATIS fits 14

   The business object model 15 • The presentation layer 15
   The business logic layer 17 • The persistence layer 18
   The relational database 19

1.3 Working with different database types 22

   Application databases 22 • Enterprise databases 23
   Proprietary databases 25 • Legacy databases 25
1.4 How iBATIS handles common database challenges 26
   - Ownership and control 26
   - Access by multiple disparate systems 27
   - Complex keys and relationships 28
   - Denormalized or overnormalized models 29
   - Skinny data models 30

1.5 Summary 32

2 What is iBATIS? 33
2.1 Mapping SQL 35
2.2 How it works 37
   - iBATIS for small, simple systems 38
   - iBATIS for large, enterprise systems 39
2.3 Why use iBATIS? 40
   - Simplicity 40
   - Productivity 41
   - Performance 41
   - Separation of concerns 42
   - Division of labor 42
   - Portability: Java, .NET, and others 42
   - Open source and honesty 43
2.4 When not to use iBATIS 43
   - When you have full control... forever 43
   - When your application requires fully dynamic SQL 44
   - When you're not using a relational database 44
   - When it simply does not work 45
2.5 iBATIS in five minutes 45
   - Setting up the database 46
   - Writing the code 46
   - Configuring iBATIS (a preview) 47
   - Building the application 49
   - Running the application 49
2.6 The future: where is iBATIS going? 50
   - Apache Software Foundation 50
   - Simpler, smaller, with fewer dependencies 51
   - More extensions and plug-ins 51
   - Additional platforms and languages 51
2.7 Summary 52

PART 2 iBATIS BASICS ........................................55

3 Installing and configuring iBATIS 57
3.1 Getting an iBATIS distribution 58
   - Binary distribution 59
   - Building from source 59
3.2 Distribution contents 62
3.3 Dependencies 62
  Bytecode enhancement for lazy loading 63 • Jakarta Commons
  Database Connection Pool 63 • Distributed caching 64

3.4 Adding iBATIS to your application 64
  Using iBATIS with a stand-alone application 64
  Using iBATIS with a web application 65

3.5 iBATIS and JDBC 65
  Releasing JDBC resources 66 • SQL injection 66
  Reducing the complexity 67

3.6 iBATIS configuration continued 68
  The SQL Map configuration file 69 • The <properties> element 70
  The <settings> element 71 • The <typeAlias> elements 73 • The
  <transactionManager> element 75 • The <typeHandler>
  element 77 • The <sqlMap> element 78

3.7 Summary 78

4 Working with mapped statements 80
4.1 Starting with the basics 81
  Creating JavaBeans 81 • The SqlMap API 85
  Mapped statement types 86

4.2 Using <select> mapped statements 89
  Using inline parameters with the # placeholders 89 • Using
  inline parameters with the $ placeholders 91 • A quick look
  at SQL injection 92 • Automatic result maps 93
  Joining related data 95

4.3 Mapping parameters 95
  External parameter maps 95 • Inline parameter
  mapping revisited 97 • Primitive parameters 99
  JavaBean and Map parameters 99

4.4 Using inline and explicit result maps 100
  Primitive results 101 • JavaBean and Map results 102

4.5 Summary 103

5 Executing nonquery statements 105
5.1 The building blocks for updating data 106
  The SqlMap API for nonquery SQL statements 106
  Nonquery mapped statements 107
5.2 Inserting data 108
  Using inline parameter mapping 108 ■ Using an external parameter map 109 ■ Autogenerated keys 110

5.3 Updating and deleting data 113
  Handling concurrent updates 114
  Updating or deleting child records 114

5.4 Running batch updates 115

5.5 Working with stored procedures 117
  Considering the pros and cons 117
  IN, OUT, and INOUT parameters 119

5.6 Summary 121

6 Using advanced query techniques 122

6.1 Using XML with iBATIS 123
  XML parameters 123 ■ XML results 125

6.2 Relating objects with mapped statements 128
  Complex collections 128 ■ Lazy loading 131
  Avoiding the N+1 Selects problem 132

6.3 Inheritance 134
  Mapping Inheritance 136

6.4 Other miscellaneous uses 137
  Using the statement type and DDL 137
  Processing extremely large data sets 138

6.5 Summary 144

7 Transactions 145

7.1 What is a transaction? 146
  A simple banking example 146
  Understanding transaction properties 148

7.2 Automatic transactions 151

7.3 Local transactions 152

7.4 Global transactions 153
  Using active or passive transactions 154 ■ Starting, committing, and ending the transaction 155 ■ Do I need a global transaction? 156

7.5 Custom transactions 156
### 7.6 Demarcating transactions 158
- Demarcating transactions at the presentation layer 159
- Demarcating transactions at the persistence layer 160
- Demarcating transactions at the business logic layer 160

### 7.7 Summary 161

### 8 Using Dynamic SQL 163

#### 8.1 Dealing with Dynamic WHERE clause criteria 164

#### 8.2 Getting familiar with the dynamic tags 166
- The `<dynamic>` tag 168
- Binary tags 169
- Unary tags 171
- Parameter tags 172
- The `<iterate>` tag 173

#### 8.3 A complete simple example 175
- Defining how to retrieve and display data 176
- Determining which database structures are involved 176
- Writing the SQL in static format 176
- Applying Dynamic SQL tags to static SQL 177

#### 8.4 Advanced Dynamic SQL techniques 178
- Defining the resulting data 178
- Defining the required input 179
- Writing the SQL in static format 180
- Applying Dynamic SQL tags to static SQL 181

#### 8.5 Alternative approaches to Dynamic SQL 183
- Using Java code 184
- Using stored procedures 187
- Comparing to iBATIS 189

#### 8.6 The future of Dynamic SQL 190
- Simplified conditional tags 190
- Expression language 191

#### 8.7 Summary 191

### Part 3 iBATIS in the Real World .......................... 193

### 9 Improving performance with caching 195

#### 9.1 A simple iBATIS caching example 196

#### 9.2 iBATIS's caching philosophy 197

#### 9.3 Understanding the cache model 198
- Type 198
- The `readOnly` attribute 199
- The `serialize` attribute 199
- Combining `readOnly` and `serialize` 200
9.4 Using tags inside the cache model 201
   Cache flushing 201
   Setting cache model implementation properties 204

9.5 Cache model types 204
   MEMORY 205 • LRU 206 • FIFO 207 • OSCACHE 208
   Your cache model here 208

9.6 Determining a caching strategy 208
   Caching read-only, long-term data 209 • Caching read-write data 211 • Caching aging static data 213

9.7 Summary 216

10 iBATIS data access objects 217

10.1 Hiding implementation details 218
   Why the separation? 219 • A simple example 220

10.2 Configuring the DAO 223
   The <properties> element 223 • The <context> element 223
   The <transactionManager> element 224 • The DAO elements 229

10.3 Configuration tips 230
   Multiple servers 230 • Multiple database dialects 231
   Runtime configuration changes 232

10.4 A SQL Map DAO implementation example 233
   Configuring the DAO in iBATIS 234 • Creating a DaoManager instance 235 • Defining the transaction manager 235 • Loading the maps 236 • Coding the DAO implementation 239

10.5 Summary 241

11 Doing more with DAO 242

11.1 Non-SQLMap DAO implementations 243
   A Hibernate DAO implementation 243
   A JDBC DAO implementation 248

11.2 Using the DAO pattern with other data sources 253
   Example: using a DAO with LDAP 253
   Example: using a DAO with a web service 258

11.3 Using the Spring DAO 260
   Writing the code 260 • Why use Spring instead of iBATIS? 262
CONTENTS

11.4 Creating your own DAO layer 262
   Separating interface from implementation 263
   Decoupling and creating a factory 263
11.5 Summary 266

12 Extending iBATIS 267
12.1 Understanding pluggable component design 268
12.2 Working with custom type handlers 269
   Implementing a custom type handler 270
   Creating a TypeHandlerCallback 271
   Registering a TypeHandlerCallback for use 275
12.3 Working with a CacheController 276
   Creating a CacheController 277 • Putting, getting, and flushing a
   CacheController 277 • Registering a CacheController for use 278
12.4 Configuring an unsupported DataSource 279
12.5 Customizing transaction management 280
   Understanding the TransactionConfig interface 281
   Understanding the Transaction interface 282
12.6 Summary 283

PART 4 iBATIS RECIPES .........................................................285

13 iBATIS best practices 287
13.1 Unit testing with iBATIS 288
   Unit-testing the mapping layer 288 • Unit-testing data access
   objects 291 • Unit-testing DAO consumer layers 293
13.2 Managing iBATIS configuration files 295
   Keep it on the classpath 295 • Keep your files together 297
   Organize mostly by return type 298
13.3 Naming conventions 298
   Naming statements 298 • Naming parameter maps 298
   Naming result maps 299 • XML files 299
13.4 Beans, maps, or XML? 300
   JavaBeans 300 • Maps 300 • XML 301 • Primitives 301
13.5 Summary 301
14 Putting it all together 303
14.1 Design concept 304
   Account 304 • Catalog 304 • Cart 305 • Order 305
14.2 Choosing technologies 305
   Presentation 305 • Service 305 • Persistence 306
14.3 Tweaking Struts: the BeanAction 306
   BeanBase 307 • BeanAction 307 • ActionContext 307
14.4 Laying the foundation 308
   src 308 • test 309 • web 309
   build 310 • devlib 310 • lib 310
14.5 Configuring the web.xml 311
14.6 Setting up the presentation 312
   The first step 312 • Utilizing a presentation bean 315
14.7 Writing your service 320
   Configuring dao.xml 321 • Transaction demarcation 322
14.8 Writing the DAO 323
   SQLMap configuration 324 • SQLMap 325
   Interface and implementation 326
14.9 Summary 328

appendix iBATIS.NET Quick Start 329
index 337