

Grails

IN ACTION

Glen Smith
Peter Ledbrook

FOREWORD BY Dierk König



contents

<i>foreword</i>	<i>xvii</i>
<i>preface</i>	<i>xix</i>
<i>acknowledgments</i>	<i>xxi</i>
<i>about this book</i>	<i>xxiv</i>
<i>about the title</i>	<i>xxviii</i>
<i>about the cover illustration</i>	<i>xxix</i>

PART 1 INTRODUCING GRAILS..... 1

1 Grails in a hurry... 3

1.1 Why Grails? 4

First there was Rails... 4 ■ *Why Grails changed the game 5*
Big idea #1: Convention over Configuration 6 ■ *Big idea #2: agile philosophy 6* ■ *Big idea #3: rock-solid foundations 7* ■ *Big idea #4: scaffolding and templating 7* ■ *Big idea #5: Java integration 8* ■ *Big idea #6: incredible wetware 8* ■ *Big idea #7: productivity ethos 9*

1.2 Getting set up 9

1.3 Our sample program: a Web 2.0 QOTD 10

Writing your first controller 11 ■ *Writing stuff out: the view 13*
Adding some style with Grails layouts 15

- 1.4 Creating the domain model 17
 - Configuring the data source* 18
 - *Exploring database operations* 19
- 1.5 Adding UI actions 20
 - Scaffolding: just add rocket fuel* 22
 - *Surviving the worst case scenario* 23
- 1.6 Improving the architecture 24
 - Your first Grails test case* 26
 - *Going Web 2.0: Ajax-ing the view* 27
 - *Bundling the final product: creating a WAR file* 29
 - *And 55 lines of code later* 29
- 1.7 Summary and best practices 30

2 *The Groovy essentials* 31

- 2.1 An introduction 32
 - Let's play with Groovy!* 32
 - *Some basic differences from Java* 34
 - *Some new operators* 36
- 2.2 Exploring types 38
 - Looking at basic types* 39
 - *Syntax sugar for lists, maps, and ranges* 42
- 2.3 Time-saving features 44
 - Who needs semicolons?* 44
 - *Native regular expressions* 46
 - *Property notation* 48
 - *Anything in Java that's not in Groovy?* 49
- 2.4 Expert Groovy 50
 - Discovering closures* 50
 - *Programming dynamically* 53
 - The Groovy JDK* 55
 - *Creating and parsing XML the easy way* 58
- 2.5 Summary and best practices 60

PART 2 CORE GRAILS 63

3 *Modeling the domain* 65

- 3.1 Hubbub: starting our example application 66
 - Domain-driven design* 67
 - *Hubbub kick-start: from 0 to first hit* 68
 - *Introducing GORM (Grails object relational mapping)* 69
- 3.2 Your first domain class object 69
 - Saving and retrieving users via tests* 70
 - *Updating user properties* 72
 - *Deleting users* 73

- 3.3 Validation: stopping garbage in and out 74
 - Standard validators* 76
 - *Custom validation with regular expressions* 78
 - *Cross-field validation tricks* 78
- 3.4 Defining the data model—1:1, 1:m, m:n 79
 - One-to-one relationships* 79
 - *One-to-many relationships* 82
 - Many-to-many relationships* 86
 - *Self-referencing relationships* 89
- 3.5 Summary and best practices 90

4 *Putting the model to work* 92

- 4.1 Creating instant UIs with scaffolding 93
 - Scaffolding Hubbub's domain classes* 93
 - *Scaffolding and validation* 95
 - *Customizing error messages* 96
 - Managing relationships via scaffolds* 98
 - *Tweaking scaffold layouts with CSS* 100
 - *What can't you do with dynamic scaffolds?* 102
 - *Static scaffolding: generating and customizing scaffold code* 102
 - *Customizing scaffolding templates: building your own UI generator* 104
- 4.2 Groovy querying with dynamic finders and Query by Example 104
 - Implementing a basic search form* 105
 - *The many faces of dynamic finders* 107
 - *Tuning dynamic finders with eager and lazy fetching* 109
 - *When dynamic finders don't deliver* 110
 - Introducing Query by Example (QBE)* 110
 - *Getting dynamic with list(), listOrderBy(), and countBy()* 111
- 4.3 More sophisticated query options 112
 - With great power: criteria querying* 112
 - *Dynamic queries with criteria* 114
 - *Creating a tag cloud using report-style query projections* 116
 - *Using HQL directly* 117
- 4.4 Bootstrapping reference data 118
- 4.5 Summary and best practices 119

5 *Controlling application flow* 121

- 5.1 Controller essentials 122
 - Implementing a timeline for Hubbub* 123
 - *Adding new posts* 125
 - *Exploring scopes* 128
 - *Handling default actions* 131
 - *Working with redirects* 132
- 5.2 Services: making things robust and maintainable 133
 - Implementing a PostService* 133
 - *Wiring our PostService to our PostController* 134

- 5.3 Data binding 136
 - Binding to an existing object* 136
 - *Whitelist and blacklist bind params* 137
 - *Complex forms: binding multiple objects* 137
 - *Error handling* 140
 - 5.4 Command objects 141
 - Handling custom user registration forms* 141
 - *Participating in injection* 144
 - 5.5 Working with images 144
 - Handling file uploads* 144
 - *Uploading to the filesystem* 146
 - *Rendering photos from the database* 146
 - 5.6 Intercepting requests with filters 148
 - Writing your first filter* 148
 - *Filter URL options* 150
 - 5.7 Creating custom URL mappings 151
 - myHubbub: rolling your own permalinks* 152
 - *Optional variables and constraints* 152
 - *Handling response codes* 153
 - *Mapping directly to the view* 153
 - *Wildcard support* 153
 - 5.8 Summary and best practices 154
- 6** *Developing tasty views, forms, and layouts* 155
- 6.1 Understanding the core form tags 156
 - A handful of essential tags* 156
 - *A pocketful of link tags* 157
 - *A tour of the form tags* 158
 - *Adding pagination to the timeline* 164
 - 6.2 Extending views with your own tags 165
 - Simple tags* 165
 - *Logical tags* 167
 - *Iteration tags* 168
 - Calling one tag from another* 169
 - 6.3 Adding delicious layouts 170
 - Introducing SiteMesh* 170
 - *Standardizing page layouts* 173
 - Fragment layouts with templates* 175
 - *Adding skinning* 176
 - Implementing navigation tabs* 177
 - 6.4 Applying Ajax tags 179
 - Choosing a JavaScript library* 179
 - *Essential Ajax form remoting* 180
 - *Sizzle++: going further with animation and effects* 182
 - 6.5 Summary and best practices 186

7 *Building reliable applications* 188

- 7.1 Why should we test software? 188
- 7.2 Unit testing 190
 - Testing domain classes* 191 ▪ *Testing services* 195 ▪ *General mocking in Grails* 198 ▪ *Testing controllers* 201 ▪ *Testing tag libraries* 204
- 7.3 Integration testing 206
 - Filling the gaps* 207 ▪ *When only an integration test will do* 210
- 7.4 Functional testing 211
 - Introducing the Functional Test plugin* 212 ▪ *Other testing tools* 215
- 7.5 Summary and best practices 216

PART 3 EVERYDAY GRAILS 219

8 *Using plugins: adding Web 2.0 in 60 minutes* 221

- 8.1 Taking advantage of others' hard work 222
 - Finding plugins* 223 ▪ *Installing plugins* 225
- 8.2 Adding charts and graphs 228
 - Installing the Google Chart plugin* 228 ▪ *Creating your first chart* 229 ▪ *What's the story with dataType?* 230 ▪ *Bar charts: setting custom colors and gridlines* 231 ▪ *Line charts: handling multiple datasets and line styles* 232 ▪ *Intranet charts: Google Chart without Google* 233
- 8.3 Adding mail support 233
 - Sending mail inline* 235 ▪ *Using a view as your mail body* 236
- 8.4 Full-text search: rolling your own search 237
 - Making objects searchable* 238 ▪ *Highlighting hit terms* 241
 - Implementing pagination* 243 ▪ *Customizing what gets indexed* 244 ▪ *Query suggestions: did you mean "Grails"?* 245
 - Searching across relationships* 246
- 8.5 GrailsUI makeover 248
 - Adding tooltips* 249 ▪ *Implementing rich-text editing* 249
 - Implementing calendar-style dates* 251 ▪ *Introducing autocomplete* 252
- 8.6 Summary and best practices 253

9 *Wizards and workflow with webflows* 255

- 9.1 What is a webflow? 256
 - Writing your first flow: a checkout wizard* 257
 - Anatomy of a flow state* 259
- 9.2 Working with webflows 261
 - Flow scope: better than Flash scope, cheaper than session scope* 261
 - Strategies for binding and validation* 262
 - Making decisions programmatically with action states* 265
- 9.3 Advanced webflows 266
 - Flow-scoped services* 266
 - Subflows and conversations* 269
- 9.4 Testing webflows 273
 - Handling input parameters* 275
 - Testing subflow transitions* 276
 - Testing flow termination* 277
- 9.5 Summary and best practices 278

10 *Don't let strangers in—security* 280

- 10.1 Why security matters 280
- 10.2 Protecting against malicious intent 281
 - Validate all your inputs* 282
 - Escape all your outputs* 284
 - SSL, encryption, and message digests* 287
 - Don't give away information* 289
- 10.3 Access control 291
 - Getting started with Spring Security* 293
 - Protecting URLs* 295
 - Getting hold of the current user* 298
 - Using a custom login page* 300
 - Testing access control* 301
- 10.4 Further exploration of Spring Security 302
 - Adding user registration* 303
 - Tightening restrictions on access* 304
 - Other authentication options* 306
- 10.5 Summary and best practices 308

11 *Remote access* 310

- 11.1 Using a RESTful solution 311
 - Your first steps in REST* 311
 - Serializing and deserializing domain instances* 315
 - Testing the API* 318
- 11.2 Negotiating the representation 320
 - REST and the `params` property* 321
 - Handling multiple response formats* 323
 - How the response format is decided* 325
- 11.3 REST in practice 327
 - Keeping the API stable* 327
 - Applying the theory* 329

- 11.4 Operation-oriented remoting 333
 - The Remoting plugin* 333
 - *Comparing the remoting protocols* 335
 - *Web Services via SOAP* 336
- 11.5 Summary and best practices 338

12 Understanding messaging and scheduling 340

- 12.1 A hitchhiker's guide to messaging 341
 - Learning to think in async: what are good messaging candidates?* 342
 - *Messaging terminology: of producers, consumers, topics, and queues* 342
 - *Installing and configuring the JMS plugin* 344
- 12.2 Using the Grails JMS plugin 346
 - Our killer Hubbub feature: IM integration with Jabber* 346
 - Sending JMS messages* 347
 - *Reading the queue* 349
- 12.3 Grails scheduling 352
 - Writing a daily digest job* 352
 - *Fine-grained scheduling with cron* 354
- 12.4 Advanced scheduling 355
 - Dealing with re-entrance and stateful jobs* 356
 - *Pausing and resuming stateful jobs programmatically* 357
 - *Persistence and clustering* 360
- 12.5 Summary and best practices 361

PART 4 ADVANCED GRAILS..... 363

13 Advanced GORM kung fu 365

- 13.1 Domain model kung fu 366
 - Exploring inheritance options* 366
 - *Embedding domain classes* 368
 - *Using maps for quick and dirty (or cheap and cheerful) tables* 368
 - *Exploring domain model events* 369
- 13.2 Caching kung fu: moving from 2 users to 2¹⁰ 370
 - Hibernate settings: should you use the second-level cache?* 370
 - Cache configuration* 371
 - *Caching individual domain classes* 372
 - *Enough talk, let's profile* 374
 - *Improving performance with indexed fields* 376
 - *What about query caching?* 377
 - *JNDI? That's so old school...* 378
- 13.3 Legacy integration kung fu: dealing with multiple data sources 379

- 13.4 Dealing with difficult legacy databases 380
 - Recycling Hibernate mappings* 381
 - *Using GORM DSL to access existing database table structures* 388
- 13.5 Summary and best practices 393

14 *Spring and transactions* 395

- 14.1 Spring and Grails 396
 - A conventional approach* 397
 - *Creating and defining your own beans* 401
- 14.2 Using transactions with GORM 406
 - Easy transactions with services* 406
 - *Transactions, the session, and me* 410
 - *Fine-grained transactions* 412
- 14.3 Summary and best practices 413

15 *Beyond compile, test, and run* 415

- 15.1 The Grails build system 417
 - Packaging an application* 417
 - *Going it alone: how to create a dist command* 420
 - *Deployment* 426
- 15.2 Build integration—not for the hobbyist 430
 - Ant* 430
 - *Maven* 434
- 15.3 Coping with a changing data model 436
 - Schema migration with Hibernate* 437
 - *Intelligent migration with Autbase* 438
- 15.4 Summary and best practices 440

16 *Plugin development* 442

- 16.1 Creating your first plugin 443
 - Are you sure it's not an application?* 443
 - *Controllers, views, and other artifacts* 446
- 16.2 Publishing your plugin 450
 - Testing plugins* 451
 - *Releasing the plugin into the wild* 452
- 16.3 Integrating with Grails 454
 - Enhancing artifacts with dynamic methods* 455
 - *Dealing with class reloading* 460
 - *Leveraging Spring* 463
 - *Playing with servlets and filters* 465
 - *Augmenting the available Grails commands* 467
- 16.4 Summary and best practices 468
 - index* 471