

The Definitive Guide to Grails

Second Edition



Graeme Rocher and Jeff Brown

Apress®

The Definitive Guide to Grails, Second Edition

Copyright © 2009 by Graeme Rocher, Jeff Brown

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (paperback): 978-1-59059-995-2

ISBN-13 (electronic): 978-1-4302-0871-6

Printed and bound in the United States of America 9 8 7 6 5 4 3 2

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editors: Steve Anglin, Tom Welsh

Technical Reviewer: Guillaume Laforge

Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, Matthew Moodie, Duncan Parkes, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Project Manager: Kylie Johnston

Copy Editors: Nina Goldschlager, Kim Wimpsett

Associate Production Director: Kari Brooks-Copony

Production Editor: Laura Cheu

Compositor: Pat Christenson

Proofreader: Kim Burton

Indexer: Becky Hornyak

Artist: April Milne

Cover Designer: Kurt Krames

Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit <http://www.springeronline.com>.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit <http://www.apress.com>.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at <http://www.apress.com/info/bulksales>.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at <http://www.apress.com>.

To Birjina, the love and support you have given me in the last few years will stay with me forever. Unquestionably yours. Maite zaitut.

—Graeme Rocher

To Betsy, Jake, and Zack, the best team ever.

—Jeff Brown

Contents at a Glance

About the Authors	xix
About the Technical Reviewer	xxi
Acknowledgments	xxiii
Introduction	xxv
CHAPTER 1 The Essence of Grails	1
CHAPTER 2 Getting Started with Grails	17
CHAPTER 3 Understanding Domain Classes	45
CHAPTER 4 Understanding Controllers	65
CHAPTER 5 Understanding Views	107
CHAPTER 6 Mapping URLs	143
CHAPTER 7 Internationalization	159
CHAPTER 8 Ajax	171
CHAPTER 9 Creating Web Flows	199
CHAPTER 10 GORM	249
CHAPTER 11 Services	289
CHAPTER 12 Integrating Grails	305
CHAPTER 13 Plugins	367
CHAPTER 14 Security	407
CHAPTER 15 Web Services	449
CHAPTER 16 Leveraging Spring	487
CHAPTER 17 Legacy Integration with Hibernate	519
APPENDIX The Groovy Language	545
INDEX	571

Contents

About the Authors	xix
About the Technical Reviewer.....	xxi
Acknowledgments.....	xxiii
Introduction	xxv
CHAPTER 1 The Essence of Grails	1
Simplicity and Power	2
Grails, the Platform.....	4
Living in the Java Ecosystem	4
Getting Started	5
Creating Your First Application	7
Step 1: Creating the Application.....	7
Step 2: Creating a Controller.....	8
Step 3: Printing a Message	10
Step 4: Testing the Code	10
Step 5: Running the Tests	12
Step 6: Running the Application.....	13
Summary	15
CHAPTER 2 Getting Started with Grails.....	17
What Is Scaffolding?	17
Creating a Domain	17
Dynamic Scaffolding	19
The Create Operation	21
The Read Operation	23
The Update Operation.....	25
The Delete Operation	26
Static Scaffolding	27
Generating a Controller	27
Generating the Views	32
Being Environmentally Friendly	33

Configuring Data Sources	34
The DataSource.groovy File.....	35
Configuring a MySQL Database	37
Configuring a JNDI Data Source.....	39
Supported Databases	39
Deploying the Application	41
Deployment with run-war	41
Deployment with a WAR file	41
Summary	42
CHAPTER 3 Understanding Domain Classes	45
Persisting Fields to the Database.....	45
Validating Domain Classes	46
Using Custom Validators	49
Understanding Transient Properties	50
Customizing Your Database Mapping	51
Building Relationships	53
Extending Classes with Inheritance.....	56
Embedding Objects	59
Testing Domain Classes	60
Summary	63
CHAPTER 4 Understanding Controllers	65
Defining Controllers	65
Setting the Default Action	66
Logging	67
Logging Exceptions.....	68
Accessing Request Attributes	68
Using Controller Scopes.....	70
Understanding Flash Scope	71
Accessing Request Parameters	73
Rendering Text.....	73
Redirecting a Request	73
Creating a Model.....	75

Rendering a View	76
Finding the Default View	76
Selecting a Custom View	76
Rendering Templates	77
Performing Data Binding	77
Validating Incoming Data	78
The Errors API and Controllers	79
Data Binding to Multiple Domain Objects	80
Data Binding with the bindData Method	80
Data Binding and Associations	81
Working with Command Objects	82
Defining Command Objects	82
Using Command Objects	83
Imposing HTTP Method Restrictions	85
Implementing an Imperative Solution	85
Taking Advantage of a Declarative Syntax	85
Controller IO	86
Handling File Uploads	86
Reading the Request InputStream	89
Writing a Binary Response	89
Using Simple Interceptors	90
Before Advice	90
After Advice	91
Testing Controllers	91
Controllers in Action	93
Creating the gTunes Home Page	94
Adding the User Domain Class	95
Adding a Login Form	96
Implementing Registration	97
Testing the Registration Code	100
Allowing Users to Log In	102
Testing the Login Process	104
Summary	106

CHAPTER 5	Understanding Views	107
	The Basics	107
	Understanding the Model	108
	Page Directives	109
	Groovy Scriptlets	109
	GSP as GStrings	110
	Built-in Grails Tags	111
	Setting Variables with Tags	111
	Logical Tags	112
	Iterative Tags	113
	Filtering and Iteration	114
	Grails Dynamic Tags	116
	Linking Tags	117
	Creating Forms and Fields	119
	Validation and Error Handling	123
	Paginating Views	125
	Rendering GSP Templates	132
	Creating Custom Tags	136
	Creating a Tag Library	137
	Custom Tag Basics	138
	Testing a Custom Tag	139
	Summary	141
CHAPTER 6	Mapping URLs	143
	Understanding the Default URL Mapping	143
	Including Static Text in a URL Mapping	144
	Removing the Controller and Action Names from the URL	145
	Embedding Parameters in a Mapping	145
	Specifying Additional Parameters	147
	Mapping to a View	148
	Applying Constraints to URL Mappings	149
	Including Wildcards in a Mapping	150
	Mapping to HTTP Request Methods	151
	Mapping HTTP Response Codes	153
	Taking Advantage of Reverse URL Mapping	154
	Defining Multiple URL Mappings Classes	155
	Testing URL Mappings	155
	Summary	158

CHAPTER 7	Internationalization	159
	Localizing Messages	159
	Defining User Messages	159
	Retrieving Message Values	161
	Using URL Mappings for Internationalization	163
	Using Parameterized Messages	164
	Using <code>java.text.MessageFormat</code>	164
	Using the <code>message</code> Tag for Parameterized Messages	165
	Using Parameterized Messages for Validation	166
	Using <code>messageSource</code>	168
	Summary	170
CHAPTER 8	Ajax	171
	The Basics of Ajax	171
	Ajax in Action	173
	Changing Your Ajax Provider	174
	Asynchronous Form Submission	175
	Executing Code Before and After a Call	177
	Handling Events	178
	Fun with Ajax Remote Linking	179
	Adding Effects and Animation	193
	Ajax-Enabled Form Fields	193
	A Note on Ajax and Performance	197
	Summary	198
CHAPTER 9	Creating Web Flows	199
	Getting Started with Flows	200
	Defining a Flow	200
	Defining the Start State	200
	Defining End States	201
	Action States and View States	202
	Flow Scopes	204
	Flows, Serialization, and Flow Storage	204
	Triggering Events from the View	205
	Transition Actions and Form Validation	206
	Subflows and Conversation Scope	206

Flows in Action	208
Updating the Domain	209
Updating the View	211
Defining the Flow	212
Adding a Start State	212
Implementing the First View State	216
Data Binding and Validation in Action	218
Action States in Action	222
Reusing Actions with Closures	227
Using Command Objects with Flows	231
Dynamic Transitions	235
Verifying Flow State with Assertions	236
Testing Flows	244
Summary	247

CHAPTER 10 GORM	249
Persistence Basics	249
Reading Objects	249
Listing, Sorting, and Counting	250
Saving, Updating, and Deleting	251
Associations	252
Relationship Management Methods	253
Transitive Persistence	254
Querying	254
Dynamic Finders	255
Criteria Queries	257
Query by Example	261
HQL and SQL	261
Pagination	262
Configuring GORM	263
SQL Logging	264
Specifying a Custom Dialect	264
Other Hibernate Properties	265

The Semantics of GORM	265
The Hibernate Session	266
Session Management and Flushing	267
Obtaining the Session	268
Automatic Session Flushing	270
Transactions in GORM	272
Detached Objects	274
The Persistence Life Cycle	274
Reattaching Detached Objects	276
Merging Changes	277
Performance Tuning GORM	278
Eager vs. Lazy Associations	278
Batch Fetching	281
Caching	282
Inheritance Strategies	285
Locking Strategies	285
Events Auto Time Stamping	287
Summary	288
CHAPTER 11 Services	289
Service Basics	289
Services and Dependency Injection	291
Services in Action	291
Defining a Service	293
Using a Service	294
Transactions	295
Scoping Services	297
Testing Services	298
Exposing Services	298
Summary	304

CHAPTER 12	Integrating Grails	305
	Grails and Configuration	305
	Configuration Basics	305
	Environment-Specific Configuration	306
	Configuring Logging	306
	Stack Trace Filtering	309
	Externalized Configuration	310
	Understanding Grails' Build System	310
	Creating Gant Scripts	312
	Command-Line Variables	313
	Parsing Command-Line Arguments	314
	Documenting Your Scripts	315
	Reusing More of Grails	316
	Bootstrapping Grails from the Command Line	317
	Gant in Action	317
	Integration with Apache Ant	325
	Dependency Resolution with Ivy	327
	Code Coverage with Cobertura	330
	Continuous Integration with Hudson	331
	Adding Support to Your Favorite IDE	335
	IntelliJ	336
	NetBeans	337
	Eclipse	338
	TextMate	342
	Remote Debugging with an IDE	344
	Integration with E-mail Servers	345
	Scheduling Jobs	349
	Installing the Quartz Plugin	349
	Simple Jobs	350
	Cron Jobs	351
	Interacting with the Scheduler	354
	Scheduling Jobs	354
	Pausing and Resuming Jobs	355
	Triggering a Job	355
	Adding and Removing Jobs	355
	Jobs in Action	356

Deployment	361
Deploying with Grails	361
Deploying to a Container	361
Application Versioning and Metadata	362
Customizing the WAR	363
Populating the Database with BootStrap Classes	364
Summary	365
CHAPTER 13 Plugins	367
Plugin Basics	367
Plugin Discovery	367
Plugin Installation	369
Local Plugins	370
Creating Plugins	370
Providing Plugin Metadata	371
Supplying Application Artefacts	373
Plugin Hooks	374
Plugin Variables	375
Custom Artefact Types	376
Providing Spring Beans	379
Dynamic Spring Beans Using Conventions	382
Using Metaprogramming to Enhance Behavior	383
Plugin Events and Application Reloading	385
Modifying the Generated WAR Descriptor	388
Packaging and Distributing a Grails Plugin	389
Local Plugin Repositories	390
Plugins in Action	391
Plugins to Add Behavior	391
Plugins for Application Modularity	397
Summary	406
CHAPTER 14 Security	407
Securing Against Attacks	407
SQL or HQL Injection	407
Groovy Injection	409
Cross-Site Scripting (XSS)	409
XSS and URL Escaping	411
Denial of Service (DoS)	412
Batch Data Binding Vulnerability	413

Using Dynamic Codecs	414
Authentication and Authorization	416
Grails Filters	417
The JSecurity Plugin	419
Authentication Realms	419
Subjects and Principals	420
Roles and Permissions	421
JSecurity in Action	421
Limiting Access Through URL Mappings	446
Summary	448
CHAPTER 15 Web Services	449
REST	450
RESTful URL Mappings	450
Content Negotiation	452
Content Negotiation with the ACCEPT Header	452
The ACCEPT Header and Older Browsers	456
Content Negotiation with the CONTENT_TYPE Header	457
Content Negotiation Using File Extensions	458
Content Negotiation with a Request Parameter	459
Content Negotiation and the View	459
Marshaling Objects to XML	460
Marshaling Objects to JSON	463
Unmarshaling XML or JSON	466
REST and Security	472
Atom and RSS	473
Creating RSS and Atom Feeds	473
RSS and Atom Link Discovery	476
SOAP	478
SOAP Web Services via Plugins	479
Calling SOAP from the Client	482
Summary	485
CHAPTER 16 Leveraging Spring	487
Spring Basics	487
Spring and Grails	489
Dependency Injection and Grails	489
The BeanBuilder DSL	490

Spring in Action.....	498
Integrating JMS with Spring JMS.....	498
Mixing Groovy and Java with Spring.....	513
Summary.....	516
CHAPTER 17 Legacy Integration with Hibernate.....	519
Legacy Mapping with the ORM DSL.....	519
Changing Table and Column Name Mappings.....	520
Changing Association Mappings.....	521
Understanding Hibernate Types.....	524
Changing the Database Identity Generator.....	529
Using Composite Identifiers.....	531
Mapping with Hibernate XML.....	532
EJB 3–Compliant Mapping.....	535
Using Constraints with POJO Entities.....	541
Summary.....	543
APPENDIX The Groovy Language.....	545
Groovy and Java: A Comparison.....	545
What’s the Same?.....	546
What’s Different?.....	546
The Basics.....	547
Declaring Classes.....	548
Language-Level Assertions.....	548
Groovy Strings.....	549
Closures.....	552
Lists, Maps, and Ranges.....	553
Expando Objects.....	555
Ranges.....	556
Groovy Power Features.....	557
Everything Is an Object.....	557
Metaprogramming.....	561
Understanding Builders.....	567
Summary.....	569
INDEX.....	571

About the Authors



■ **GRAEME KEITH ROCHER** is a software engineer and head of Grails development at SpringSource (<http://www.springsource.com>), the company behind the Spring Framework that underpins Grails. In his current role, Graeme leads the ongoing development of the Grails framework, driving product strategy and innovation for the Grails framework.

Graeme started his career in the e-learning sector as part of a team developing scalable enterprise learning management systems based on Java EE technology. He later branched into the digital TV arena, where he faced increasingly complex requirements that required an agile approach as the ever-changing and young digital TV platforms evolved. This is where Graeme was first exposed to Groovy and where he began combining Groovy with Cocoon to deliver dynamic multichannel content management systems targeted at digital TV platforms.

Seeing an increasing trend for web delivery of services and the complexity it brought, Graeme embarked on another project to simplify it and founded Grails. Grails is a framework with the essence of other dynamic language frameworks but is targeted at tight Java integration. Graeme is the current project lead of Grails and is a member of the Groovy JSR-241 executive committee.

Before SpringSource, Graeme cofounded G2One Inc.—The Groovy/Grails Company—along with Guillaume Laforge (Groovy project lead) and Alex Tkachman (former JetBrains COO). G2One provided consulting, training, and support for the Groovy and Grails technologies. In October 2008, SpringSource acquired G2One, and Graeme, along with his colleagues at G2One, joined the number-one provider of enterprise software in the Java space. SpringSource now provides training, support, consulting, and products for Groovy and Grails, as well as the frameworks that underpin them such as Spring and Hibernate.



■ **JEFF BROWN** is a software engineer at SpringSource and a member of the Groovy and Grails development teams. Jeff has been involved with software engineering since the early 1990s and has designed and built systems for industries including financial, biomedical, aerospace, and others.

Jeff began his software engineering career building business systems in C and C++ targeting the Unix, OS/2, and Windows platforms. As soon as the Java language came along, he realized that it was going to be a very important technology moving forward. At this point, Jeff joined Object Computing Inc. (<http://www.ocweb.com/>) based in St.

Louis, Missouri, where he spent the next 11 years building systems for the Java platform, coaching and mentoring developers, developing and delivering training, and evangelizing.

While fully appreciating the power and flexibility offered by the Java platform, Jeff was frustrated with the unnecessary complexity often associated with Java applications. In particular, web application development with Java seemed to have a ridiculous amount of complexity that really had nothing at all to do with the real problems solved by the application. Jeff discovered the Grails framework soon after Graeme founded the project. Here were the beginnings of a solution that made so much more sense in so many ways. After digging in to the source code of the project, Jeff began making contributions and eventually became a member of the Grails development team.

Jeff eventually joined the team at G2One Inc.—The Groovy/Grails Company—where he would help drive the professional services side of the business. In late 2008, Jeff joined SpringSource when G2One and SpringSource came together to leverage synergies between the technologies created and supported by each company.

Through his entire career Jeff has always been a hands-on technologist actively involved in software development, training, and mentoring. He is also an international public speaker, having been featured regularly on the No Fluff Just Stuff Software Symposium tour (<http://www.nofluffjuststuff.com/>) for a number of years.

About the Technical Reviewer



■ **GUILLAUME LAFORGE** is the Groovy project manager and the spec lead of JSR-241, the Java specification request standardizing the Groovy dynamic language. He coauthored Manning's best-seller *Groovy in Action*.

Along with Graeme Rocher, he founded G2One Inc., the Groovy/Grails company dedicated to sustaining and leading the development of both Groovy and Grails and providing professional services (expertise, consulting, support, and training) around those technologies. In November 2008, SpringSource acquired G2One, and now Groovy and Grails bring additional weapons to the SpringSource portfolio to fight the war on enterprise Java complexity.

You can meet Guillaume at conferences around the world where he evangelizes the Groovy dynamic language, domain-specific languages in Groovy, and the agile Grails web framework.

Acknowledgments

First and foremost, I'd like to thank my wife, Birjinia, for her beauty, wisdom, and continued love and support. Over the last few years you have given me your total support and made sacrifices to the cause that I will value forever. Te quiero. Also, to my kids, Alex and Lexeia, who provide little pockets of inspiration to me every day, and to all of my and Birjinia's family, thanks for your support and encouragement.

Also, thanks to everyone at Apress that I have worked with from Steve Anglin and Tom Welsh to the people on the production team such as Nina Goldschlager and Kim Wimpsett (my copy editors), Laura Cheu (production editor), and, in particular, Kylie Johnston (project manager) for keeping the whole thing on track.

To Peter Ledbrook whose insight and contributions have been unbelievably valuable to the community and me. To Marc Palmer for providing a voice of reason, intelligent debate, and continued valuable contribution to Grails. To Alex Tkachman for his inspirational leadership at G2One and continued friendship. To the core members of the Groovy team, such as Guillaume Laforge and Jochen "blackdrag" Theodorou, whose continued responsiveness makes Grails' existence possible.

Also, without the support of the Grails community in general, we wouldn't have gotten very far. So, thanks to all the Grails users, in particular to Sven Haiges and Glen Smith for producing the Grails podcast and screencasts and to all the plugin developers who make Grails a thriving hive of activity.

Last, but most certainly not least, thanks to Rod Johnson, Adrian Coyler, Peter Cooper-Ellis, and everyone at SpringSource for seeing the potential of Grails and granting me the privilege of working for a fantastic company.

Graeme Rocher

I have to start by thanking my lovely wife, Betsy, and our unbelievable boys, Zack and Jake. Thank you all for putting up with me being closed behind the door of my home office many evenings as I worked on this book. You are all the absolute best!

Thanks to Graeme for his support as we worked through this project. It has been a lot of hard work and a whole lot of fun.

I owe a great debt to Alex Tkachman, Graeme Rocher, and Guillaume Laforge. G2One was absolutely the most exciting, challenging, and rewarding professional experience I have ever been involved with. It truly is an honor and a pleasure to know and work with you guys.

Thanks to Matt Taylor for all of the great work we have done together starting back at OCI, then G2One, and now SpringSource.

For more than a decade of professional accomplishments, I have to thank all my friends at OCI. I especially want to thank my friend Dr. Ebrahim Moshiri for the great opportunities and years of support. Thank you, sir. Also, I thank Mario Aquino. There are so many folks at OCI who

I enjoyed working with and continue to enjoy a friendship with, none of them more than Mario. Thanks for everything, man. The next one is on me.

Thanks to the whole team at Apress. I appreciate all of your hard work and patience. Thanks to Kylie Johnston (project manager) for helping us navigate through the whole thing.

I also have to thank Rod Johnson and the whole team at SpringSource. We have a lot of really exciting stuff ahead of us, and I truly look forward to it.

Jeff Brown

Introduction

In the late '90s I was working on a project developing large-scale enterprise learning management systems using early J2EE technologies such as EJB 1.0 and the Servlet framework. The Java hype machine was in full swing, and references to “EJB that, and Java this” were on the cover of every major IT publication.

Even though what we were doing—and learning as we did it—felt so horribly wrong, the industry kept telling us we were doing the right thing. EJB was going to solve all our problems, and servlets (even without a view technology at the time) were the right thing to use. My, how times have changed.

Nowadays, Java and J2EE are long-forgotten buzzwords, and the hype machine is throwing other complex acronyms at us such as SOA and ESB. In my experience, developers are on a continued mission to write less code. The monolithic J2EE specifications, like those adopted by the development community in the early days, didn't help. If a framework or a specification is overly complex and requires you to write reams of repetitive code, it should be an immediate big red flag. Why did we have to write so much repetitive boilerplate code? Surely there was a better way.

In the end, developers often influence the direction of technology more than they know. Why do so many developers favor REST over SOAP for web services? Or Hibernate over EJB for persistence? Or Spring over JNDI for Inversion of Control? In the end, simplicity often wins the day.

Certainly, working with Spring and Hibernate feels a lot better than traditional J2EE approaches; in fact, I strove to use them whenever possible, usually in combination with WebWork, and delivered a number of successful projects with this stack. Nevertheless, I still felt I had to deal with the surrounding infrastructural issues and configuration, rather than the problem at hand. After all, the more efficient I could be as a developer when doing “real” work, the more time I would have to do what should be driving every developer: spending time with loved ones and learning new and exciting technologies.

In 2003, Groovy entered the picture. I had always been fond of looser rules governing dynamic languages in certain contexts, having worked extensively with Perl, Visual Basic, and JavaScript in the past, and after quickly hacking the WebWork source code, I was able to write MVC controllers (or *actions* in WebWork lingo) with Groovy in no time.

Groovy was perfect for controllers whose sole responsibility should be to delegate to business logic implemented by a service and then display an appropriate view. I was starting to have even more time for the good things in life. Then came the storm of dynamic language-based frameworks led by Ruby on Rails.

Unfortunately, it was all a little late. Java, the community, the tools, the frameworks, and the mind share are well-embedded. The size that Java has grown to is quite staggering, and having been in the training business for many years, I see it showing no signs of slowing, contrary to popular belief. Still, Java has its problems, and I wanted to write less code. Grails was born with this goal in mind in the summer of 2005 after I, Steven Devijver, and Guillaume Laforge kicked off a discussion about its conception on the Groovy mailing list.

Fundamentally, there is nothing at all wrong with many of the specifications that form part of J2EE. They are, however, at a rather low level of abstraction. Frameworks such as Struts,

WebWork, and more recently JSF have tried to resolve this issue; however, Java and its static typing don't help. Groovy, on the other hand, allows that higher level of abstraction. Having used it for controllers, it was now time to take it to every layer—from controllers to tag libraries and from persistence to the view technology.

The APIs you can create with Groovy's metaprogramming support are amazingly simple and concise. Grails uses every single dynamic trick, at both runtime and compile time, from custom domain-specific languages to compile-time mixins, with two fundamental goals in mind: write less code and be Java friendly.

Are Groovy and Grails a replacement for Java, like other dynamic language frameworks? No, on the contrary, they're designed to work with Java. To embrace it. To have Java at their very core. Grails is Java through and through, and it allows you to pick and choose which features to implement with dynamic typing and which to entrust to the safer hands of static typing.

Grails was born from the realization that there is never only just one tool for the job. Grails is about providing an entry point for the trivial tasks, while still allowing the power and flexibility to harness the full Java platform when needed. I hope you enjoy the book as much as I have enjoyed writing it and being part of the Grails community.

—Graeme Rocher

Who This Book Is For

Grails forms just one framework that is driving the movement toward dynamic language-based frameworks. In this sense, anyone who is interested in dynamic languages, whether Perl, Ruby, or Python, will gain something from reading this book, if just to acquire insight into what the alternatives are.

If platform is not a choice and Java is the way your project is going, Grails can provide features like no other framework. In this circumstance, Grails may have the answers you are looking for. Primarily, however, this book will be of most benefit to those who know and love the Java platform—those who appreciate the Java language for all its strong points but want something better as a web framework.

Grails is providing the answers to the long search for something better in the Java world by presenting a framework that solves the common problems in an unobtrusive, elegant manner. But this does not mean that the subject matter of this book is trivial. We'll be challenging you with advanced usages of the Groovy language and real-world examples.

Furthermore, you'll be pushing the boundaries of what is possible with a dynamic language like Groovy, extending it into every tier of a typical web application from the view layer with Ajax-enabled technology to the persistence tier with rich domain models. For experienced Java developers, it should be an enlightening experience, because we'll explore features not found in Java such as closures, builders, and metaprogramming.

Through all this, however, although the subject matter and examples are advanced, the solutions are simple, and along the way you may learn a new way to approach web application development.

How This Book Is Structured

This book is divided into 17 chapters and one appendix. Unlike the first edition, coverage of Groovy is saved for the appendix. If you have no experience using Groovy, then it is recommended that you read the appendix first as the chapters themselves dive straight into Grails starting with Chapter 1, which covers the basic philosophy behind Grails.

In Chapter 2 we take you through a kick-start, demonstrating how you can quickly get productive with Grails. Then from Chapter 3 onward we delve into detailed coverage of each concept within Grails from domain classes in Chapter 3 to views in Chapter 5. By this point, you should have a good understanding of the basics.

The book will then dive straight into the nitty-gritty details of Grails in Chapter 6 with coverage of URL mappings, followed by the multilingual experience that is internationalization in Chapter 7. If you haven't had enough excitement by this point, then Chapter 8 should solve that with coverage of Grails' support for adaptive Ajax.

In Chapter 9 the book will begin to cover some of the more advanced features of Grails starting with Web Flow. In Chapter 10 you'll get a much better understanding of how GORM works, while in Chapter 11 you'll learn how to leverage declarative transactions with Grails services.

Chapter 12 goes into a lot of detail on how you can integrate Grails into your existing ecosystem; then in Chapter 13 you will get to become a Grails plugin developer as you explore the features offered by Grails' plugin system. Security is the focal point for Chapter 14, while in Chapter 15 we'll cover publishing web services with Grails.

Finally, Chapter 16 and Chapter 17 are dedicated to the more advanced topics of integrating Grails with the underlying Spring and Hibernate frameworks.

Conventions

This book uses a diverse range of languages, including HTML, XML, JavaScript, Groovy, and Java. Nonetheless, each example is introduced appropriately and appears in a fixed-width Courier font. We have also endeavored to be consistent in the use of naming conventions throughout the book, making the examples as clear as possible.

In many cases, the original source code has been reformatted to fit within the available page space, with additional line breaks and modified code indentation being common. To increase the clarity of the code, some examples omit code where it is seen as unnecessary. In cases where the code is omitted between two blocks of code, an ellipsis (...) is used to indicate where the missing code would have been.

Prerequisites

This book shows you how to install Grails; in the examples, we use the 1.1 release. As of this writing, the 1.1 release was not quite final, but by the time of publication, Grails 1.1 should be final (or nearly so). However, Grails itself is dependent on the existence of an installed Java Virtual Machine. As a minimum, you will need to install JDK 1.5 or newer for the examples in this book to work.

Installing an application server, such as Tomcat, and a database server, such as MySQL, is entirely optional, because Grails comes bundled with an embedded server and database. Nevertheless, to use Grails in production, you may at least want to set up a database server.

Downloading the Code

The code for the examples in this book is available in the Source Code section of the Apress web site at <http://www.apress.com>. Chapter-by-chapter source code is also available in the Codehaus Subversion repository at <http://svn.codehaus.org/grails/trunk/grails-samples/dgg>.

Contacting the Authors

Graeme is an active member of the open source community and welcomes any comments and/or communication. You can reach him via e-mail at graeme.rocher@gmail.com or via his blog at <http://graemerocher.blogspot.com>. You can reach Jeff via e-mail at jeff@jeffandbetsy.net or via his blog at <http://javajeff.blogspot.com>. Alternatively, you can simply pop a message on the Grails mailing lists, the details for which can be found here: <http://grails.org/Mailing+lists>.



The Essence of Grails

Simplicity is the ultimate sophistication.

—Leonardo da Vinci

To understand Grails, you first need to understand its goal: to dramatically simplify enterprise Java web development. To take web development to the next level of abstraction. To tap into what has been accessible to developers on other platforms for years. To have all this but still retain the flexibility to drop down into the underlying technologies and utilize their richness and maturity. Simply put, we Java developers want to “have our cake and eat it too.”

Have you faced the pain of dealing with multiple, crippling XML configuration files and an agonizing build system where testing a single change takes minutes instead of seconds? Grails brings back the fun of development on the Java platform, removing barriers and exposing users to APIs that enable them to focus purely on the business problem at hand. No configuration, zero overhead, immediate turnaround.

You might be wondering how you can achieve this remarkable feat. Grails embraces concepts such as Convention over Configuration (CoC), Don't Repeat Yourself (DRY), and sensible defaults that are enabled through the terse Groovy language and an array of domain-specific languages (DSLs) that make your life easier.

As a budding Grails developer, you might think you're cheating somehow, that you should be experiencing more pain. After all, you can't squash a two-hour gym workout into twenty minutes, can you? There must be payback somewhere, maybe in extra pounds?

As a developer you have the assurance that you are standing on the shoulders of giants with the technologies that underpin Grails: Spring, Hibernate, and, of course, the Java platform. Grails takes the best of dynamic language frameworks like Ruby on Rails, Django, and TurboGears and brings them to a Java Virtual Machine (JVM) near you.

Simplicity and Power

A factor that clearly sets Grails apart from its competitors is evident in the design choices made during its development. By not reinventing the wheel, and by leveraging tried and trusted frameworks such as Spring and Hibernate, Grails can deliver features that make your life easier without sacrificing robustness.

Grails is powered by some of the most popular open source technologies in their respective categories:

- *Hibernate*: The de facto standard for object-relational mapping (ORM) in the Java world
- *Spring*: The hugely popular open source Inversion of Control (IoC) container and wrapper framework for Java
- *SiteMesh*: A robust and stable layout-rendering framework
- *Jetty*: A proven, embeddable servlet container
- *HSQLDB*: A pure Java Relational Database Management System (RDBMS) implementation

The concepts of ORM and IoC might seem a little alien to some readers. ORM simply serves as a way to map objects from the object-oriented world onto tables in a relational database. ORM provides an additional abstraction above SQL, allowing developers to think about their domain model instead of getting wrapped up in reams of SQL.

IoC provides a way of “wiring” together objects so that their dependencies are available at runtime. As an example, an object that performs persistence might require access to a data source. IoC relieves the developer of the responsibility of obtaining a reference to the data source. But don’t get too wrapped up in these concepts for the moment, as their usage will become clear later in the book.

You benefit from Grails because it wraps these frameworks by introducing another layer of abstraction via the Groovy language. You, as a developer, will not know that you are building a Spring and Hibernate application. Certainly, you won’t need to touch a single line of Hibernate or Spring XML, but it is there at your fingertips if you need it. Figure 1-1 illustrates how Grails relates to these frameworks and the enterprise Java stack.

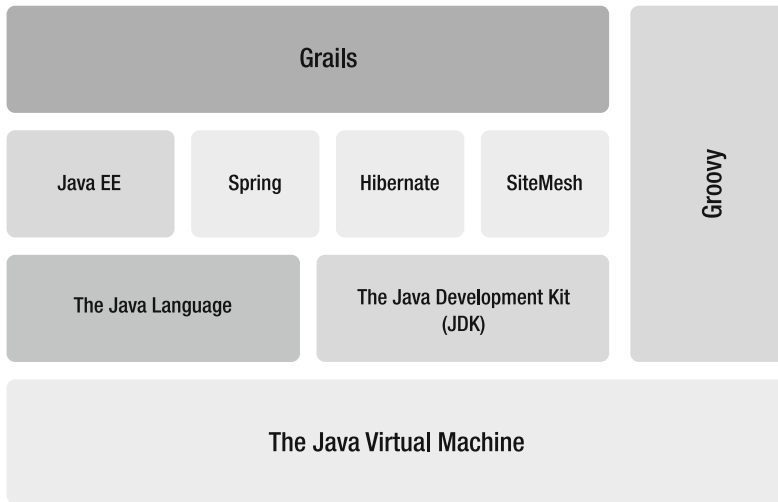


Figure 1-1. *The Grails stack*

Grails, the Platform

When approaching Grails, you might suddenly experience a deep inhalation of breath followed by an outcry of “not another web framework!?” That’s understandable, given the dozens of web frameworks that exist for Java. But Grails is different, and in a good way. Grails is a full-stack environment, not just a web framework. It is a *platform* with ambitious aims to handle everything from the view layer down to your persistence concerns.

In addition, through its plugins system (covered in Chapter 13), Grails aims to provide solutions to an extended set of problems that might not be covered out of the box. With Grails you can accomplish searching, job scheduling, enterprise messaging and remoting, and more.

The sheer breadth of Grails’ coverage might conjure up unknown horrors and nightmarish thoughts of configuration, configuration, configuration. However, even in its plugins, Grails embraces Convention over Configuration and sensible defaults to minimize the work required to get up and running.

We encourage you to think of Grails as not just another web framework, but the *platform* upon which you plan to build your next web 2.0 phenomenon.

Living in the Java Ecosystem

As well as leveraging Java frameworks that you know and love, Grails gives you a platform that allows you to take full advantage of Java and the JVM—thanks to Groovy. No other dynamic language on the JVM integrates with Java like Groovy. Groovy is designed to work seamlessly with Java at every level. Starting with syntax, the similarities continue:

- The Groovy grammar is derived from the Java 5 grammar, making most valid Java code also valid Groovy code.
- Groovy shares the same underlying APIs as Java, so your trusty javadocs are still valid!
- Groovy objects are Java objects. This has powerful implications that might not be immediately apparent. For example, a Groovy object can implement `java.io.Serializable` and be sent over Remote Method Invocation (RMI) or clustered using session-replication tools.
- Through Groovy's joint compiler you can have circular references between Groovy and Java without running into compilation issues.
- With Groovy you can easily use the same profiling tools, the same monitoring tools, and all existing and future Java technologies.

Groovy's ability to integrate seamlessly with Java, along with its Java-like syntax, is the number-one reason why so much hype was generated around its conception. Here we had a language with similar capabilities to languages such as Ruby and Smalltalk running directly in the JVM. The potential is obvious, and the ability to intermingle Java code with dynamic Groovy code is huge. In addition, Groovy allows you to mix static types and dynamic types, combining the safety of static typing with the power and flexibility to use dynamic typing where necessary.

This level of Java integration is what drives Groovy's continued popularity, particularly in the world of web applications. Across different programming platforms, varying idioms essentially express the same concept. In the Java world we have servlets, filters, tag libraries, and JavaServer Pages (JSP). Moving to a new platform requires relearning all of these concepts and their equivalent APIs or idioms—easy for some, a challenge for others. Not that learning new things is bad, but a cost is attached to knowledge gain in the real world, which can present a major stumbling block in the adoption of any new technology that deviates from the standards or conventions defined within the Java platform and the enterprise.

In addition, Java has standards for deployment, management, security, naming, and more. The goal of Grails is to create a platform with the essence of frameworks like Rails or Django or CakePHP, but one that embraces the mature environment of Java Enterprise Edition (Java EE) and its associated APIs.

Grails is, however, one of these technologies that speaks for itself: the moment you experience using it, a little light bulb will go on inside your head. So without delay, let's get moving with the example application that will flow throughout the course of this book. Whereas in this book's first edition we featured a social-bookmarking application modeled on the del.icio.us service, in this edition we'll illustrate an entirely new type of application: gTunes.

Our gTunes example will guide you through the development of a music store similar to those provided by Apple, Amazon, and Napster. An application of this nature opens up a wide variety of interesting possibilities from e-commerce to RESTful APIs and RSS or Atom feeds. We hope it will give you a broad understanding of Grails and its feature set.

Getting Started

Grails' installation is almost as simple as its usage, but you must take into account at least one prerequisite. Grails requires a valid installation of the Java SDK 1.5 or above which, of course, you can obtain from Sun Microsystems at <http://java.sun.com>.

After installing the Java SDK, set the `JAVA_HOME` environment variable to the location where you installed it and add the `JAVA_HOME/bin` directory to your `PATH` variables.

Note If you are working on Mac OS X, you already have Java installed! However, you still need to set `JAVA_HOME` in your `~/profile` file.

To test your installation, open up a command prompt and type **java -version**:

```
$java -version
```

You should see output similar to Listing 1-1.

Listing 1-1. *Running the Java Executable*

```
java version "1.5.0_13"  
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_13-b05-237)  
Java HotSpot(TM) Client VM (build 1.5.0_13-119, mixed mode, sharing)
```

As is typical with many other Java frameworks such as Apache Tomcat and Apache Ant, the installation process involves following a few simple steps. Download and unzip Grails from <http://grails.org>, create a `GRAILS_HOME` variable that points to the location where you installed Grails, and add the `GRAILS_HOME/bin` directory to your `PATH` variable.

To validate your installation, open a command window and type the command **grails**:

```
$ grails
```

If you have successfully installed Grails, the command will output the usage help shown in Listing 1-2.

Listing 1-2. *Running the Grails Executable*

```
Welcome to Grails 1.1 - http://grails.org/  
Licensed under Apache Standard License 2.0  
Grails home is set to: /Developer/grails-1.1
```

```
No script name specified. Use 'grails help' for more info or 'grails interactive' to  
enter interactive mode
```

As suggested by the output in Listing 1-2, typing **grails help** will display more usage information including a list of available commands. If more information about a particular command is needed, you can append the command name to the help command. For example, if you want to know more about the `create-app` command, simply type **grails help create-app**:

```
$ grails help create-app
```

Listing 1-3 provides an example of the typical output.

Listing 1-3. *Getting Help on a Command*

```
Usage (optionals marked with *):  
grails [environment]* create-app
```

```
grails create-app -- Creates a Grails project, including the necessary  
directory structure and common files
```

Grails' command-line interface is built on another Groovy-based project called Gant (<http://gant.codehaus.org/>), which wraps the ever-popular Apache Ant (<http://ant.apache.org/>) build system. Gant allows seamless mixing of Ant targets and Groovy code.

We'll discuss the Grails command line further in Chapter 12.