

Beginning EJB™ 3

Application Development

From Novice to Professional



Raghu R. Kodali and Jonathan Wetherbee
with Peter Zadrozny

Beginning EJB™ 3 Application Development: From Novice to Professional

Copyright © 2006 by Raghu R. Kodali and Jonathan Wetherbee with Peter Zadrozny

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 (pbk): 978-1-59059-671-5

ISBN-10 (pbk): 1-59059-671-4

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

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.

Java™ and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc., in the US and other countries. Apress, Inc. is not affiliated with Sun Microsystems, Inc., and this book was written without endorsement from Sun Microsystems, Inc.

Lead Editor: Steve Anglin

Technical Reviewer: Tom Marrs

Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Keir Thomas, Matt Wade

Senior Project Manager: Sofia Marchant

Copy Edit Manager: Nicole LeClerc

Copy Editor: Damon Larson

Assistant Production Director: Kari Brooks-Copony

Production Editor: Kelly Winkist

Compositors: Dina Quan, Gina Rexrode

Proofreader: Linda Marousek

Indexer: Julie Grady

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 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit <http://www.apress.com>.

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> in the Source Code/Download section. You will need to answer questions pertaining to this book in order to successfully download the code.

Contents

Foreword	xix
About the Authors	xxiii
About the Technical Reviewer	xxv
Acknowledgments	xxvii
CHAPTER 1 Introduction to the EJB 3 Architecture	1
An Introduction to EJB	2
What Is EJB?	2
The EJB Component Model	3
The EJB Framework	3
Core Features of EJB	3
Progression of the EJB Spec	5
EJB 3 Simplified Development Model	6
Distributed Computing Model	9
How This Book Is Organized	10
Chapter 1: Introduction to the EJB 3 Architecture	11
Chapter 2: EJB 3 Session Beans	11
Chapter 3: Entities and the Java Persistence API	11
Chapter 4: Advanced Persistence Features	11
Chapter 5: EJB 3 Message-Driven Beans	11
Chapter 6: EJB 3 and Web Services	12
Chapter 7: Integrating Session Beans, Entities, Message-Driven Beans, and Web Services	12
Chapter 8: Transaction Support in EJB 3	12
Chapter 9: EJB 3 Performance and Testing	12
Chapter 10: Migrating EJB 2.x Applications to EJB 3	12
Chapter 11: EJB 3 Deployment	12
Chapter 12: EJB 3 Client Applications	13

Getting Started	13
Prerequisites for Installing the GlassFish Application Server	14
Installing the GlassFish Application Server	15
Environment Variables Setup	17
Starting and Testing the GlassFish Installation	20
Conclusion	25
 CHAPTER 2 EJB 3 Session Beans	 27
Introduction	27
Introduction to Session Beans	27
Types of Session Beans	28
When Do You Use Session Beans?	28
Stateless Session Beans	31
The Bean Class	31
The Business Interface	32
Business Methods	35
Dependency Injection	36
Callback Methods	37
Interceptors	38
Stateful Session Beans	40
The Bean Class	40
The Business Interface	41
Business Methods	43
Callback Methods	44
Interceptors	46
Exception Handling	46
Client View for Session Beans	46
Packaging, Deploying, and Testing the Session Beans	50
Prerequisites	50
Compiling the Session Beans	51
Packaging the Session Beans	52
Deploying the Session Beans	53
Compiling the Client Programs	54
Running the Client Programs	54
Conclusion	56

CHAPTER 3	Entities and the Java Persistence API	57
An Entity Example		59
A Simple JavaBean: Customer.java		59
A Simple Entity: Customer.java		59
An Entity with Defaults Exposed: Customer.java		61
Coding Requirements		64
Example: Annotating Instance Variables		66
Example: Annotating Property Accessors		67
Declaring the Primary Key		68
Simple Primary Key		69
Composite Primary Key		70
Summary of Entity Examples		73
The Persistence Archive		73
The persistence.xml File		73
The EntityManager		74
Persistence Context		74
Acquiring an EntityManager Instance		75
Transaction Support		77
The Entity Life Cycle		77
The Life Cycle of a New Entity Instance		77
O/R Mapping		79
The @Table Annotation (Revisited)		80
The @Column Annotation (Revisited)		81
Complex Mappings		82
Entity Relationships		82
@OneToOne		82
@OneToMany and @ManyToOne		83
@ManyToMany		84
Lazy vs. Eager Field Bindings		85
Cascading Operations		86

JPQL	86
@NamedQuery and @NamedQueries	87
Binding Query Parameters	88
Dynamic Queries	88
Bulk Update and Delete Operations	89
Complex Queries	90
Forward Generation vs. Reverse Engineering	90
Forward Generation == Persistence	90
Reverse Engineering == Adaption	90
Which One Is Right for Your Project?	91
Example Application: CustomerOrderManager	91
Customer.java	91
CustomerOrder.java	93
Address.java	96
CustomerOrderManager.java	99
CustomerOrderClient.java	100
persistence.xml	101
Conclusion	102

■ CHAPTER 4 **Advanced Persistence Features** 103

Mapping Entity Inheritance Hierarchies	104
Getting Started	105
Entity Inheritance Mapping Strategies	105
Single-Table-per-Class Inheritance Hierarchy (InheritanceType.SINGLE_TABLE)	108
Common Base Table with Joined Subclass Tables (InheritanceType.JOINED)	118
Single-Table-per-Outermost Concrete Entity Class (InheritanceType.TABLE_PER_CLASS)	122
Comparison of O/R Implementation Approaches	126
Using Abstract Entities, Mapped Superclasses, and	
Non-Entity Classes in an Inheritance Hierarchy	127
Abstract Entity Class	127
Mapped Superclass (@MappedSuperclass)	128
Non-Entity Class	130
Embedded Non-Entity Class Usage (@Embeddable and @Embedded)	131

Polymorphic Relationships	133
Relationship Mapping	134
Polymorphic JPQL Queries	134
Using Native SQL Queries	135
Composite Primary Keys and Nested Foreign Keys	136
Using an Embedded Composite Key (@EmbeddedId)	136
Exposing Composite Key Class Fields Directly on the Entity Class (@IdClass)	138
Mapping Relationships That Use Composite Keys	139
Support for Optimistic Locking (@Version)	141
Support for Autogenerated Primary Key Values (@GeneratedValue) ...	142
Interceptors: Entity Callback Methods	144
Conclusion	145

■ CHAPTER 5 **EJB 3 Message-Driven Beans** 149

Introduction	149
Message-Oriented Architecture	149
What Is JMS?	150
Messaging Application Architecture	151
Using MDBs	153
When Do You Use MDBs?	153
MDB Classes	154
Configuration Properties	157
Dependency Injection in MDBs	161
Callback Methods	163
Interceptors	163
Exception Handling	165
Client View	165
Packaging, Deploying, and Testing MDBs	169
Prerequisites	169
Compiling the Session Beans and MDBs	169
Packaging the Session Beans and MDBs	170
Creating the JMS and JavaMail Resources	171
Deploying the Session Beans and MDBs	173
Compiling the Client Programs	174
Running the Client Programs	175
Conclusion	176

CHAPTER 6	EJB 3 and Web Services	177
	Introduction	177
	What Are Web Services?	177
	UDDI	178
	WSDL	179
	SOAP	183
	When Do You Use Web Services?	185
	Java EE and Web Services	185
	JAX-WS	185
	JAXB	186
	JAXR	186
	SAAJ	186
	JSR 181	186
	EJB 3 Stateless Session Beans As Web Services	186
	Developing a New Web Service	187
	Packaging, Deploying, and Testing Web Services	191
	Prerequisites	191
	Compiling the Session Bean	192
	Packaging the Session Bean	193
	Deploying the Session Bean	194
	Testing the Credit Service Using the GlassFish Console	195
	Web Service Client View	197
	Developing a Java Client That Accesses the Web Service	198
	Session Beans as Web Service Clients	202
	Conclusion	203
CHAPTER 7	Integrating Session Beans, Entities, Message-Driven Beans, and Web Services	205
	Introduction	205
	Application Overview	205
	Application Components and Services	206
	The Shopping Cart Component	206
	The Search Façade Component	207
	The Customer Façade Component	207
	The Order Processing Façade Component	207
	Persistence Services	207

The E-mail Service	207
The Credit Service	207
The Order Processing Service	208
The Wines Online Application Business Process	208
In-Depth Component/Service Walkthrough	209
Persistence Services	209
The Customer Façade Component	210
The Search Façade Component	213
The Shopping Cart Component	215
The Order Processing Façade Component	222
The Order Processing Service	229
The E-mail Service	236
The Credit Service	238
The Database Schema	238
Packaging, Deploying, and Testing the Application	239
Prerequisites	240
Deploying the Credit Service	240
Compiling the Persistence Unit	241
Packaging the Persistence Unit	242
Compiling the Business Services (Session Beans and MDBs)	243
Packaging the Business Services	244
Assembling the Application	245
Creating the Database Schema	245
Creating Data Sources, JMS Resources, and Mail Resources	245
Deploying the Application	250
A Simple Test Client for the Application	250
Running the Client Program	253
Conclusion	255

■ CHAPTER 8 **Transaction Support in EJB 3** 257

What Is a Transaction?	258
Distributed Transactions	258
The ACID Properties of a Transaction	259
The JTA	260
The Two-Phase Commit Protocol	260

Transaction Support in EJB	260
EJB Transaction Services	261
Session Bean Transactional Behavior in the Service Model	262
Container-Managed Transaction Demarcation	262
Bean-Managed Transaction Demarcation	266
Entity Transactional Behavior in the Persistence Model	267
How Entities Become Associated with a Transaction Context	267
Container-Managed vs. Application-Managed Persistence Context	268
Transaction-Scoped Persistence Context vs. Extended Persistence Context	268
JTA vs. Resource-Local EntityManager	269
A Transactional Scenario from the Wines Online Application	269
Setting Up the Examples	270
Stateless Session Beans with CMT Demarcation	271
Stateful Session Beans with BMT Demarcation and Extended Persistence Context	281
Conclusion	290

CHAPTER 9	EJB 3 Performance and Testing	291
	The Testing Methodology	292
	Performance Criteria	293
	Simulating Application Usage	295
	Defining Test Metrics	296
	The Grinder	298
	The Test Application	300
	The Performance Test	304
	The Test Environment	304
	The Test Script	305
	Setup	306
	Preliminary Tests	311
	Sample Size	313
	Calibration	313
	The Actual Test Runs	314
	Analyzing the Results	316
	Conclusion	320

CHAPTER 10	Migrating EJB 2.x Applications to EJB 3	321
	Getting Started	322
	Session Bean Migration	323
	EJB 2.1 Session Bean	323
	EJB 2.1 ejb-jar.xml File	324
	EJB 3 Session Bean Class	325
	Migrating the EJB 2.1 Session Bean Class	326
	Migrating the EJB 2.1 Session Bean Interfaces	327
	Migrating the EJB 2.1 Session Bean ejb-jar.xml Properties	328
	Migrating an EJB 2.1 Web Service Endpoint Interface	330
	Session Bean Migration Wrap-Up	331
	MDB Migration	331
	Entity Bean Migration	333
	EJB Client Migration	335
	Migrating an EJB 2.1 Session Bean Client	335
	End-to-End EJB Application Migration Example	336
	EJB 2.1 Application Source	337
	EJB 3 Application Source	348
	Migrating Our Application to Run Outside a Java EE Container	355
	EJB 3 Session Bean Class Running Outside the EJB Container	356
	EJB 3 persistence.xml File with an Outside-the-Container Persistence Unit	357
	EJB 3 Application-Managed EntityManager Client	358
	Conclusion	359
CHAPTER 11	EJB 3 Deployment	361
	A Note on Deployment Tools	362
	Overview of the Deployment Process	363
	The Provider	363
	The Assembler	363
	The Deployer	366
	Java EE Deployment Infrastructure	368
	The Java EE Server	368
	The Java EE Containers	368

Java EE Deployment Components	369
The Java EE Application	369
Java EE Module Types	369
Library Components	372
Application Servers and Platform Independence	375
Deployment Tools	376
The Deployment Plan	376
Deployment Roles	376
The Application Assembler	377
The Application Deployer	380
Assembling an EJB JAR Module	381
Naming Scope	382
Assembling a Persistence Unit	382
Naming Scope	383
Conclusion	383

■ CHAPTER 12 EJB 3 Client Applications

Application Architecture	385
JSF	390
Evolution of Java EE Web Technologies	391
JSF Architecture	393
JSF Tools and Components	395
Developing Web Applications Using JSF and EJB	396
The Login Page	398
The New Customer Registration Page	402
The Links Page	409
The Search Page	413
The Wine List Page	419
The Display Selected Wine Details Page	423
The Display Cart Items Page	428
The Notification Page	431
Packaging, Deploying, and Testing the Application	432
Prerequisites	432
Compiling and Packaging the JSF Application	433

Assembling the Wine Store Application	434
Deploying the Wine Store Application	434
Running the Wine Store Application	435
The Application Client Container	442
Conclusion	443
APPENDIX Performance: EJB 2 vs. EJB 3	445
The DTO Design Pattern	446
The Session Façade Design Pattern	456
Container-Managed Relationships	460
Conclusion	464
INDEX	465

Foreword

EJB 3 is a very important milestone for the specification. Not only is it significantly easier to use, but also for the first time (in my opinion), the specification is now built around the proven needs of the development community, standardizing existing best practices instead of being the result of design by committee. The reader of this book is most likely a developer, so I will present some historical context of how EJB came to be and why it matters today, from the perspective of developers.

I speak at conferences fairly often, and at a certain point during a talk I will ask the audience how many have used EJB. Usually, 90 percent put up their hand. Then, I ask them, “How many of you have used EJB as distributed objects—meaning, where you have a separate physical tier for your business logic and a separate tier for your presentation (servlets/JSP) tier?” Usually only 15 percent of the 90 percent will put up their hand. These results have been consistent at conferences I’ve spoken at in North America, Europe, and Japan. The result still never ceases to amaze me, since the early days of EJB forced you to apply distributed semantics on your code, which is useful on large-scale multi-physical tier projects—but in fact, most of the EJB audience was using it as a local framework for their small-to-medium-sized web apps.

Why is this so? If you look at the wider context of the times (1999 to 2003), it starts to make sense. If you look at the core values that EJB 1+ proposed, you could boil them all down to three simple categories:

- *Framework benefits:* At the time, if you were doing any kind of web or enterprise development in Java, you were living in a proprietary, confusing world; or you were using nuts-and-bolts tools like RMI and servlets, and creating your own frameworks. Couple that with the fact that during this time, most software developers were new, attracted by the dot-com boom—the industry was just waiting to be given a standard, agreed-upon way to do enterprise development. EJB provided that—a standard framework for handling transactions, security, stateful components, object persistence, and so on. Having a standard framework solved a real and present need—since at the time there was no open source movement and there were no web frameworks.
- *Distribution benefits:* EJB standardized a programming model and platform for building business logic with distributed objects. Hot on the heels of RMI and CORBA, this too was needed. If you wanted to do distributed objects, EJB was the answer.

- *Component benefits:* This is where things got nutty. In my opinion, Sun was reacting to the success of the Visual Basic (VB) component market and dreamed of having a similarly active market in the area of Java business components. I remember the early days when documentation and marketing around EJB centered on component reuse. There were even attempts at building online marketplaces for EJB components. As a result, EJB gained a lot of weight in the APIs, deployment, and package semantics in order to have EJB components' binaries run consistently well across app servers.

In particular, the Java community was really interested in standards/solutions for persistence, as there were only a few solutions at the time for object persistence—and rolling your own O/R mapper is no easy task.

So that's where we came from. Now let's step forward to the EJB 2.1 time frame, for which we'll fast forward to circa late 2003/early 2004. When giving this talk at conferences, I then proceed to where the community is going, in terms of these three value propositions (this is before EJB 3 plans were unveiled):

- *Framework:* At this point, we had open source frameworks that provided transactionality, pooling, security, and all the other good programming model benefits of EJB—and they did it much more simply and way better. Tools like XDoclet and Spring, and methodologies such as AOP, were bringing all the framework benefits of EJB to lighter-weight environments. Most notably, entity beans were almost universally criticized as being poor as an O/R mapping solution (indeed, they were designed around persisting components, not objects), and Hibernate rose to the most popular O/R mapping solution in Java, making entity beans irrelevant.
- *Distribution:* RMI was no longer the de facto standard for remoting in Java; other open source APIs/protocols now existed. Also, SOA (service-oriented architecture) principles frowned on certain distribution cases in which remote objects may have been used. EJB 2.0's introduction of the local interfaces felt like a hack, which also must have further complicated the spec in many ways.
- *Components:* Who cares? The enterprise component market is dead. When Sun was eyeing the success of the VB market, they failed to notice that that marketplace consisted mostly of UI widgets and “utility”-type components that are reusable across projects. We did not see things like payroll classes or the notion of a “user” captured as a VB component. The fact is, business logic is generally not reusable across projects, and so all the weight added to EJB was not necessary. Binary compatibility between EJB JARs was also a feature that had not really been used. It was common to simply add the EJB source to your build and package it at build time for different application servers. If building reusable components was not an objective of the EJB specification team, perhaps we would have had something closer to EJB 3 back in 2001.

Now, bringing the survey question I started off with back into perspective, notice that the majority of people were *not* using EJB for its component or distribution benefits. The majority of people were using it for its framework benefits, and those benefits were better served by lightweight open source. In 2003, I think the only real value proposition EJB brought to the table was its use as a distributed object framework for real large-scale systems—where it was originally intended to be used to begin with. Thus, as I would tell the audience at the talk, “You’ve all misused EJB.” The implication here being that the 15 percent who actually used EJB to distribute their business logic across physical tiers did not misuse it—they were already dealing with a situation in which they had decided that distribution was necessary and EJB certainly was the right technology choice for that.

So where are we today? EJB 3 has finally been rebuilt to be optimized for the 85 percent of the audience, based on best practices that had been established in the community by frameworks like Spring and Hibernate. Instead of a specification for building distributed, transactional, persistent components, we now have a specification for a powerful, easy-to-use, POJO-based framework providing transactions, security, O/R mapping, and distribution. The additions of basic interception, dependency injection, and annotation-driven configuration also bring to EJB 3 proven best practices that have become popular in the community over the last few years.

Things have gotten so much better that there will not be a new edition of my own book, *EJB Design Patterns*, which came out in 2002. For many of the patterns, there were workarounds to make EJB more usable for the 85 percent, and luckily those have all been addressed.

EJB 3 finally serves the needs of developers, and it is thus a great time to be reading this book.

Floyd Marinescu

Author, EJB Design Patterns

Cofounder and chief editor, InfoQ.com Enterprise Software Development Community

Creator, TheServerSide.com (J2EE community)

About the Authors



■ **RAGHU R. KODALI** is a consulting product manager and SOA evangelist for Oracle Fusion Middleware. A native of India, Raghu has worked in the software industry for over ten years as a developer, consultant, and presales engineer in emerging technologies. He is responsible for J2EE features, and has expertise in SOA technologies, web services, and application development frameworks. Raghu has been the lead product manager for EJB design-time features in Oracle JDeveloper since the EJB 1.1 specification. His current area of focus is evangelizing service-oriented infrastructure and implementing service-oriented applications.

Prior to his career in product management, Raghu held presales and marketing positions in Oracle Asia Pacific. Before joining Oracle, he worked as a software developer in Singapore. Raghu has spoken at many international conferences, including Oracle Open World, JavaOne, JavaZone, JAOO, Sun Technology Days, and EclipseWorld. He has also written numerous articles for many leading IT magazines, including *Java Developers Journal*, *Java Pro*, *SOA Web Services Journal*, *JavaWorld*, and *ODTUG Technical Journal*. Raghu holds a Master of Science degree in computer applications.



■ **JONATHAN WETHERBEE** is a consulting engineer and tech lead for EJB development tools on Oracle's JDeveloper IDE. He has over ten years of experience in development at Oracle, working on a variety of O/R mapping tools and holding responsibility for Oracle's core EJB toolset since EJB 1.1.

Prior to joining Oracle's development staff, Jonathan was a product manager for Oracle's CASE (computer-aided software engineering) tools. In 1999, he received a patent for his work on integrating relational databases in an object-oriented environment. Jonathan received a Bachelor of Science degree in cognitive science from Brown University.



■ **PETER ZADROZNY** brings over 20 years of experience to StrongMail Systems, where he serves as chief technology officer. Zadrozny joined StrongMail Systems from Oracle, where he was vice president and chief evangelist for Oracle Application Server. Previously, Zadrozny served as chief technologist of BEA Systems for Europe, the Middle East, and Africa—a role he held since launching WebLogic's operations in Europe in 1998. Prior to BEA, Zadrozny held executive and technical positions in many countries around the world for companies such as Macromedia, McKesson, Electronic Data Systems, Petróleos de Venezuela, and Sun Microsystems, for whom he started operations in Mexico.

Zadrozny authored *J2EE Performance Testing with BEA WebLogic Server*, coauthored *Professional J2EE Programming with BEA WebLogic Server*, and is the founding editor of *WebLogic Developer's Journal*. He has written numerous technical papers and articles, and is a frequent speaker on technology issues around the world. Zadrozny holds a degree in computer engineering from Universidad Simón Bolívar in Caracas, Venezuela.

About the Technical Reviewer

■ **TOM MARRS** is a principal architect with CIBER, where he specializes in SOA. He designs and implements mission-critical business applications using the latest SOA, J2EE, and open source technologies; he also provides architecture evaluation and developer training and mentoring services.

Tom is a coauthor of *JBoss at Work: A Practical Guide*, he speaks regularly at software conferences, and he reviews best-selling technical books for major publishers. An active participant in the local technical community, Tom recently founded the Denver JBoss Users Group (www.denverjbug.org) and has served as president of the Denver Java Users Group (www.denverjug.org).

Acknowledgments

While only few names are mentioned here, I must say that I've been lucky enough to have a strong team with representation from friends, family, colleagues at work, and special people from the publishing house. Without this swat team, we would still only be thinking about the book.

I would like to thank my friend Peter Zadrozny, who encouraged and convinced me that I could do this book, and who introduced me to Apress. Without his experience, craft, and guidance, this book would not have been possible.

Thanks to Sofia Marchant, Damon Larson, and Kelly Winkquist at Apress, for being patient and helping us through the process of putting our chapters together and making it all happen.

It is tough to work on a book when you have a day job to take care of. Thanks to my manager Roel Stalman for encouraging me to take on this book project and keeping keen interest.

I would also like to thank Floyd Marinescu for agreeing to write the foreword for this book, and for highlighting facts about EJB technology based on his experiences.

A special thanks goes to my friend Sri Rajan, who is inspiring as always. Sri helped us to work on the performance part of the book by providing the hardware and letting us use it for a few months.

To my dad, Chandra Sekhara Kodali, who has always taught me that patience and perseverance will always pay off in the long run, and who has always encouraged me to reach new heights.

To my wife, Lakshmi, and our two-year-old son, Yash, for patiently waiting for me during my long hours of writing this book.

Raghu R. Kodali

This book is borne of the efforts and insights of many people who provided both technical input and pure inspiration throughout its life. In particular, I would like to thank John Bracken and Doug Clarke for many design meetings and discussions of EJB and JPA best practices. Chris Carter and Michel Trudeau supported me on my quest, even when it took my attentions away from JDeveloper. They knew that the insights gained from researching, testing, and writing this book would surely pay dividends back to the team. Raghu Kodali has shown constant grace, patience, and wisdom during our numerous conversations and dealings with book matters. Dave Clark offered me an excellent forum to test out my knowledge and garner feedback. Steve Anglin, Sofia Marchant,

Damon Larson, Kelly Winkvist, and the excellent editorial and publication teams at Apress proved they are truly experts in their domain.

With the technical help from all of the above, writing a book on this topic that I love dearly would have been a mere marathon, if it weren't for all of the diversions of everyday life! For these welcome distractions, I have the following people to thank: David Silver provided much-needed head clearings in the middle of the grind. Our conversations, punctuated by long runs in the park, were pure soul satisfaction. Adam Beyda has given me a lifetime of insight and perspective on what really matters. Moby Coquillard's practical guidance and navigation through the psychological waters of writing a book was essential.

To my parents, Andrea and Peter Wetherbee, who have provided love and encouragement, and a constant reminder that you are my biggest fans. And Bob and Holly Spain, I could not have done it if you had not graciously taken the kids for so many weekends, and freed me up to do my research and writing.

In the end, the primary motivation for punching this thing through came from my desire to spend time again with my wife, Laurel, and our boys. This has been the busiest year we've ever been through together, and there is no doubt—we have been truly blessed with interesting times!

Jonathan Wetherbee