

*XDoclet in Action*



# *XDoclet in Action*

CRAIG WALLS  
NORMAN RICHARDS



MANNING  
Greenwich  
(74° w. long.)

For online information and ordering of this and other Manning books, go to [www.manning.com](http://www.manning.com). The publisher offers discounts on this book when ordered in quantity. For more information, please contact:

Special Sales Department  
Manning Publications Co.  
209 Bruce Park Avenue  
Greenwich, CT 06830

Fax: (203) 661-9018  
email: [orders@manning.com](mailto:orders@manning.com)

©2004 by Manning Publications Co. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in the book, and Manning Publications was aware of a trademark claim, the designations have been printed in initial caps or all caps.

⊗ Recognizing the importance of preserving what has been written, it is Manning's policy to have the books they publish printed on acid-free paper, and we exert our best efforts to that end.



Manning Publications Co.  
209 Bruce Park Avenue  
Greenwich, CT 06830

Copyeditor: Tiffany Taylor  
Typesetter: Denis Dalinnik  
Cover designer: Leslie Haines

ISBN 1-932394-05-2

Printed in the United States of America

1 2 3 4 5 6 7 8 9 10 – VHG – 07 06 05 04 03

*For my wife, Raymie*  
*C.W.*

*For Vincent*  
*N.R.*



# *brief contents*

---

<b>PART 1 THE BASICS .....</b>	<b>1</b>
1 ■ A gentle introduction to code generation	3
2 ■ Getting started with XDoclet	21
<b>PART 2 USING XDOCLET WITH ENTERPRISE JAVA.....</b>	<b>43</b>
3 ■ XDoclet and Enterprise JavaBeans	45
4 ■ XDoclet and the web-layer	83
5 ■ XDoclet and web frameworks	109
6 ■ XDoclet and application servers	135
<b>PART 3 OTHER XDOCLET APPLICATIONS.....</b>	<b>161</b>
7 ■ XDoclet and data persistence	163
8 ■ XDoclet and web services	210
9 ■ XDoclet and JMX	232
10 ■ XDoclet and mock objects	262
11 ■ XDoclet and portlets	275

**PART 4 EXTENDING XDOCLET..... 291**

- 12 ■ Custom code generation with XDoclet 293
- 13 ■ XDoclet extensions and tools 335

**APPENDIXES**

- A ■ Installing XDoclet 350
- B ■ XDoclet task/subtask quick reference 354
- C ■ XDoclet tag quick reference 382
- D ■ XDt template language tags 491
- E ■ The future of XDoclet 566

# contents

---

*foreword xvii*  
*preface xxi*  
*acknowledgments xxiii*  
*about this book xxvi*  
*about the title xxx*  
*about the cover illustration xxxi*

---

## **PART 1 THE BASICS** ..... 1

---

- 1** ***A gentle introduction to code generation*** 3
- 1.1 What is XDoclet? 4
  - 1.2 Types of code generation 5
    - Passive style: one-time code generation* 6 ■ *Active style: integrated code generation* 6
  - 1.3 Code generation input sources 7
    - Models as an input source* 8 ■ *Data files as input* 9
    - Source files as input* 10
  - 1.4 How XDoclet fits in 10
  - 1.5 Deciding when to use XDoclet 13
    - Should you generate code?* 13 ■ *Should you use a tool or build the generator yourself?* 15 ■ *Should you choose XDoclet?* 16

- 1.6 Code generation wisdom 16
  - Don't generate what you don't understand* 16
  - *Test the generated code* 17
  - *Don't be afraid to change your design to accommodate generation* 17
  - *Generate layer by layer, piece by piece* 18
  - *Keep generated files out of the code repository* 19
  - *Look for repetition* 20
- 1.7 Summary 20

## 2 **Getting started with XDoclet 21**

- 2.1 XDoclet in action 22
  - A common issue* 22
  - *Adding an XDoclet tag* 23
  - Integrating with Ant* 23
  - *Generating a professional-looking todo list* 24
- 2.2 Tasks and subtasks 25
  - XDoclet tasks* 25
  - *XDoclet subtasks* 26
- 2.3 Invoking tasks from Ant 28
  - Declaring tasks* 28
  - *Using tasks* 29
- 2.4 Tagging your code with attributes 30
  - The anatomy of an attribute* 31
- 2.5 Code generation patterns 33
  - Template basics* 34
  - *Template tags* 36
- 2.6 Customizing through merging 37
- 2.7 The big picture 39
- 2.8 Summary 41

---

## PART 2 USING XDOCLET WITH ENTERPRISE JAVA ..... 43

### 3 **XDoclet and Enterprise JavaBeans 45**

- 3.1 Building the web-log application 46
  - The web-log component model* 47
  - *Creating the EJB code generation build file* 47
- 3.2 Defining the EJBs 50
- 3.3 Adding the subtasks for the EJB application 51
  - Letting XDoclet write deployment descriptors* 52
  - *Generating home and local home interfaces* 52
  - *Generating remote and local interfaces* 53
  - *Generating utility objects* 54

- Generating concrete EJB implementation classes* 56
  - Including EJB references* 58 ▪ *Including container-managed persistent fields* 59 ▪ *Declaring relationships* 60
  - Generating value objects* 61
  - 3.4 Managing EJB security 66
    - Container-managed authorization* 66 ▪ *Bean-managed authorization* 66 ▪ *Identity propagation* 68
  - 3.5 Using query methods with entity beans 68
    - Find methods* 69 ▪ *Select methods* 69
  - 3.6 How you've benefitted from XDoclet so far 70
  - 3.7 Managing transactions 71
    - Container-managed transactions* 72 ▪ *Bean-managed transactions* 74
  - 3.8 Working with Data Access Objects 75
    - Generating DAO interfaces* 75 ▪ *Adding methods to the DAO interface* 77
  - 3.9 Working with message-driven beans 78
    - Defining message selectors* 78 ▪ *Setting an acknowledge mode* 79 ▪ *Specifying destinations* 80
    - Setting subscription durability* 81
  - 3.10 Summary 81
- 4 XDoclet and the web-layer 83**
- 4.1 Adding web-layer generation to the build file 84
  - 4.2 Working with servlets 86
    - Configuring servlets in web.xml* 88
  - 4.3 Referencing EJBs 91
  - 4.4 Configuring servlet security 94
    - Declaring security roles* 96 ▪ *Programming security in servlets* 97 ▪ *Propagating security roles* 98
  - 4.5 Working with servlet filters 99
    - Configuring filters in web.xml* 101
  - 4.6 Applying XDoclet to listeners 104
  - 4.7 Writing custom JSP tags 105
  - 4.8 Summary 108

- 5 XDoclet and web frameworks 109**
- 5.1 Merging framework servlets into web.xml 110
    - Merging ActionServlet for Struts 112* ▪ *Merging ServletDispatcher for WebWork 112*
  - 5.2 Using XDoclet with Jakarta Struts 113
    - Enabling Struts generation in the build files 114*
    - Implementing an Action 117* ▪ *Declaring the Struts Action 119* ▪ *Defining ActionForms 121*
  - 5.3 Using XDoclet with WebWork 130
    - Configuring Actions in actions.xml 130* ▪ *Configuring Actions in views.properties 133* ▪ *Documenting actions 133*
  - 5.4 Summary 133
- 6 XDoclet and application servers 135**
- 6.1 Why we need vendor-specific tasks 136
    - J2EE development roles 136* ▪ *J2EE application deployment 137* ▪ *Generating application server deployment descriptors 138*
  - 6.2 Deploying on JBoss 139
    - Deploying an EJB JAR on JBoss 140* ▪ *Specifying database schema 140* ▪ *Mapping foreign keys 142* ▪ *Handling relation tables 143* ▪ *Creating database tables 145* ▪ *Specifying physical JNDI names 146*
  - 6.3 Deploying a WAR file on JBoss 149
    - Setting a default web context 149* ▪ *Setting a security domain 150* ▪ *Setting a virtual host 150*
  - 6.4 Deploying on WebLogic 151
    - Deploying an EJB JAR on WebLogic 151* ▪ *Specifying database mapping 152* ▪ *Managing tables 153* ▪ *Using WebLogic-specific features 154* ▪ *Deploying a WAR file on WebLogic 155*
  - 6.5 Working with multiple application servers 156
  - 6.6 Working with multiple deployments 158
  - 6.7 Summary 159

---

**PART 3 OTHER XDOCLET APPLICATIONS ..... 161**

---

**7 XDoclet and data persistence 163**

## 7.1 Hibernate data 164

*Preparing the build for Hibernate 165* ▪ *Tagging classes for Hibernate 166* ▪ *Using the Hibernate classes 175*

## 7.2 Persisting data with JDO 176

*Adding JDO generation to the build 177* ▪ *Tagging classes for JDO persistence 179* ▪ *Using the JDO persistence-capable classes 183* ▪ *Working with vendor extensions 187*

## 7.3 Persisting data with Castor 192

*Adding Castor generation to the build 192* ▪ *Persisting objects using Castor JDO 193* ▪ *Using objects persisted with Castor JDO 199* ▪ *Working with Castor XML 202*

## 7.4 Summary 209

**8 XDoclet and web services 210**

## 8.1 Generating deployment descriptors for Apache

SOAP 211

*Writing simple Java web services for Apache SOAP 212*  
*Exposing EJBs as Apache SOAP web services 215* ▪ *Mapping custom types 217*

## 8.2 Generating deployment descriptors for Axis 222

*Writing simple Java web services for Axis 223* ▪ *Exposing EJBs as Axis web services 226* ▪ *Mapping custom types 228*

## 8.3 Summary 231

**9 XDoclet and JMX 232**

## 9.1 A quick JMX overview 233

## 9.2 Preparing the build for JMX generation 234

## 9.3 Generating MBean interfaces 235

## 9.4 Generating mlet files 239

*Deploying the mlet using the mlet service 243*

## 9.5 Working with MBean services in JBossMX 245

*Creating JBossMX services 246* ▪ *Generating XML for JBossMX model MBeans 249*

- 9.6 Generating MBean description classes for MX4J 254
  - Preparing the build for MX4J* 254
  - *Tagging MBeans for MX4J* 255
  - *Running the build* 256
  - *Deploying the MBean into MX4J* 258

- 9.7 Summary 261

## 10 **XDoclet and mock objects** 262

- 10.1 What are mock objects? 263
  - Knowing when to mock* 264
  - *Testing from the inside out* 265
- 10.2 Generating mock objects with XDoclet 266
  - Adding mock-object generation to the build* 266
  - Tagging interfaces to generate mock implementations* 268
  - Testing FullServiceStation.java with mock objects* 269
- 10.3 Summary 273

## 11 **XDoclet and portlets** 275

- 11.1 Introducing JSR-168 (the portlet API) 276
  - Writing a simple portlet* 277
  - *Deploying a portlet* 278
- 11.2 Adding portlet.xml generation to the build file 279
- 11.3 Writing a portlet 280
  - Defining portlet basics* 282
  - *Initializing portlets* 283
  - Supporting multiple display options* 284
  - *Defining preferences* 285
  - *Validating preferences* 286
- 11.4 Running the build 288
- 11.5 Summary 290

---

## PART 4 EXTENDING XDOCLET..... 291

## 12 **Custom code generation with XDoclet** 293

- 12.1 When should you bother with custom code generation? 294
  - The risks of custom generation* 295
  - *The rewards of custom generation* 295
  - *Making the leap* 296
- 12.2 Using XDoclet templates 297
  - Using aggregate generation* 297
  - *Using transformation generation* 299

- 12.3 Exploring design alternatives 304
  - Registering commands* 305
  - Generating the command processor* 306
  - Generating a configuration file* 310
  - Choosing a generation method* 312
- 12.4 Template tag concepts 313
  - Block and content tags* 313
  - Tag namespaces* 314
  - Types of tags* 315
  - Using some basic template tags* 318
- 12.5 Creating custom template tags 322
  - Creating a content tag* 324
  - Creating a body tag* 326
  - Refactoring common functionality into a tag* 328
- 12.6 Creating custom tasks 329
  - Creating the Ant task* 331
  - Creating the subtask* 331
  - Distributing custom tasks* 333
  - Registering custom tasks in Ant* 334
- 12.7 Summary 334

## 13 **XDoclet extensions and tools 335**

- 13.1 The role of tools in XDoclet 336
- 13.2 IntelliJ IDEA 337
  - Helping IDEA find your generated classes* 337
  - Configuring IDEA to accept XDoclet tags* 338
  - Using IDEA's live templates to generate XDoclet tags* 340
- 13.3 Eclipse 342
  - Using JBoss IDE* 343
  - Generating an XDoclet build file* 344
- 13.4 AndroMDA 346
- 13.5 Middlegen 347
- 13.6 Summary 349

***appendix A: Installing XDoclet 350***

***appendix B: XDoclet task/subtask quick reference 354***

***appendix C: XDoclet tag quick reference 382***

***appendix D: XDt template language tags 491***

***appendix E: The future of XDoclet 566***

***recommended reading 575***

***index 577***



## *foreword*

---

Let's face it: We're all lazy. At least, most of us are. Some of us have come up with rationalizations for this laziness, such as, "I get more done if I do less," or "By reducing complexity, I can build more complex systems," or the ever-popular "I just want to get the job done so I can go home." No matter which rationalization you choose, the concept of code generation and using metadata is probably appealing to you—if you're a programmer.

The story of XDoclet began a couple of years ago when I was involved in building the EJB container for the JBoss application server. Being the lazy programmer that I am, I found it a burden to write all those interfaces and XML descriptors related to EJB components. The editing slowed me down and made it difficult to manage a large application. But what really ticked me off was how punishing the whole process became when I wanted to change something! A simple change to a method meant editing many source files, as well as the XML descriptor containing the metadata for that method. There had to be a better way!

And there was. *JavaWorld*, which at the time was the leading online Java magazine, published an article about how to use the javadoc API to write custom doclets in order to create custom HTML documentation. I thought "Hey, that's a neat idea, but I'm not really interested in writing HTML: I want it to generate code!" A little tinkering told me this was a great approach for solving most of my problems. I could create custom javadoc tags and run the javadoc tool on my sources, and the doclet I wrote generated both source code and

XML instead of HTML. Now all information about a component was centralized in a single file: the EJB bean code, from which I could generate everything else. Neat! I called the tool EJBDoclet and published it on my homepage using an open-source license. I knew other lazy programmers would be interested.

It is now a couple of years later, and XDoclet has evolved from its initial specialized EJBDoclet incarnation to a highly customizable generic tool that can help generate almost anything from your source code. It's also used for, among others, servlets, tag libraries, as well as WebWork and Struts actions, and it supports a number of server-specific XML descriptors for EJB. It has better documentation, a fairly large user base, and a thriving community built around it. There are even tools whose purpose is to create source code that contains XDoclet tags. In short, XDoclet has become a standard tool that can help most programmers become more productive.

If we take a step back and look at the bigger picture, we see that XDoclet combines two main points: code generation and code metadata. The last couple of years have seen an explosion of new standards, each of which mandates that a particular API be exposed by the components which implement it. Design patterns have also become more popular. These two factors provide good reasons for code generation because it is easy to create templates for both. If all you have to do to expose a standard API or use a particular design pattern is create source code that contains all the base information, and then use a code generation tool like XDoclet, which creates the artifacts required by those standards and design patterns, then it becomes significantly easier to perform those tasks. One of the main problems in software engineering is knowledge—actually, lack of knowledge. Both design patterns and standard APIs require that programmers know them well in order to create code that implements them. By using code generation based on templates, such as with XDoclet, it becomes easier to capture and reuse such knowledge, making it available to programmers who lack these skills.

XDoclet is great out of the box, but what's even more interesting is the support for custom modules that you can add to it. This lets you capture your own framework and design pattern standards, which is very helpful in a large company or team.

XDoclet is an open-source project with an active and responsive community of users and developers. You should feel free to contribute your ideas and code to it. As long as new standards, products, and best practices emerge, XDoclet will need to keep evolving in order to help developers where it matters the most. Our opportunity to be lazy depends on other developers' willingness to be creative.

The book you are now reading has been written by two developers who clearly understand all of the above ideas and principles. Through their concise writing they will show you how to apply XDoclet as efficiently as possible in your own projects, and will help you understand everything from the basics of XDoclet to how to expand it with your own plugins. No matter whether you choose to use XDoclet as a lazy developer or as a creative plugin writer, this book is your essential guide.

—Rickard Öberg  
*Creator of XDoclet*



## *preface*

---

When I first heard of XDoclet (then it was called *EJBDoclet*), I thought the whole idea was nonsense. The comment blocks in my Java code were for documentation, not for programming. Why would I ever put anything in a comment block that impacted the functionality of my program? How absurd! Besides, at the time I wasn't doing much with EJBs, so what use did I have for XDoclet?

But XDoclet wouldn't leave me alone. It kept crossing my path, with its name mentioned in web-logs, presentations, and mailing lists. I was being haunted by XDoclet.

Finally I gave in to those ghosts and gave XDoclet another look. And I'm glad I did. I found out that XDoclet was for more than just EJBs and that it addressed many of the code maintenance headaches I dealt with every day. It freed me from Deployment Descriptor Hell.

I eventually got past my hang-up with putting code in comment blocks. After all, javadoc comments aren't the real documentation—they're merely metadata used to generate the real documentation. In that light, metadata used to generate deployment descriptors and interfaces is just as appropriate in comment blocks as javadoc documentation.

Newly enlightened, I dove in head first to learn as much as possible about XDoclet. I sought out every book and every article I could find. But in my quest for XDoclet knowledge, I came up short. Unfortunately, very little had been written about XDoclet. Even XDoclet's own documentation was sparse (and, in some cases, inaccurate).

I decided to remedy that problem by writing an article documenting everything I learned about XDoclet. It wasn't long before I was commissioned by a prominent Java development magazine to write an article on XDoclet.

I shared my excitement over this new writing opportunity with my friend, Norman Richards. Much to my surprise, he informed me that he had just been commissioned to write an XDoclet article for the same magazine. Upon comparing notes, we learned that our articles<sup>1</sup> covered two sides of the same XDoclet coin. Moreover, we discovered that between the two of us, we probably had enough content to produce an entire book on XDoclet.

Several months and many sleepless nights later, the book you're now reading is the book we were looking for when we started learning XDoclet. We wrote this book for ourselves. We'll refer to it often as we get stuck working with XDoclet. Even as we were writing it, we were wishing we had a book like this to guide us.

We also wrote this book for you. Perhaps you've heard of XDoclet and wondered what it's all about and how you can apply it in your projects. Maybe you're a skeptic, and you're looking for evidence that XDoclet is as useful as is claimed. Or maybe your projects are already successfully employing XDoclet and you just need a reference to this wonderful tool. Whatever the case, this book is for you.

—Craig Walls

---

<sup>1</sup> Due to various reasons, including a huge book-writing project, neither of our articles were ever published.

## *acknowledgments*

---

This book is more than a collection of words penned by two authors. In addition to those whose names are on the cover, there are many others who played very important roles and deserve credit for this book's existence.

First and foremost, we'd like to acknowledge the fine group of people we've worked with at Manning Publications. The professionalism of each and every one of you has made this project a true pleasure. Many thanks to Marjan Bace for believing in this project and giving us this awesome opportunity. And to everyone else we've worked with at Manning: Susan Capparelle, Denis Dalinik, Lee Fitzpatrick, Leslie Haimes, Ted Kennedy (in memoriam), Mary Piergies, David Roberson, Iain Shigeoka, Marilyn Smith, Tiffany Taylor, and Helen Trimes.

We'd also like to acknowledge the reviewers who gave us the criticism we needed to shape the book: Dan Berezki, Ryan Breidenbach, Daniel Brookshier, Kevin Curley, Ryan Daigle, Jeff Duska, Nathan Egge, Erik Hatcher, Jack Herrington, Ernest Hill, David Loeffler, Rickard Öberg, David Paine, Ben Sullins, and Michael Yuan.

A huge high-five to everyone in the XDoclet community for continuing to make XDoclet such a great tool. So many people have contributed in one way or another that it would be difficult to list them all here. But, to everyone who has written a module, addressed an issue in JIRA, or just answered a question on the mailing lists, our appreciation for everything you do.

Finally, we'd like to give special thanks to Rickard Öberg for his vision, for creating XDoclet in the first place, and for contributing to our book with his feedback and foreword.

CRAIG WALLS I wish to especially thank my beautiful and loving wife, Raymie, for her encouragement and patience during this very long project. I can't believe how fortunate I am to have you in my life. I love you more than you can possibly know or than I can possibly express. Can you believe that I'm *finally* finished with this thing?

I extend my gratitude to Norm for being such a great co-author. There's no way I could've written all of this myself.

Much appreciation to Erik Hatcher for convincing me early on that I could do this and for answering tons of e-mails and questions along the way.

Many thanks to my team at Michaels: Ryan Breidenbach, Marianna Krupin, Van Panyanouvong, and Tonji Zimmerman. I couldn't imagine working with a more talented bunch of developers. You continue to challenge me every day.

Head scratches and belly rubs are owed to the furry and feathered friends who surrounded me during many of my writing sessions: Buster, Max, Frasier, Dodger, Caesar, Hamlet, Echo, Squit, Scuttle, Faith, Cricket, Othello, and especially Juliette, who watched over my shoulder while I banged away at the keyboard.

I wish to express my gratitude to my parents, who instilled in me a desire to learn and who got me started tinkering with computers way back when they bought me that Commodore VIC-20. (Fortunately, my programming activities have advanced beyond BASIC!)

Finally, I'd like to acknowledge a mixed group of people who have inspired me in one way or another throughout my life. Most of you probably don't even know it, but you've had a profound impact on my life: James Bell, Frank Cavallito, Bob Drummond, Jamie Duke, Robert Gleaton, Carolyn Gunn, Gary and Pat Henderson, Brad Lartigue, Hue McCoy, and Hubert Smith.

NORMAN RICHARDS would like to thank...

Vincent, for not complaining too much when your dad's idea of quality time together for the last six months has been taking the laptop to Chuck E. Cheese's and working on the book while you played. I hope I can give you all the opportunities my parents gave me.

Michael Yuan, for showing me that writing a book wasn't such a far-out dream. I hope Enterprise J2ME does well.

Julie, for hope, dreams, and encouragement. 28-34-13-9-21-11-4. I hope you finally win.

All my friends, who shared my excitement about the book: Vedran, Pyoung Gyu, Sunny, Cinderella, Eriko, Thebes, Jessica, Kevlyn, and Devin.

Greg Lavender, for giving me my first chance; and Risto Miikkulainen, for helping me believe I had something valuable to say.

XML Austin, AustinJUG, and Capital Macintosh users groups, for making Austin such a wonderful place for techies. Thanks to the local nerds who inspire me, particularly Stu, Sam, Eitan, Damon, and Brent.

The core dump, for not picking on me about my inflated ego while I was in the room (that's what AIM is for). May you always know the joy of the ServiceLocator.

Schlotzsky's, Texpresso, Mozart's, and the Alamo Drafthouse, for all the free wireless I consumed while working on the book.

Chango's, for the Maximo Burrito. If you had wireless, I'd live there.

Xi, for the 999 chain. Alea Jacta Est!

Megatokyo, Homestar Runner, and Mr. Sinus, for reminding me to laugh every now and then.

Manning Publications, for giving me this page to ramble semi-coherently on.

## *about this book*

---

*XDoclet in Action* is a complete resource for applying XDoclet code generation to your Java development process. XDoclet spans a wide range of Java technologies, and we'll show practical examples of XDoclet with each of the major technologies. Throughout the book, we'll build a simple web application that demonstrates how you can use XDoclet across multiple subsystems to build complete applications.

Our goal in each chapter is not to teach the technology we're looking at but to demonstrate how you can apply XDoclet code generation to that technology. However, you don't need to be an expert in any of the technologies. For example, you don't need to be an Enterprise JavaBeans guru to tackle the EJB chapter, but we'll assume some basic familiarity with EJB concepts. We've tried not to assume more than is necessary to understand how to apply XDoclet, and we've provided pointers to learning resources for the technologies we touch if you want to explore them further.

### ***How to read this book***

---

Whether you're completely new to XDoclet or you're an experienced XDoclet user, we recommend reading part 1. Regardless of which technologies you'll be using XDoclet with, the first section provides valuable background information that will help you better understand code generation with XDoclet.

XDoclet generates code for a wide range of domains. We don't expect that you'll use every type of code generation that XDoclet provides. The best place to start using XDoclet is with the systems you are already working with. The chapters can be used independently and in any order. Many of the chapters build on the web-log application we develop in this book. Our goal is simply to show how XDoclet can be used across multiple technologies, not to suggest that you should design an application using all of them. You can pick and choose among the various applications of XDoclet that are relevant to your own development needs.

Experienced XDoclet users will find value in part 4, which covers custom code generation and higher-level tools. The appendixes also provide the most complete XDoclet reference material available for the more advanced or obscure XDoclet options.

### ***Part 1: The basics***

Chapter 1 presents the basic concepts of code generation and shows where XDoclet fits in the bigger picture of code generation.

Chapter 2 gets you started with XDoclet and introduces the core XDoclet concepts you'll see throughout the book.

### ***Part 2: Using XDoclet with Enterprise Java***

Chapter 3 starts our look at J2EE with the premier application of XDoclet, generating code for EJBs. This chapter shows how XDoclet lets you generate a bean's interfaces, helper classes, and deployment information directly from the bean class.

Chapter 4 moves to the web-layer, showing XDoclet generation for servlets, filters, tag libraries, and listeners. This chapter explains how XDoclet can be applied to all the core Java web technologies.

Chapter 5 explores code generation for Struts and WebWork, two popular web frameworks.

Chapter 6 completes our look at enterprise technologies by showing how XDoclet can help manage the deployment descriptors you need to deploy your application in a J2EE application server. We look at deploying applications on both JBoss and WebLogic

### ***Part 3: Other XDoclet applications***

Chapter 7 examines data persistence frameworks. You'll see how XDoclet can be used with Hibernate, JDO, and Castor.

Chapter 8 jumps to web services. This chapter shows how to use XDoclet with Apache SOAP and its successor, Axis.

Chapter 9 introduces XDoclet code generation for JMX, including MBean interfaces and mlets. This chapter also explains how to generate JBoss-specific JMX deployment files.

Chapter 10 shows you how to use XDoclet to generate mock objects to help you unit-test your code.

Chapter 11 rounds out our look at the standard XDoclet generation tasks by showing how you can use XDoclet when developing portlet applications.

#### **Part 4: Extending XDoclet**

Chapter 12 digs deep into the internals of XDoclet and demonstrates how to extend XDoclet with custom templates and tasks. If you're thinking about generating code on your own, this chapter will show you how XDoclet can help.

Chapter 13 shows how to configure the popular Eclipse and IntelliJ IDEs for use with XDoclet. This chapter also shows higher-level code generation environments like AndroMDA and Middlegen, which use XDoclet code generation.

#### **Appendixes**

Appendix A details how to install XDoclet, including some common pitfalls that you may encounter along the way.

Appendixes B, C, and D give the most complete reference to XDoclet's tasks, subtasks, merge points, metadata tags, and template tags anywhere.

Appendix E looks at the future of XDoclet and tells you how you can help shape that future by getting involved in the XDoclet project.

#### **Source code**

---

The source code for the examples in this book is available for download at <http://www.manning.com/walls>.

#### **Typographical conventions**

---

The following conventions are used throughout the book:

- *Italic* typeface is used to introduce new terms.
- **Bold type** indicates text that you should enter.
- `Courier` typeface is used to denote code samples, as well as elements and attributes, method names, classes, interfaces, and other identifiers.

- Bold face **Courier** identifies sections of code that differ from previous, similar code sections.
- Code annotations accompany many segments of code. Certain annotations are marked with bullets such as ❶. These annotations have further explanations that follow the code.
- The ® symbol is used to indicate menu items that should be selected in sequence.
- Code line continuations use the ↵ symbol.

### **Author online**

---

Purchase of *XDoclet in Action* includes free access to a private web forum run by Manning Publications where you can make comments about the book, ask technical questions, and receive help from the author and from other users. To access the forum and subscribe to it, point your web browser to <http://www.manning.com/walls>. This page provides information on how to get on the forum once you are registered, what kind of help is available, and the rules of conduct on the forum.

Manning's commitment to our readers is to provide a venue where a meaningful dialog between individual readers and between readers and the authors can take place. It is not a commitment to any specific amount of participation on the part of the authors, whose contribution to the AO remains voluntary (and unpaid). We suggest you try asking the authors some challenging questions lest their interest stray!

The Author Online forum and the archives of previous discussions will be accessible from the publisher's web site as long as the book is in print.

### **About the authors**

---

**Craig Walls**, an XDoclet project committer, has been a software developer since 1994 and a Java fanatic since 1996. He lives in Dallas, Texas.

**Norman Richards** has developed software for a decade and has been working with code generation techniques for much of that time. He is an avid XDoclet user and evangelist. Norman lives in Austin, Texas.

## *about the title*

---

Manning’s *in Action* books combine an overview with how-to examples to encourage learning *and* remembering. Cognitive science tells us that we remember best through discovery and exploration. At Manning, we think of exploration as “playing.” Every time computer scientists build a new application, we believe they play with new concepts and new techniques—to see if they can make the next program better than the one before. An essential element of an *in Action* book is that it is example-driven. *In Action* books encourage the reader to play with new code and explore new ideas. At Manning, we are convinced that permanent learning comes through exploring, playing, and most importantly, *sharing* what we have discovered with others. People learn best *in action*.

There is another, more mundane, reason for the title of this book: Our readers are busy. They use books to do a job or solve a problem. They need books that allow them to jump in and jump out easily—books that will help them *in action*. The books in this series are designed for these “impatient” readers. You can start reading an *in Action* book at any point, to learn just what you need just when you need it.

## *about the cover illustration*

---

The figure on the cover of *XDoclet in Action* is an “Aldeano de la China que recoge la basura para los campus,” a Chinese villager who collects odds and ends, a more discreet way of saying that he was the trash collector in his village. Perhaps the young man with the basket over his shoulder was a tinker or scavenger—looking to make a living from what others had discarded—rather than a man designated by the village to keep it clean. We do not know enough about village customs in China two hundred years ago to come up with a more accurate description of his function.

The illustration is taken from a Spanish compendium of regional dress customs first published in Madrid in 1799. The book’s title page states:

*Coleccion general de los Trages que usan actualmente todas las Naciones del Mundo descubierta, dibujados y grabados con la mayor exactitud por R.M.V.A.R. Obra muy util y en especial para los que tienen la del viajero universal.*

which we translate, as literally as possible, thus:

*General collection of costumes currently used in the nations of the known world, designed and printed with great exactitude by R.M.V.A.R. This work is very useful especially for those who hold themselves to be universal travelers*

Although nothing is known of the designers, engravers, and workers who colored this illustration by hand, the “exactitude” of their execution is evident in

this drawing. The “Aldeano de la China que recoge la basura para los campus” is just one of many figures in this colorful collection. Their diversity speaks vividly of the uniqueness and individuality of the world’s towns and regions just 200 years ago. This was a time when the dress codes of two regions separated by a few dozen miles identified people uniquely as belonging to one or the other. The collection brings to life a sense of isolation and distance of that period and of every other historic period except our own hyperkinetic present.

Dress codes have changed since then and the diversity by region, so rich at the time, has faded away. It is now often hard to tell the inhabitant of one continent from another. Perhaps, trying to view it optimistically, we have traded a cultural and visual diversity for a more varied personal life. Or a more varied and interesting intellectual and technical life.

We at Manning celebrate the inventiveness, the initiative, and, yes, the fun of the computer business with book covers based on the rich diversity of regional life of two centuries ago brought back to life by the pictures from this collection.