Effective Enterprise Java

Effective SOFTWARE DEVELOPMENT SERIES

Scott Meyers, Consulting Editor

The **Effective Software Development Series** provides expert advice on all aspects of modern software development. Books in the series are well written, technically sound, of lasting value, and tractable length. Each describes the critical things the experts almost always do—or almost always avoid doing—to produce outstanding software.

Scott Meyers (author of the *Effective C++* books and CD) conceived of the series and acts as its consulting editor. Authors in the series work with Meyers and with Addison-Wesley Professional's editorial staff to create essential reading for software developers of every stripe.

TITLES IN THE SERIES

Elliotte Rusty Harold, *Effective XML: 50 Specific Ways to Improve Your XML* 0321150406

Diomidis Spinellis, *Code Reading: The Open Source Perspective* 0201799405

For more information on books in this series please see www.awprofessional.com/esds

Effective Enterprise Java

Ted Neward

Addison-Wesley

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data

Neward, Ted.
Effective Enterprise Java / Ted Neward.
p. cm.
ISBN 0-321-13000-6 (pbk. : alk. paper)
1. Java (Computer program language) I. Title.

QA76.73.J38N48 2004
005.13'3—dc22 2004012164

ISBN 0-321-13000-6
Text printed on recycled paper
1 2 3 4 5 6 7 8 9 10—CRS—0807060504
First printing, August 2004

Contents

Foreword

Designing and implementing large-scale enterprise systems is hard. Building effective enterprise Java deployments is even harder. I see these difficulties on a daily basis. When consulting on enterprise projects, I see the real-world issues that developers are facing. I have also seen discussions, frustrations, and solutions to some of the issues on a daily basis on TheServerSide.com (Your Enterprise Java Community). TheServerSide.com really grew from the needs of developers faced with the new world of J2EE. It was the water cooler that allowed us to chat about solutions that worked for us, and it saw the growth of enterprise Java patterns.

Developing for the enterprise is a very different beast when compared to building smaller, standalone applications. We have to consider issues that we can safely ignore in the other world. As soon as we have to *share data* among multiple users, we start down the enterprise path. Then we start facing questions: What is the best solution for allowing concurrency to this data? How coherent and correct does it have to be? How can we scale up from 2 to 50 to 1,000 clients? These are all significant questions, and I don't feel that the average developer has enough help in answering them. Well, simply answering the questions may not be the correct focus. We need to be taught about the various issues involved and shown techniques that can help with the various problems. With Ted Neward's book, we are now armed with the knowledge that will allow us to come up with the right balance in the solution for each particular problem.

No book has attacked these problems quite like *Effective Enterprise Java* does. The most important part of this book is that it teaches you two things really well.

You will understand the general issues of enterprise computing.

These enterprise problems are far from new. Ted has been around the block, and he understands the core issues at work. A non-Java developer would get a lot out of this book for this very reason. What you learn here

will be with you for as long as you develop enterprise solutions. The language and APIs may change, but you will understand the issues in building a good architecture, the options you have for communication, the choices for where to store state, the various security concerns, and so much more.

You will be able to attack the problems by using enterprise Java.

Although the book offers genuine insight into the general enterprise problems, it also gives you tools to solve them with enterprise Java today. You will understand more about where the various enterprise Java technologies fit together. When would you use Web Services? What can messaging do for you? What is EJB good for? This book provides answers to these questions.

It is great to have some answers to these common questions. The style of the book, in which you are given a set of "effective items," gets right to the point. Get stuck in, and enjoy the ride!

Dion Almaer
Editor-in-Chief, TheServerSide.com

Preface

Those who cannot remember the past are doomed to repeat it.

—George Santayana

These are heady days for Java programmers. Commercially available for less than a decade, Java has nevertheless emerged as the language of choice for enterprise systems on nearly all major computing platforms. Companies and individuals with challenging programming problems increasingly embrace the language and platform, and the question faced by those who do not use Java is often *when* they will start, not *if*. The breadth and scope of the specifications and libraries defined by and under the Java 2 Enterprise Edition Specification—which both dwarfs and subsumes that of the Java language itself—makes it possible to write rich, complex systems without sacrificing performance or implementing common algorithms or data structures from scratch. The Java language and virtual machines grow ever more powerful. Tools and environments for Java development grow ever more abundant and robust. Commercial libraries all but obviate the need to write code in many application areas.

Scott Meyers wrote much the same in his opening to *More Effective C++* [Meyers97] almost a decade ago. It seems a fitting tribute to use that paragraph, suitably modified, as the opening in this book. As a matter of fact, the two paragraphs are deliberately side-by-side similar. In many ways, we now find ourselves in a part of the Golden Age of Java, looking out over a landscape that stretches from horizon to horizon, with ample space and established borders that seem to have no limit. Just as C++ ruled the landscape in 1996, Java rules the landscape in 2004.

The chief aim in drawing these parallels is to recognize the scenario—not more than two years after Scott wrote that paragraph, C++ was toppled from its throne by this upstart named Java. Just as C++ developers had

finally begun to "figure everything out," this new language and environment leapt forward into the fray, and almost overnight, it seemed, took over. In turn, Java now faces fierce competition from Microsoft's .NET platform. The natural concern, then, is to see that history doesn't repeat itself. To do that, Java developers must make sure that the systems they develop meet or exceed expectations set for them. To do *that,* Java developers need to know how to make the most of the language and platform they use.

It has been said, in many places by many people, that it takes about five years to "figure out" a technology and how best to use it. Certainly this was true for C++: in 1990, we looked at C++ as simply another object-oriented language, and therefore using it should mirror the best-usage practices discovered with Smalltalk. By 1995, we were well out of that world and starting to explore the uniqueness that C++ provides (such as templates and the STL). Certainly this was also true of HTTP: in 1995, when the browser debuted, we looked at HTTP as the means by which HTML was delivered. Now, we look at HTTP as a universal transport by which to transmit all sorts of data.

Thus, the timing is fortuitous for Java. It officially debuted in 1995; for all intents and purposes, however, Java truly entered the mindscape of the average developer in 1997 or so, having by that point built enough of a "critical mass" to win its way past the critics and skeptics. Almost a decade later, we have been writing Java applications for most of that time, and we're starting to see the practices and patterns that have emerged to assist (but not necessarily guarantee) successful deployments. As a community, we're just starting to hit our stride.

Some things aren't different from the C++ days. We have the same questions, modified for the Java world, as those we asked (and Scott answered) a decade ago about C++. As the language and platform have matured and our experience with Java has increased, our needs for information have changed. In 1996, people wanted to know *what* Java was. "It has something to do with the Internet, whatever that is" was a common explanation. Initially, developers focused on using applets, making rich browser clients, and harnessing Java's cross-platform portability. By 1998, they wanted to know *how* to make it work: "How do I access a relational database? How do I internationalize? How do I reach across physical machine boundaries?" Now, Java programmers ask higher-level questions: "How can I design my enterprise systems so they will adapt to future demands?

How can I improve the efficiency of my code without compromising its correctness or making it harder to use? How can I implement sophisticated functionality not directly supported by the language or platform?"

As if this weren't enough, a new dimension has arisen in the whole enterprise system arena, neatly captured in two words: Web Services. Even as Java developers are asking the hard higher-level questions about Java, they face the start of the cycle with Web Services—what is a Web Service, how does it work, and, perhaps most importantly, how does it relate to Java?

In this book, I answer these questions and many like them.

About the items

One thing I feel compelled to point out, before we get too deeply into it all, is that readers may notice a significant difference between the items in this book and those from books like *Effective Java* [Bloch] and *Effective C++* [Meyers95]. In particular, the scope of the items in this book is much larger than that in other similar books—in here, there's less focus on language and/or APIs and more on design-level constructs and/or usage patterns.

This is not an accident; in fact, I believe that this is in keeping with the larger scope of an enterprise application as a whole. Certainly, without question, all of *Effective Java* also applies to building enterprise applications, but to simply remain at that level misses the larger point, that enterprise systems have much more to worry about, things outside of the scope of the language or APIs.

For example, an unsuccessful EJB application usually begins not with misuse of a particular method call or interface but with the design of entity beans that are called directly from a client. This isn't so much an implementation problem as a design problem, and solving it requires a more "high-level" view of what the entity bean is trying to provide in general. (See Item 40 for more details about entity beans and their consequences.)

As a result, the items presented in this book strive to help developers recognize efficiency not at a language level but at a systemic and architectural level. Many of these items will be familiar ground to some; many will be simple codification of something some readers "always knew." That's OK—what's "intuitive" to one reader will be new to another, and vice versa.

In addition, I have carefully tried to avoid walking on familiar territory. A number of books have been released that discuss best practices and effective use of the virtual machine and/or the language; see the Bibliography for a complete list. As a result, discussions of material covered there won't be repeated here unless it has some particular relevance or application within the enterprise space.

Toward that end, I will refer frequently to patterns already established within the enterprise Java literature space; in particular, I will tend to lean heavily on Fowler's *Patterns of Enterprise Application Architecture* (Addison-Wesley, 2002), Hohpe and Woolf's *Enterprise Application Integration* (Addison-Wesley, 2004), and Alur, Crupi, and Malks's *Core J2EE Patterns,* 2nd ed. (Addison-Wesley, 2003), among others. (Again, see the Bibliography for a complete list.)

Where a pattern is cited by name, I use the standard Gang-of-Four pattern citation format, citing the pattern name with its page number in parentheses behind it; however, because these patterns come from different sources, I also put the author's names (or "GOF" for the Gang-of-Four book, *Design Patterns*) as part of the page citation. So a reference to the Data Transfer Object pattern from Fowler's *Patterns of Enterprise Application Architecture* book will be cited as "Data Transfer Object ([Fowler401])".

Acknowledgments

Authors have a tendency to spend a lot of time thanking people; there's a reason for that.

First and foremost, I want to thank my peers in the industry, most of all my fellow instructors at DevelopMentor. Kevin Jones, Brian Maso, Stu Halloway, Simon Horrell, Dan Weston, and Bob Beauchemin all served as sounding boards for the topics in this book at some point over its 30-month gestation. Tim Ewald, Don Box, Fritz Onion, Keith Brown, Mike Woodring, Ingo Rammer, and Peter Drayton all gave me insights into the Java platform by routinely smashing my preconceptions and assumptions in discussions that had nothing to do with Java. Outside of DevelopMentor, my fellow NoFluffJustStuff Symposium speakers Dion Alamer, Bruce Tate, Mike Clark, Erik Hatcher, Glenn Vandenburg, Dave Thomas, Jason Hunter, James Duncan Davidson, and Ron Bodkin, among others, all forced me to justify my assertions, questioned my conclusions,

and offered suggestions and tips designed to make the book a stronger work. Thanks to Jay Zimmerman for inviting me to be a part of NoFluffJustStuff in the first place. Numerous other speakers at conferences far and wide (far too many to list here) played a similar role, known in the common vernacular as "keeping me honest."

Second, I must tip the hat to the staff at Addison-Wesley. This book took over twice as long as it was supposed to, and never once did Mike Hendrickson, the editor who started the project, nor Ann Sellers, the editor who inherited the project, use anything but polite language when asking if it was done—their patience far exceeded what mine would have been in their shoes. The reviewers, both of the content when it was hosted on the Web via my blog, as well as the more finalized manuscript drafts, did a Herculean job reading through the material and offering up copious corrections, suggestions, enhancements, and ideas. Thanks to Matt Anderson, Kevin Bentley, Dave Cooke, Mary Dageforde, Kevin Davis, Matthew P. Johnson, and Bruce Scharlau for all their help.

Writing this book has been both a labor of absolute love and an exercise in abject terror. While I've always looked for a project like this to let me rant about my thoughts on enterprise Java development, few books have "raised the bar" as the previous *Effective* books have done. With *Effective C++,* Scott Meyers gave me (and millions of other neophyte C++ programmers) the leg up I needed to start using C++, rather than just flailing around with it. Then Joshua Bloch wrote *Effective Java,* bringing the beauty of the item format to the Java platform. Elliote Rusty Harold further upped the ante with *Effective XML.* If ever an author were looking for an intimidating set of authors to follow, those three fill out the set admirably. Fortunately, help was available in the form of Scott Meyers, who spent almost as much time reviewing and criticizing (in the good sense) this book as I did writing it. His comments and insights helped transform a tolerable collection of suggestions into the book you see in front of you. Scott, I owe you a tremendous debt of gratitude, both for your help during the last year and your help 10 years ago as I struggled to understand C++. It has been a privilege and an honor to do this with you; thank you.

Finally, of course, I must thank my family and friends, those loving people who periodically staged interventions and dragged me, kicking and screaming, back into this thing they kept calling "the real world": parties, holidays, even just hanging out for a night or two relaxing over the

Nintendo64 or XBox. Though I'll never admit it out loud, they kept me sane in the face of the mounting pressure of writing this book during the last two years.

Reporting bugs, making suggestions, and getting book updates

I have tried to make this book as accurate, readable, and useful as possible, but I know there is room for improvement. (There is *always* room for improvement.) If you find an error of any kind—technical, grammatical, typographical, spiritual, *whatever*—please tell me about it. I will try to ensure the mistake is corrected in a future printing of the book, and if you are the first person to report it, I will happily add your name to the book's acknowledgments. Likewise, if you have suggestions or ideas on how to improve the book for subsequent revisions or editions, I'm all eyes and ears.

I continue to collect guidelines for effective enterprise programming in Java. If you have ideas for new guidelines, I'd be delighted if you'd share them with me. You can either find me on one of several public Java programming mailing lists, the predominant one being the ADVANCED-JAVA list at DISCUSS.DEVELOP.COM, or you can reach me at:

Ted Neward
c/o Addison-Wesley Professional/Prentice Hall PTR
Pearson Technology Group
75 Arlington St., Suite 300
Boston, MA 02116

Alternatively, you can drop me an email at ted@neward.net.

I maintain a list of changes to this book since its first printing (including bug fixes, clarifications, commentary, and technical updates) on the book's blog, http://www.neward.net/ted/EEJ/index.jsp. Please feel free to post comments and/or errata there if you wish to share them with your fellow readers.

Enough preliminaries. On with the show!

List of Abbreviations

ACID	atomic, consistent, isolated, and durable
AWT	Abstract Windowing Toolkit
BLOB	Binary Large Object
BMP	Bean-Managed Persistence
CMP	Container-Managed Persistence
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
DCOM	Distributed COM
DHTML	Dynamic HTML
DMZ	demilitarized zone
DNS	Domain Name System
DOM	Document Object Model
DTC$	distributed transaction controllers
EJB	Enterprise Java Bean
HTML	Hyper-Text Markup Language
HTTP	Hyper-Text Transmission Protocol
IDL	Interface Description Language
IIOP	Internet Inter-Orb Protocol
IPC	interprocess communication
J2EE	Java 2 Enterprise Edition
J2SE	Java 2 Standard Edition
JAAS	Java Authentication and Authorization Service
JAXB	Java API for XML Binding
JAXM	Java API for XML Messaging

JAXP	Java API for XML Parsing
JAX-RPC	Java API for XML RPC
JCA	Java Connector API
JDBC	Java DataBase Connectivity
JDK	Java Development Kit
JDO	Java Data Objects
JESS	Java Expert System Shell
JIT	just-in-time
JITA	just-in-time activation
JMS	Java Message Service
JMX	Java Management Extensions
JNDI	Java Naming and Directory Interface
JNI	Java Native Interface
JNLP	Java Network Launch Protocol
JRE	Java Runtime Environment
JSP	Java Server Pages
JSR	Java Specification Request
JSSE	Java Secure Sockets Extension
JTA	Java Transaction API
JVM	Java Virtual Machine
JVMDI	Java Virtual Machine Debug Interface
JVMPI	Java Virtual Machine Profiler Interface
JVMTI	Java Virtual Machine Tools Interface
LAN	local area network
MDBs	Message-Driven Beans
MIB	Message Information Block
MVC	Model-View-Controller
NAT	Network Address Translation
NFS	Network File System
OODBMS	object-oriented database management system
ORB	Object Request Broker
OSI	
OWASP	Open Web Application Security Project
POJOs	plain old Java objects
POP3	Post Office Protocol v 3

RDBMS	relational database management system
RMI	Remote Method Invocation
RMI/IIOP	RMI over IIOP
RMI/JRMP	RMI over Java Remote Method Protocol
RPC	Remote Procedure Call
SAX	Streaming API for XML
SMTP	Simple Mail Transport Protocol
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
STL	Standard Template Library
SWT	Standard Widget Toolkit
TLS	Transport Layer Security
TPC	two-phase commit
TTL	time-to-live value
URI	Universal Resource Identifier
URL	Universal Resource Locator
URN	Universal Resource Name
VM	virtual machine
W3C	World Wide Web Consortium
WSDL	Web Services Definition Language
WS-I	Web Services-Interoperability
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	XSL:Transformation, commonly also written as XSL:T