

# Book Review

## BOOK DETAILS



### JMX in Action

Authors: Benjamin G. Sullins, Mark B. Whipple

Publisher: Manning Publications Co.

Publish Date: October 2002

Pages: 396

ISBN: 1-930110-56-1

[Publisher's Book Description](#)

Review Date: January, 2004

## REVIEWER

Tom McQueeney is a Denver-based Java architect and developer for Idea Integration, a national consulting company. He was elected as the 2003 Vice President responsible for scheduling speakers and the 2004 President of the Denver Java Users Group.

## REVIEW

*JMX in Action* starts gently, with a discussion of application management and how JMX fills that role, then builds your knowledge step-by-step into advanced application management techniques. The JMX terms -- *agent*, *MBean*, *MBean server*, *manageable resource*, *management application* -- are clearly defined in chapter 1, which provides an overview of the JMX architecture. By chapter 2, you're writing "Hello World" the JMX way. The book then digs into the JMX guts by describing the three types of MBeans that were defined as of the JMX 1.0 specification. By the end of the book, you're using the sample code to monitor and manage EJBs.

The book lives up to its title by providing working, straightforward code samples in almost every chapter to help explain the topic at hand. The authors also provide a few helper classes to make coding your MBeans easier.

*JMX in Action* assumes no prior experience with JMX, and requires no Java knowledge beyond J2SE. Readers will benefit by having some familiarity with RMI, and the advanced chapters on integrating JMX with J2EE are more easily understood with some knowledge of EJBs and JMS. But the authors describe what the advanced sample applications are doing for those new to J2EE.

JMX, the Java Management Extensions, is an optional extension to the J2SE. It provides a standard interface for managing and monitoring applications. JMX isn't magic, in that it has always been possible to write an application that can be

manipulated and monitored while running. But JMX provides standard APIs to make these handy features easier to use. Once your application or component has been "instrumented" with a managed bean (called an MBean), any JMX-aware management application will know how to manage and monitor that application by talking to the MBean. Additionally, with JMX helper classes available from Sun Microsystems (free for internal use), developers are saved from writing the JMX connectors that provide remote management capabilities.

Who can benefit from using JMX? Developers who wish they could change an application's configuration without stopping it, or who wish they could pry open a running application to figure out what it is doing -- without having to slog through log files trying to figure out the application's current state.

Besides being able to manipulate an application and monitoring its state, JMX defines a standard way for an MBean (which can be a component monitoring an application or hardware, or could be the application itself) to notify external monitoring applications that something interesting has happened. The "interesting" event could be as critical as a crash, or as routine as a monitored cash register exceeding its maximum cash level, telling a manager it is time to move some of the money to the safe. An MBean can be written to emit notifications, or an MBean can be added to monitor another MBean for interesting events.

This flexible and dynamic nature of JMX makes it a powerful tool worth learning. The entire JMX framework is predicated on the notion that everything should be configurable and modifiable without ever stopping your application or your MBean server. In fact, not only can you add, remove and change MBeans at runtime, an MBean server can load classes outside of the MBean server's classpath using the remote class loading "M-let" service, described in the book.

JMX's remote manipulation and configuration facilities bring up a security question of just who should be allowed remote access to MBeans. But security is one area the book lacks information. The Sun-provided RMI and HTTP "protocol adapters," used in the book's examples to control MBeans remotely, perform no security checks. Of course, developers and vendors are free to write their own security-enabled protocol adapters.

This book is geared for action: *JMX in Action* has you running your first managed application by page 35. By the end of the book, you feel you can write MBeans to manage standalone applications and web applications running on a J2EE server.

**Some of the book's other high points:**

- The book explains the various types of MBeans and why you'd want to use them. The book starts with the simple, Standard MBean, and moves to the more dynamically configurable Dynamic MBeans and Model MBeans. And although the book was written before the Open MBean's specification was

finalized in JMX 1.1 and made mandatory in version 1.2, there is an appendix describing the uses and reasons behind Open MBeans being added to JMX.

- The code samples, downloadable from the book's web site, <http://mannings.com/sullins>, are clear, simple and highlight exactly what that section of the book introduced. All code samples can be compiled and run with free software. For nearly all the code, you just need a Java 2 JDK, Sun's JMX reference implementation, and Sun's JMX "remoting" add-on.
- The authors provide excellent online support. The publisher's website has an "Author Online" section where you can post a question. The primary author, Benjamin Sullins, typically responds the same day -- often within a few hours. The book does have a few errors, which readers have pointed out, but the errors usually are obvious. Only one error in a code sample had me hunting around for a half-hour trying to figure out why an MBean wouldn't remotely deploy. The stack trace, though, was enough to point me to a missing constructor.

#### **Some of the book's flaws:**

- You have to hunt around to find the necessary JAR files to compile the sample code. The book was written before the JMX 1.1 and 1.2 specifications were published, and uses the 1.0 reference implementation that is no longer available. To compile the code, you should download the 1.1 reference implementation plus Sun's "Remoting" package available on the JMX web site (<http://java.sun.com/products/JavaManagement>). If you want to be brave and use JMX 1.2, you'll need to update the implementation of an MBean server in chapter 9 with methods added to the interface in 1.2.
- Like many technical books, the source-code listings have inconsistent and confusing indentation that likely took place in the typesetting process. JMX is definitely useful, and is starting to gain attention as more J2EE application server vendors add JMX support. For example, WebLogic Server comes with an MBean server that allows server runtime changes with JMX, and the JBoss application server uses MBeans as its component model, building features like the servlet container as runtime-configurable MBeans.

But even without using an application server, and even before more commercial JMX products come on the market, *JMX in Action* will show you how you can take advantage of JMX today to make your applications more configurable and manageable -- without having to stop and restart them.