Those Pesky Errors

Catching and handling exceptions is as important as writing your application

by Peter **VARHOL**

Peter Varhol is a technical evangelist for Compuware Corporation. He has written 10 books on computer software and a mystery/suspense novel called *On Deadline* (published by Writer's Showcase Press, October 2001). You can reach him at **Peter.Varhol@compuware.com**.

Nothing is more annoying in a Web application than error messages. In many (but not all) cases, an error means that you did something wrong, and after spending a lot of time and effort in developing the application, an error seems to get great pleasure out of pointing out your mistakes.

Another cause for errors is the failure of some software or hardware component associated with your application. The operating system hosting one of the servers may have crashed, the database connection could have hung, or the network could be down once again. A few seemingly innocent activities, combined together, could bring down an application. Often a Web-based application works fine, but is brought down by a volume of traffic that exposes scalability flaws.

But errors do more than simply annoy the programmer. They also annoy users; and they may annoy users enough that they decide not to bother with the application. Errors may cost you or your employer money, either in lost business or in calculating costs, inventory, or other types of transactions.

The quality of an application is more important than just about anything else in the development process. And no matter how well you write code, you will have errors in your end result, often because end users use your application in very unexpected ways.

This means that, rather than finding and fixing all possible errors and exceptions, your goal is to guard against them in your code. Identifying and catching exceptions involves anticipating everything that can go wrong in the application and handling them as gracefully as possible.

There are Errors, and Then There are Errors

Application errors can be classified into three categories. Language errors result from improper use of the programming language. For example, you may pass the wrong type of variable, such as an integer when your code expects a string. These types of errors are almost always caught before your application goes into real use.

Logic errors, the most common types of errors, occur when you don't think through the implications of a particular statement or operation. Often this means that a particular operation, normally legal, becomes illegal under certain circumstances. Probably the most obvious of this type of error is division by zero.

Infrastructure errors are errors within your support programs, or with interactions between your code and support programs. These errors can occur for many reasons, most of which have to do with the stability of the support programs, such as the servlet engine, database, or Java virtual machine (JVM). Usually, you have at your disposal only a minimum number of procedures to protect your application from infrastructure errors.

In the case of logic errors, much of the burden is on you to write code that minimizes such errors. In all likelihood, even the best design and most rigorous testing won't catch all of them, and depressingly few applications have the advantage of both superior design and superb testing. So it's incumbent on you to realize that errors will exist in your code, and users will find a way to use the application in ways you never expected. So no matter how confident you are in your code, catching and handling errors and exceptions is as critical as writing the application itself.

Java offers facilities that aid in the catching, identification, and diagnosis of run-time errors. Because Java is an object-oriented language, it creates objects that store information about an error. The primary error object classes are RuntimeException and Error. RuntimeException is the class of primary interest in your application. Often you have a greater level of control over this type of error, because it frequently emanates from your code. It's often possible to use the information in RuntimeException objects to recover from the error.

The Error object is created whenever the error is with support software such as the JVM or servlet engine. This object is of less interest, only because such errors are by their very nature random and varied. These types of errors do occur, but you have far fewer tools at your disposal to handle them. Often it's not possible to recover from an error that creates an Error exception.