ABLECOMMERCE
www.ablecommerce.com

RACKSPACE

www.rackspace.com

CORDA

www.corda.com

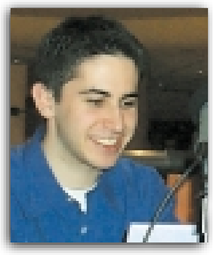
subscriptions:
For subscription requests please call
1 800 513-7111 or go to: **www.sys-con.com**
cover price $8.99/issue
domestic $89.99/yr. (12 issues)
canada/mexico $99.99/yr
overseas $129.99/yr
back issues $12 U.S. $15 all other

editorial offices: SYS-CON MEDIA, INC.
135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 **Fax:** 201 782-9600
COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
is published monthly (12 times a year)
for $89.99 by **SYS-CON Publications, Inc.,**
135 Chestnut Ridge Rd.,Montvale, NJ 07645

postmaster: send address changes to:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON MEDIA
135 Chestnut Ridge Rd.,Montvale, NJ 07645

distribution in USA:
by **International Periodical Distributors**
674 Via De La Valle, Suite 204,
Solana Beach, CA 92075
Phone: 619 481-5928

<CF Exterminate>



BY **ROBERT DIAMOND**

No, <CF Exterminate> isn't a new ColdFusion tag coming in Neo that you just haven't read about yet. It's my witty (at least I think it's witty) way of introducing debugging, one of the focuses of this month's issue. It's something that every developer has certainly faced at one time or another, and if you think you haven't – well, you're probably kidding yourself.

We've got two great features this month on debugging that are must-reads for less experienced developers and should-brush-up-ons for the advanced. Charlie Arehart, our **Journeyman** extraordinaire, has written this month's cover story on e-testing. Kevin Schmidt follows up with coding tips and some practical advice for dealing with the stresses of bug-fixing. Ian Rutherford starts Part 1 of a two-part series on stored procedures; Jackson Moore writes about protecting Web applications from untrusted data sources using validations and other ColdFusion features to keep you safe. Guy Rish has contributed Part 6 of his exceptional series on Java and CF. All this and much more in this month's issue…

On the Way from Macromedia…

Macromedia has also provided some exciting new information regarding the release of ColdFusion 5 UltraDev 4 Studio being unveiled at Macromedia DevCon. Stay tuned for full coverage in upcoming issues, but here are some of the highlights…

ColdFusion Studio and UltraDev work so well together because of their complementary strengths, and in the latest bundled release, whose beta we've been playing with here at ***CFDJ***, they're not going to disappoint. The improvements and new features in ColdFusion Studio 5 are useful for those like me who live off hand-coding. By bundling it with UltraDev 4, what they hope to provide is the best of both worlds for those who code by hand and those who prefer a visual tool.

For starters, all of the core editor features such as Tag Editors, Tag Insight, and Tag Inspector now have full, integrated support for the new CF 5.0 tags. There's also a new Custom Tag Wizard to write custom tags and then create and edit them with the Tag Insight and Tag Inspector. The new Studio also contains XHTML support, based on the W3C specifications for XHTML 1.0. The extensibility of XHTML gives you the flexibility to incorporate emerging Web standards.

They've also added lots of updates designed to make common tasks simpler…. The new Secondary Files tab will let you browse local and remote files simultaneously and drag and drop files between the two files' tabs. Auto Backup – which I'm sure will save me from time to time – automatically saves files at specified time intervals and allows users to recover and restore backed-up files. Last but not least, of course, is tighter integration with Macromedia's other products, and a number of smaller updates. Check out ***CFDJ*** next month for more information on those…

Robert Diamond

ROBERT@SYS-CON.COM

ABOUT THE AUTHOR
Robert Diamond is editor-in-chief of **ColdFusion Developer's Journal** *as well as* **Wireless Business & Technology**. *Named one of the "Top thirty magazine industry executives under the age of 30" in Folio magazine's November 2000 issue, Robert recently graduated from the School of Information Studies at Syracuse University with a BS in information management and technology.*

CFDJ FEATURE

BY CHARLES AREHART

E-Testing: Debugging Your Projects

When running the code isn't sufficient

You're nearly finished with the code you've been slaving over for hours (or days or weeks, or maybe it was just a few seconds' effort). You're about to turn it loose for your customers to enjoy. The questions are: Is it ready? Do you know it will work? Have you tested it? If so, how? Indeed, should you be turning it over to the users as the next step?

Running the code to make sure it works is the most common "test" performed by beginning and even many experienced developers, but it's rarely adequate. There are just too many things that can go wrong within a given CF template, its queries, and its resultant output to trust to eyeballing the code and crossing your fingers. Further, what may work for you may fail for your users for any number of reasons.

Factor in the consequences of a larger load, or fragile server hosting, and the potential for problems with your program, application, and site multiply. If you add multiple developers, then the need for testing, indeed for "configuration management," becomes more evident. Finally, consider that a bug is much less costly to repair the earlier it's caught in the life cycle of an application.

Whether you're a lone code-slinger or part of a large team, you need to know about testing, but many CF developers aren't aware of the options that may be available to them already, without cost or much effort.

This article delves into the issue of application testing, or *e-testing* as some have come to call it, which spans a variety of tools and programming practices. Along the way we'll cover HTML- and CFML-oriented tools, as well as some that span Web application testing in general.

A CF Programmer's Testing/Debugging Tool Bag

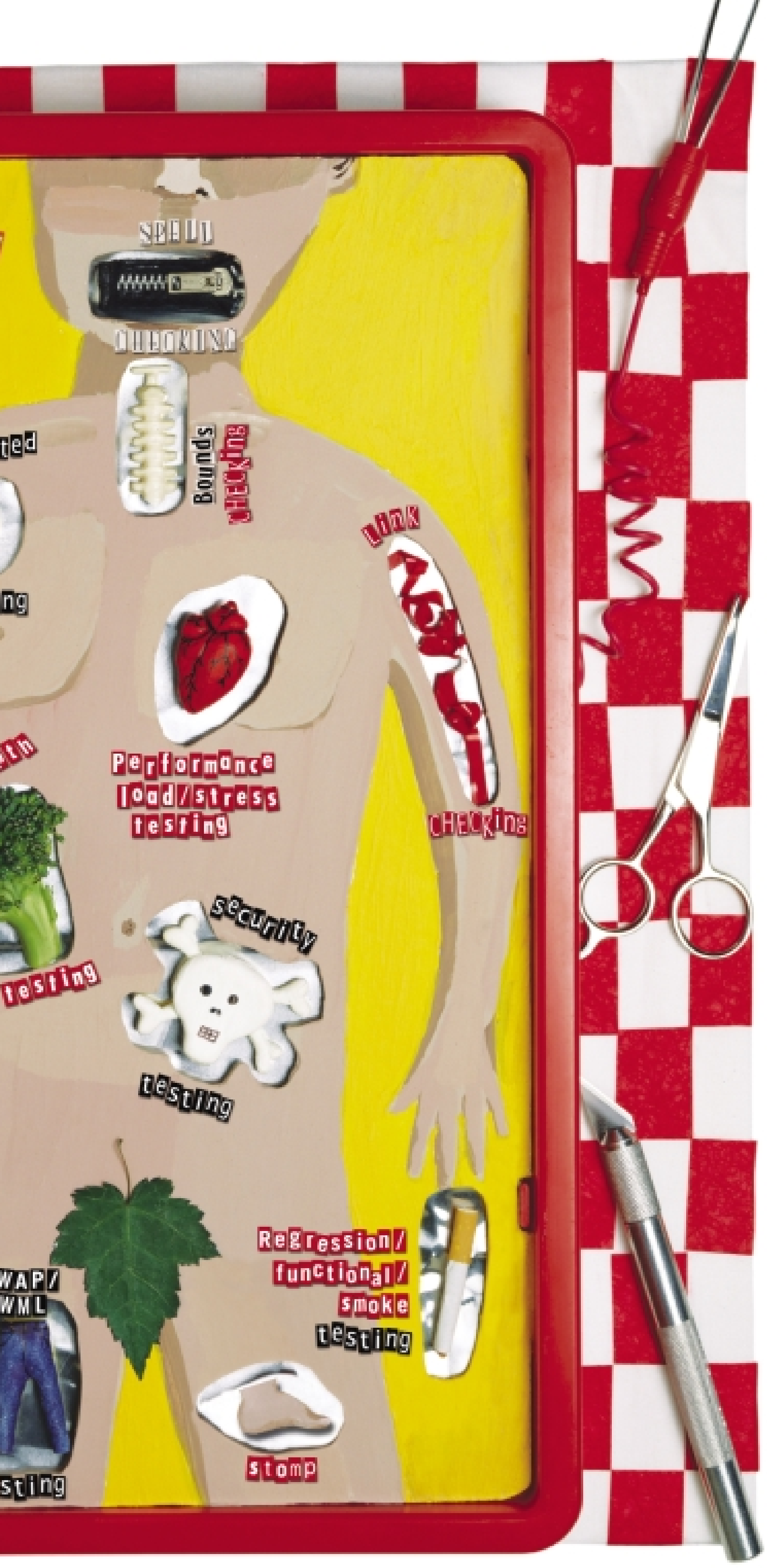
If you're not performing any of the following testing (or using any of these sorts of tools) in your day-to-day programming practice, you're exposing yourself, your company, and your clients to a preventable risk that may be costing additional time and effort in debugging. The testing/debugging opportunities are broken into three categories:

1. HTML-oriented testing
2. CFML code testing
3. Web application testing

We'll take a brief look at each of these categories of tools and tell you where you can learn more about them.

HTML-Oriented Testing

The first group, HTML-oriented testing, includes at least 11 different kinds of tests. CF programmers often dismiss them, but these freely available tools can be valuable if used effectively. The challenge is that generally they can't be used against your source CFML code but instead must be used against the generated result of your templates (and tested for a variety of browsers). I'll show you how to solve that. Let's look at each of these 11 tests.



The simplest testing you can do is HTML validation: making sure that the HTML you're creating is valid. This may seem incredibly mundane, perhaps even beneath you. But incorrect HTML may render your display flawed at best, or cause your page or part of it to fail to appear to the end user at all. Have you ever left off a closing table tag? Such a page may render fine in Internet Explorer, yet there's a total failure to display that table (and perhaps the entire content of your table-formatted page) in Netscape.

The problem for CF developers is that while we do create HTML in our templates, more often we create CFML code that in turn creates HTML. Many HTML validation tools built into editors like HomeSite and Studio are useless: they can't tell that an apparently errant tag is being sent to the browser only at runtime based on a CFIF test for a given variable. More important, the errant HTML may only be generated at runtime.

But we shouldn't conclude that no HTML testing is possible. Instead, we need to test the *output* of our pages. There are two ways to do that. One, you could manually copy and paste the output of a page into a new blank page in HomeSite/Studio, then run Tools>Validate Document. (If you don't care for the kinds of errors reported, or want to change the browsers you intend to support, see Studio's Options> Settings> Validation to modify the configuration.)

The other approach, which may be less labor-intensive and can even be automated, is to use a Web-based HTML validation service, such as that at http://validator.w3.org. You can point it at a CFML-generated page (meaning a URL to a .cfm file) just as you can point it at an HTML file (remember, to the browser – and this tool – it's HTML either way).

Other HTML-oriented tests available at this site (and others like it) include:

- Link checking
- Spell checking
- Document weight (download time) testing
- Cascading Style Sheet validation
- Color depth (browser-safe palette) testing
- Image compression testing

The first two can also be performed within Studio via cut-and-paste from your CFML output. The others can be done at the w3.org site as well as at NetMechanic.com and Netscape's "Web Site Garage" at http://websitegarage. netscape. com/O=wsg/tuneup_plus/index.html. Do a search on "html validation" in your favorite search engine to find more. Some problems are still more challenging to test than these simple HTML validations.

We developers often work with the latest and greatest in terms of bandwidth and monitor size, and it's all too easy to forget that not everyone experiences the same luxuries. And for some people, disabilities create an extreme burden in visiting your site.

The first issue is browser-size testing: Will your page fit in various screen resolutions? What looks great on your 19-inch monitor at 1024x768 resolution may not fit at all on a more typical 15- or 17-inch monitor running at 800x600 (or worse, 640x480). This is generally just a matter of making good HTML design choices, such as not hard-coding excessive widths in <TABLE> and <TEXTAREA> tags. Fortunately, there are tools built into HomeSite/Studio that will help: while viewing a page within the internal browser (F12, or View>Toggle Edit/Browse), there's an option that looks like two crossed arrows offered within a set of tools shown just above the browsed page (see Figure 1). Hovering over this icon will display the name "Browser Size" and clicking it will offer the choice to display the page within a frame that simulates 800x600 or 640x480 resolution. This is much easier than actually changing the resolution on your monitor. Be sure to set it back to "Fit to Window" when done.

There are also Web-based services that will provide you with a rendering of your page at various sizes, including BrowserPhoto at the aforementioned NetMechanic.com and the popular shareware tool BrowserSizer (search for it at your favorite shareware repository such as download.cnet.com).

Continuing the theme of considering your visitors' needs, there's no more important issue for some developers and visitors than the issue of accessibility testing, that is, ensuring that disabled Web visitors (blind, color-blind, deaf, etc.) can use your site. For some CF developers, it's not a matter to be dismissed: Section 508 of the Federal Rehabilitation Act mandates that federal agencies make their electronic and information technology accessible to people with disabilities. Even if you're not a federal agency, consider that you may increase your audience significantly at a small cost. Most existing Web sites require only minor modifications to make them accessible to the estimated 750 million people worldwide who have disabilities. The modifications can be beneficial to all users. There are tools to help test (and rectify) accessibility issues, the most notable being "Bobby" at www.cast.org/bobby. For more resources see www.microsoft.com/enable/.

I may have saved the best form of HTML testing for last, in terms of sharing something that a lot of developers don't seem to have: JavaScript error awareness. No, this isn't quite about testing (while there may be tools that actually test JavaScript code, I'm not aware of them). I'm referring to the simple matter of your being aware when your JavaScript code is generating errors. Daily, I encounter sites with errors that the developers clearly have missed. The really pernicious thing is that they don't even know they're there. Did you know that, by default, modern browsers (both IE and Netscape) hide JavaScript errors? As a developer, you need to turn them back on! In IE 5.5, use Tools>Internet Options>Advanced>Browsing>Display a Notification About Every Script Error. To enable them in Netscape, see the Netscape client-side JavaScript Guide at http://developer.netscape.com/docs/manuals/js/client/jsguide/console.htm#1045065. Of course, this will now expose you to all the errors on pages you visit, but this is a small price to be sure that you see any errors in your own code.

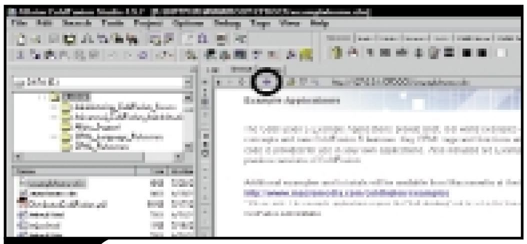


FIGURE 1: Browser size tool within Studio/HomeSite

CFML Code Testing

Some readers may feel cheated by the "testing" mentioned so far, with the focus on HTML tools. Others may have wondered from the beginning if any aspect of ColdFusion code can be tested. It can. And many developers don't know about these at all.

The first is CFML syntax checking. Why would we need this at all? Well, going back to the original premise of the article, if you're about to release your code for users to experience, have you really tested it thoroughly? Or did you just make a "simple change" that you "know" will work? Maybe you're reluctant to take the time to even just run the code to fully test it (it involves too many steps to drill down to it). At a minimum, before turning the code over to production (or uploading it to the live server), you should at least ensure that you haven't introduced any CFML syntax errors.

How can you do that? The built-in CFML Syntax Checking tool is the solution. Available as of Release 4 and originally intended to serve as a tool to ensure that old code compiled in the new release, it's useful for just the kind of testing we're discussing. Simply point your browser at http://localhost/CFDOCS/cfmlsyntaxcheck/cfmlsyntaxcheck.cfm. It's a form that asks for the name of a directory (and/or file name) whose code you want to test. It can scan through subdirectories (check the option "recurse") and will report if any template has a CFML syntax error. It stops on the first error it finds for a given template, so after fixing any error, rerun it until no errors are reported. It's designed to be run only from your local workstation, but the application.cfm in CFDOCS/cfmlsyntaxcheck can be modified to remove that restriction.

As useful as that can be, it's still just a pretty simple test: Will the code compile? It doesn't tell you much about the quality of your code, or whether it will perform well, or whether there are any of the many common CFML coding mistakes people often make. For this we need CFML code validation. Sound like a pipe dream? Believe it or not, there is a tool to do that. It's not free. It's a Web service from CF maven Steve Nelson, available at www.secretagents.com/products/index.cfm?fuseaction=product&product id=1. Some will find it worth every penny.

The service, named *Stomp* (as in stomping out bugs), is available for trial and performs over a hundred very useful CF-specific tests. You zip up and submit a set of code to the parsing tool and after several minutes (typically) it will e-mail you a link to a report on the site. It will show each template and its errors, as well as explain why the problem is worth fixing and, most important, how to do so. Some of these are classic; some you may not have considered.

For instance, in a sample test of just a few templates of my own code, I was warned about:

- Not having maxlength defined for a form input field (which could lead to truncation problems on a database update in the action page)
- Using Select * in some SQL statements (which may retrieve more columns than needed)

This was just a small set of code I tested. Stomp looks for many other

errors. Stomp also gave me several useful suggestions just in that small code sample, including:

- Recommending that I use <CFSETTINGENABLECFOUTPUTONLY="yes> for some templates that had no text for display (or all the text was within CFOUTPUTs)
- Detecting that I'd failed to test for a file's existence before using it as a source file in CFFILE
- Detecting that I had a .cfm file with no CF code and recommending that I change it to an .htm extension instead
- Reminding me to consider the case-insensitive versions of the functions Find, Replace, and Compare, and related list and regular expression equivalents
- Suggesting that I test for a recordset having zero records before doing a CFOUTPUT loop over the query result set
- Reminding me to consider using <CFIF NOT len(x)> rather than <CFIF len(x) IS 0>
- Suggesting I write the CFID and CFTOKEN variables on any links or form actions rather than rely on cookies to enable session support, and even catching each instance of code where I had an HREF or Form ACTION that failed to pass the tokens

If I were to apply these suggestions (many of which I'd failed to use in multiple templates), I might improve the quality of my site dramatically. Of course, some will argue, "I already know to do (or not do) that stuff." Hey, so do I. But due to laziness (or sloppiness or old code not reviewed in a long time), the problems crept in. It was a great lesson in humility. Then again, Stomp can catch hundreds of ColdFusion-related coding errors, so perhaps I should look at the bright side!

It's not a perfect tool. Some mistakes flagged as "hard failures" really ought to have been flagged as "potential failures" (the use of "Select *" was flagged as a "Hard Failure-ODBC"). That's only a minor annoyance considering the value it offers. Perhaps it should flag some errors as "potential failures." In any case another cool feature of the site is that since it's a Web-based service, you can provide such feedback or suggestions about each reported error via an available link. For a complete list of the tests (which itself may be useful) see www.secretagents.com/content/index.cfm?fuseaction=testlist.

Another tool that does some CFML code parsing is ParseDog (http://aloha-webdesign.com/dloads/cf/parsedog/releasenotes.htm), which focuses on identifying whether custom tag and CFINCLUDE files exist and has some extra features that may interest you.

Another aspect of CFML code testing is code timing: How long does it take to execute given code or a code block? There's no tool available (yet) to provide automated profiling of CFML code, but you can "roll your own." Look at the GetTickCount() function. The CFML documentation on the function gives a simple but functional example of its use. Another aspect of timing is how long code takes to compile, and how long queries take to execute. Answers can be found in the debugging information that can be enabled in the CF Administrator. It may not be appropriate to enable this in production, but that's why you ought to have a test environment that models your production environment.

The final aspect of CFML code testing may not really seem about testing at all, but in a way it is. Studio has a set of features that can help you see if the code you're creating is indeed valid. Consider it a "poor man's" CFML validation. While the cursor is on a CFML tag or its attributes, press Ctrl-F4 (or use Tags>Edit Current Tag). The tag's available attributes and their values will be displayed. Similar values can be obtained with the Tag Insight feature: while entering a tag and its attributes, wait one second after pressing the space bar and Studio will show you the tag's available attributes and values. Finally, Tag Help allows you to press F1 on a tag to see the CFML documentation for it. As of Release 4.01, the Insight and Help features are also available for functions. see the Macromedia documentation called "Using ColdFusion Studio" for more on these and related features.

Web Application Testing

I've identified the final group of testing somewhat more generically as Web application testing. It may not be about your CFML in particular, but how the application as a whole (including HTML, JavaScript, and CFML) works in both reasonable and unexpected situations. This sort of testing includes:

1. ***Data validation and "bounds" checking:*** Does your code handle unexpected input well?
2. ***Functionality testing:*** Does the code work as expected?
3. ***Security testing:*** Are only authenticated, authorized activities able to take place?
4. ***Regression/functional/smoke testing:*** Does the code produce the expected result when run after performing code changes?
5. ***Performance/load/stress testing:*** Can the code sustain a large load?
6. ***Concurrency testing:*** Are there problems when multiple users access the code, or when a single user opens multiple browsers?
7. ***Site monitoring:*** Is the site up? Is anyone aware when it goes down?

"Stomp can catch hundreds of ColdFusion-related coding errors"

The first three are tests that aren't easily automated. You simply need to create a set of tests to ensure that things work as expected and that unexpected things are handled properly. Most important: make sure the code does what the user expects (more on that, and "acceptance testing," later).

Still, while the creation of such tests can't be automated easily, it's possible to record the execution of your tests and then play back the tests in a repetitive manner to detect whether code that worked at one point in your development is no longer able to pass the recorded tests.

This is often called *regression testing,* though it's known by other names as well. The point is that you're trying to determine if solved problems remain solved, or if your code has "regressed" to a nonfunctioning point. Going back to the original premise, if you've made a change in some code that's deep in a set of steps that the visitor must do (perhaps order multiple items on a shop-

ping cart and then delete one), such automated testing tools can make it easy to retrace those steps as often as you like.

Predominant companies with tools in this space include:

- ***Empirix:*** www.empirix.com (formerly RSW Software)
- ***Mercury-Interactive:*** www.mercury-interactive.com
- ***Rational***: www.rational.com
- ***Segue***: www.segue.com

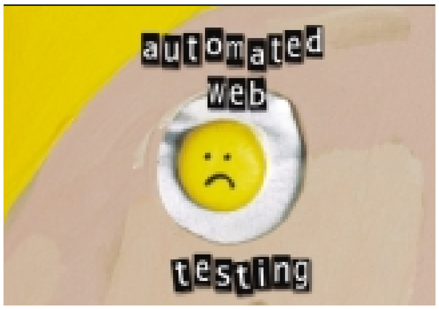
These companies' tools are mature and relatively expensive, but of substantial value for the benefit provided. A smaller and relative newcomer with a purportedly different approach and a much lower price is eValid (www.soft.com/eValid/).

Most of these Web app testing tools work as easily as a VCR, literally allowing you to hit a record button to start watching you run through your application while it tracks all your button clicks, mouse clicks, links taken, even JavaScript or Flash events fired (in better tools). They can then be played back repeatedly with relatively little effort and interaction. In fact, most tools include features to allow you to parameterize the tests when rerun so that some or all aspects of processing are driven by variable data extracted from a database or flat file.

Most of these tools allow the recorded, parameterized regression tests to be used for the next type of testing, *performance/load/stress testing*, which answers the question: Can the code (and environment) scale and sustain a large load? This may be where these tools really shine, because while the regression testing aspect can be a bit cumbersome when dealing with constantly changing code bases, the load test is more straightforward: you record the tests, parameterize them, and then use the tools' ability to create "virtual users" that mimic many users (perhaps thousands) hitting your site. These better tools do more than simply bang on a single page; they run through the same sort of multipage execution that works the way a user would. The tests are limited only by our creativity and the time to create the tests. While some tools use scripting languages to create tests, which are powerful but can be cumbersome to learn, others use the simpler record/play back approach.

Each of the aforementioned companies' tools can perform load/stress testing, but free tools are also available, including Microsoft's own Web Application Stress test tool (http://msdn.microsoft.com/library/en-us/dndu-won/html/d5wast_2.asp) and the Paessler Webserver Stress Tool (www.paessler.com/tools/WebStress/webstress.htm), which is free for five users, with a modest cost to add more users.

"Most of these Web app testing tools work as easily as a VCR"



Even if you don't think your site will need to support thousands (or hundreds) of concurrent users, consider using some form of load-testing tool to see what happens when even more than one user visits your site. Many an application has been brought to its knees by problems with, for example, sessions being unexpectedly shared among multiple users. This sort of problem can be caught with *concurrency testing*. And it's not only about multiple users accessing the code, but even a single user opening multiple browsers. Even using the simplest freeware tool causing multiple users to visit the site may reveal unexpected problems.

These automated testing tools are often dismissed as too expensive or too much work for a given project, but the cost and effort can be justified in the testing time saved (or facilitated). While free tools achieve some of this functionality, you often do get what you pay for. Most vendors of commercial products offer free trials, of course, but the good news is that some vendors are making available (or planning) low-cost developer editions or even limited function (but nonexpiring) trial versions. Rational now offers a free, full-function 50-user license of its SiteLoad tool. These are exciting developments for those who've dismissed the pricier tools in the past.

That last point about hosted services is worth expanding: most of the testing tool providers are now offering hosted services that can be quite useful and cost effective.

The final form of Web application testing can also leverage these other recorded scripts and can tell you about the ongoing state of the application once it's in production. *Site monitoring* asks the questions: Is the site up? Is anyone aware when it goes down? Besides the aforementioned toolmakers, other players in the space include freshwater.com, tracert.com, and others, some of which merely provide a useful mechanism to ping your site (or a given page) to ensure the site is up, and e-mail you when it goes down. Indeed, CF5's own probe mechanism within the new Administrator Tools might be used to provide a similar capability.

Other Testing

Still other tools can test different components or features you may be using:

- COM/EJB/CORBA/Java component testing
- Web service testing
- WAP/WML testing
- Search engine result optimization testing

We don't have space to get into all facets of Web application testing: I'll leave you to look into these on your own. Additional forms of testing often mentioned in the world of e-testing – including installation, configuration, usability, unit, integration, reliability/recovery, end-to-end, white/gray/black-box, and documentation testing – are also worthy of your further research.

Another aspect of Web application testing asks whether the code you're building is really what the client wants. This isn't really testing so much as prototyping, but it's an important matter nonetheless. Certainly, before rolling your application out from its initial design stage (indeed, before and during that design stage), you should use various means to prototype the application to obtain user approval. This is often called *acceptance testing*. The Fusebox community is strongly embracing "wireframing" to create simple but adequate representations of the look and feel (and flow) of a site. Even non-FuseBox applications can benefit from this approach, and previous ***CFDJ*** articles have addressed this (see

"Exploring the Development Process," by Hal Helms (Vol. 3, issue 3).

Finally, a CF project usually exists within the context of several other servers providing related services. Other opportunities for testing include testing the availability and performance of the Web server (and cluster, if any), the entire application server, the particular database (and database server, if any), the mail server, the ftp server, and so on. Consider these as you develop a testing plan.

• • •

I hope this trip around the testing world proves helpful for you. There's a lot more that we just couldn't cover, and many resources are worth reviewing. I've decided to develop a one-day seminar as a result of my research for this article. If you're interested, drop me a line.

Resources

Articles

1. "Stranger in a Strange Land: Bringing Quality Assurance to a Web Startup," by Lisa Crispin: www.stickyminds.com/docs_index/XUS247559file1.doc
2. "Testing Your Web Site," by Eric Kaufman: www.veritest.com/testers'network/Web_testing1-1.asp
3. "Understanding Performance Testing": http://msdn.microsoft.com/library/en-us/dnduwon/html/d5dp_lyover.asp
4. "Web-Site Monitoring Derails Problems": www.informationweek.com/805/webdev.htm
5. "Testing Considerations for Web-Enabled Applications," by Gerry Ocampo: www.veritest.com/tester-s'network/testing_considerations1.asp
6. "Why Test the Web? How Much Should You Test?" by Mike Powers: www.veritest.com/testers'network/Web_testing21.asp
7. White paper on Web testing: http://members.spree.com/oceansurfer/webtesting.htm

Books

1. Nguyen, H. (2000). *Testing Applications on the Web*. Wiley.
2. Plaine, S., et al. (2001). *The Web Testing Handbook*. Software Quality Engineering.
3. Dustin, E., et al. (1999). *Automated Software Testing: Introduction, Management, and Performance*. Addison-Wesley.
4. Graham, D., et al. (1999). *Software Test Automation: Effective Use of Test Execution Tools*. Addison-Wesley.
5. Perry, W.E., et al. (1997). *Surviving the Top Ten Challenges of Software Testing*. Dorset House.
6. Perhaps on a par with a book is this informative and contemporary thesis on Web application testing by Jesper Ryden and Par Svensson: www.stickyminds.com/docs_index/XUS512798file1.zip

Portal/Information Resources URLs

1. www.qaforums.com
2. www.sqe.com
3. www.stickyminds.com
4. www.softwareqatest.com
5. www.qacity.com
6. www.veritest.com/testers'network/
7. www.empirix.com/empirix/web+test+monitoring/resources/default.html
8. www.keynote.com/services/html/product_lib.html
9. www-svca.mercuryinteractive.com/resources/library/

About the Author

Charles Arehart is a certified Macromedia trainer/developer and CTO of SysteManage, a Macromedia partner. He contributes to several CF resources, is a frequent speaker at user groups throughout the country, and provides training, coaching, and consultation services.

carehart@systemanage.com

CFDYNAMICS

www.cfdynamics.com

CF USER GROUPS

Birmingham, AL
Jeff Bryant
jeff.bryant@mindspring.com

Huntsville, AL
Brian Yagar
brian.yager@redstone.army.mil

Phoenix, AZ
Sean Tierney
sean@pubcrawl.net

Tucson, AZ
Darcey Spears
dmspears@uswest.net

Bay Area, CA
Nathan Dintenfass
nathan@changemedia.com

Southern California
Leon Chalnick
lchalnick@advantasolutions.com

Southern California - Los Angeles
Sid Sheres
sheres@gte.net

Orange County, CA
Peter Amiri
peter@amiri.net

Sacramento, CA
Geoff Lilley
geofflilley@hotmail.com

San Diego, CA
Robert Munn
rob@illuminez.com

Orleans, Ontario
Clovis Paquette
clovispa@comnet.ca

Edmonton, Alberta
Jeff Acker
jeff.acker@topham.ab.ca

Lethbridge, Alberta
Chris Sorensen
chris@csystems.ab.ca

Montreal, Canada
John Kopanas
kopanas@videotron.ca

Ottawa, Canada
James Milks
jamesm@videotron.ca

Toronto, Canada
Kevin Towes
KTOWES@PangaeaNewMedia.ca

Vancouver, Canada
Cameron Siguenza
cameron@evolutionb.com

Colorado Springs, CO
Anthony Starkey
astarkey@mentiscorp.com

Denver, CO
Robi Sen
robi@granularity.com

Rocky Mountain
Mike Miller
mmiller@ewtechnology.com

Hartford, CT
Chris Graves
graves@rapidcf.com

New Haven, CT
Alfred DiMarzio
cfug@hbgraphics.com

Stamford/Westchester, CT/NY
Barry Hyman
bah@abodofoto.com

Delaware
John McKown
john@delaware.net

Gainesville, FL
Taner Aktas
aktas@ufl.edu

Jacksonville, FL
Russ Johnson
russ.johnson@cyberfusionsolutions.com

Orlando, FL
Randy Drisgill
rdrisgill@vshift.com

South Florida
Kevin Langevin
kevin@cfug-sfl.org

Tallahassee, FL
Benjamin Bloodworth
ben@electronet.net

Tampa Bay, FL
Kenneth Beard
kbeard2323@hotmail.com

Atlanta, GA
Adam Churvis
president@prodenhance.com

Atlanta, GA
Seth Land
gacfug@figleaf.com

Columbus, GA
Rob Seebach
rseebach@sfcts.com

Boise, ID
Paul Jenkins
pjenkins@doi.state.id.us

Chicago, IL
Robert Burns
bob-burns@mediaone.net

East Central Illinois
William Steiner
WilliamS@hkusa.com

Quad Cities
Jamie DeVolder
JamieD@4CS.COM

Springfield, IL
Gary Ashbaugh
ashbagd@mail.ioc.state.il.us

Central Indiana
Joshua Kay
josh@dataquix.com

Northern Indiana
Graham Pearson
gpearson@nicfug.org

Cedar Valley CFUG
David Chandler
David@ChandlerResearch.com

Des Moines, Iowa
Kevin Schmidt
schmidt@hungrycow.com

Louisville, KY
Tim Newton
TNewton@MercuryFusion.com

New Orleans, LA
Kathy Hester
kathyhes@yahoo.com

Annapolis, MD
Matt Schuster
mattschuster@erols.com

Baltimore, MD
Bret Peters
bmorecfug@figleaf.com

Broadneck High School CFUG
Rick Blaha
IRiCkYI@aol.com

Maryland
Michael Smith
Michael@teratech.com

Southern Maryland
Russ Mahan
rmahan@erols.com

New England Develpor's Network
Steve Casco
steve@ateaze.com

Detroit, MI
Matt McDonald
matt@stoneage.com

Mid-Michigan Group
Rick Mason
Rick@SeedChoices.com

Twin Cities, MN
Dan Chick
dan@danchick.com

Kansas City, MO
Bryan Laplante
directors@kcfusion.org

SoMoCFUG
Jeremiah Andrick
jandrick@theinteriorlife.com

St. Louis Metro Area
Dan Anderson
dan@psiwebstudio.com

Helena, MT
Tom Marino
tmarino@falcon.com

Omaha, NE
Mark Kruger
Mkruger@CFWebtools.com

Las Vegas, NV
Dave Byers
pegarm@lvcm.com

Concord, NH
Neil Giarratana
NGiarratana@ininet.com

Central New Jersey
Hien Nguyen
HNguyen7@prius.jnj.com

Northern New Jersey
Cindy Pomarlen
cjp@neotech-associates.com

South Jersey
Mark Stewart
m-stewart@home.com

Albuquerque, NM
Christopher Jungmann
jungmann@swcp.com

Albany, NY
Thomas McKeon
tmckeon@newkirk.com

Long Island, NY
Christopher Collins
chrisc@sitespan.com

New York, NY
Michael Dinowitz
mdinowit@houseoffusion.com

Rochester, NY
Dave Horan
dhoran@fusionproductions.com

Syracuse, NY
Laurie Ferger
lferger@twcny.rr.com

Charlotte, NC
Dave Crawford
dcrawford@charlotte-cfug.org

Greenville, NC
Steven Forehand
forehands@mail.ecu.edu

Raleigh, NC
Cheryl Elia
ccfug.mgr@listserv.unc.edu

Mid-Ohio Valley
Dave Hannum
hannum@ohio.edu

Cleveland, OH
Matt Newman
mnewman@newchanneltech.com

Columbus, OH
Angelo McComis
Angelo@CompuServe.com

Oklahoma City, OK
Glen Collymore
Glen-Collymore@ouhsc.edu

Tulsa, OK
Casey Bourbonnais
casey@cbfenterprises.com

Portland, OR
Jeff Cram
jcram@isitedesign.com

Roseburg, OR
Sharon Bland
sbland@douglasesd.k12.or.us

Harrisburg, PA
Neil Ross
info@codesweeper.com

Philadelphia, PA
Chris Swansen
cswansen@etechsolutions.com

Pittsburgh, PA
Kyle Jenkins
kyle@vaporbiz.com

State College, PA
Brian Panulla
brian@elmwoodmedia.com

Greenville, SC
David Goldfield
Dave@goldfield.com

Knoxville, TN
Doug McCaughan
doug@colknox.com

Memphis, TN
Michael Kubicki
mkubicki@univ-solutions.com

Nashville, TN
Chris Mosier
chris.mosier@e-maginative.com

Austin, TX
Mike Dickinson
cranch@texas.net

Dallas, TX
Patrick Steil
pmsteil@imailbox.com

Houston, TX
User Groups
usergroups@allaire.com

San Antonio, TX
Douglas White
doug@dwhite.ws

Utah State College
Jon Nelson
jnelson@nstep.net

Salt Lake City, UT
Mike George
ueberhund@hotmail.com

Provo, UT
Josh Trefethen
josh@exciteworks.com

Vermont
Sarah Moulton
info@mtbytes.com

Hampton Roads, VA
Raymond Camden
morpheus@deathclock.com

Northern Virginia
Bret Peters
novacfug@figleaf.com

Richmond, VA
Robert Baird
bbaird@btgi.net

Figleaf CFUG
Bret Peters
cfug@figleaf.com

Seattle, WA
Shannon Smith
president@cfseattle.org

Walla Walla, WA
Sarah Lewis
sarahlewis@alhena-design.com

Milwaukee, WI
Ron Kurtus
ron@ronkurtus.com

Wyoming
Matt Rodosky
mrodosky@wyoming.com

Nepal, Asia
Sudhir Raj Joshi
sjoshi@iapex.com

Australia
Marc Dimmick
marc@ibaseglobal.com

Canberra, Australia
Kunal Bhatia
kbhatia3@csc.com.au

New South Wales, Australia
Kym Kovan
kymk@microset.com.au

Northern Territory, Australia
Fred Lecul
flecul@paspaley.com.au

Queensland, Australia
Barry Moore
barry_moore@yahoo.com

Southern Australia
Brett Hales
brett.hales@baesystems.com

Tasmania, Australia
Mark Heseltine
HeseltineM@logica.com

Victoria, Australia
Jordan Thomas
jordan@vector7.com.au

Belgium
Denis Wauthy
d.wauthy@switchon.com

Rio de Janeiro, Brazil
Marcello Frutig
frutig@astrolabio.com.br

Sao Paulo, Brazil
Marcantonio Silva
marco@channelmedia.com.br

Central Europe
Sven Slazenger
slazenger@interlake.net

China
Feng Jian Feng
webmaster@cfwindow.com

Finland
Tero Pikala
tero.pikala@kolumbus.fi

France
Fabien Graille
FG500001@exchange.FRANCE.NCR.com

Ramstein Air Base, Germany
Tom Nunamaker
tom@nunamaker.com

Hong Kong
Shashank Tripathi
shashank.dp.tripathi@hk.pwcglobal.com

Hungary
Zolton Sheshler
ozoltan@yahoo.com

India
Zahid Java
zahidjava@yahoo.com

Indonesia
Agus Basuni
agus@digitaldevelopment.com

Ireland
Daryl Fullerton
daryl@thebiznet.net

Dublin, Ireland
Justin MacCarthy
maccarthy@iol.ie

Italy
Andrea Veggiani
andrea@dinamica.it

Okinawa CFUG
Michael Bradford
michael@devapps.com

Korea
HyeHyun Seo
hhseo@otozone.com

Malaysia CFUG
Halmi Yasin
affroman@affroman.com

Netherlands
Ron Pasch
ronnieo@dds.nl

Auckland, New Zealand
Amanda Irvine
amanda.irvine@delta.co.nz

Romania
Andrei Oprea
andrei@oprea.org

Johannesburg, South Africa
Steven Ringo
steven@tutuka.com

Spain
Jorge Pueyo
jpueyo@lander.es

Sweden
Stefan Gudmundsson
stefan@cfug-se.org

Zurich, Switzerland
Martin Buerlimann
bue@sercon.ch

Taiwan
Jack Jair
jack@leetide.net

Bangkok, Thailand
Paul Hastings
paul@tei.or.th

Turkey
Oguz Demirkapi
oguz@cftr.net

Dubai
Maryam Ebrahimi
CFUG@digiba.com

London
Niklas Richardson
niklas@ukcfug.org

Northern England
Stephen Moretti
stephen@cfug.org.uk

Southern UK
Stefan Elliott
cfug@four-runner.com

Ask the Training Staff

A source for your CF-related questions

BY
BRUCE
VAN HORN

Q: ***What's OnRequestEnd.cfm?***

A: OnRequestEnd.cfm is the twin brother of Application.cfm. When the code inside a file named Application.cfm is automatically included at the beginning of every .cfm file in the same subdirectory (and any directory below, provided that there isn't another Application.cfm in those directories), any code placed in a file named OnRequestEnd.cfm is automatically included at the end of every .cfm page.

Q: ***I come from an ASP background and am finding certain tasks difficult in ColdFusion. Here's my question: How do I find the size of a file on the server? I've looked everywhere. I have used the FileExists() function, but there really should be a FileSize() function.***

A: I agree. There should be a FileSize() function (could be a great user defined function!). The size of a file, however, can be easily (though not intuitively) retrieved using the CFDIRECTORY tag. This tag is usually used for listing the contents of an entire directory or for creating, deleting, or renaming directories, but it can be used to retrieve information about a single file. Here's an example:

```
<cfdirectory action="LIST"

directory="C:\Inetpub\WWWRoot\
Datafiles"
              name="qDir"

filter="AskCFDJ.mdb">

<cfoutput>#qDir.size#</cfoutput>
```

Q: ***I need to build a form that allows users to upload multiple files to my server. The <input type="file"> tag only allows a single file to be selected. How can I do this?***

A: While the HTML <input type="file"> tag only allows you to select a single file, you can use as many of these tags as you like in your form. The trick is to have a CFFILE tag on the action page for every file uploaded. You can accomplish this by naming all of your <input type="file"> tags with similar names, like upload1, upload2, upload3, etc. You can then loop over your form fields to dynamically create the necessary CFFILE tags in your action page. Listing 1 provides an example.

Q: ***I'm having trouble feeding a quoted list to the IN operator in SQL. I run a query to retrieve certain city names from a database and then use the QuotedValueList() function to turn the results into a quoted list to be used in another query to retrieve only those records in which the cities match the list. My second query looks like this:***

```
<CFQUERY NAME="GetCustomers"
DATASOURCE="#dsn#">
SELECT Name, Address1,
Address2, City, Phone
FROM Customers
WHERE City IN (#CityList#)
</CFQUERY>
```

When I look at the list, the city names are there with the appropriate single quotes around each entry (a requirement for the SQL IN operator when querying a text field), but when I run this query in CF, I get a database error. What am I missing?

A: The problem is that CF automatically "escapes" single quotes inside the CFQUERY tag, so your list actually winds up with two sets of single quotes around each entry. What you need to do is use the PreserveSingleQuotes() function on your list when you run it in CFQUERY. For example:

```
<CFQUERY NAME="GetCustoners"
DATASOURCE="#dsn#">
SELECT Name, Address1,
Address2, City, Phone
FROM Customers
WHERE City IN
(#PreserveSingleQuotes(CityList)#)
</CFQUERY>
```

Q: ***I really like the ability to create my own functions (UDFs) in CF 5. Is there a place like the Developers Exchange at Allaire.com where I can upload the ones I've written and see what other people have created?***

A: Yes. You should visit www.cflib.org, a great source of free UDFs founded by Raymond Camden and Rob Brooks-Bilson.

• • •

Please send your questions about ColdFusion (CFML, CF Server, or CF Studio) to AskCFDJ@sys-con.com. Visit our archive site at www.Netsite-Dynamics.com/AskCFDJ.

BRUCE@NETSITEDYNAMICS.COM

ABOUT THE AUTHOR
Bruce Van Horn is president of Netsite Dynamics, LLC, a certified ColdFusion developer/instructor, and a member of the CFDJ International Advisory Board.

Listing 1

```
Form page:
<form action="upload.cfm"
       method="post"
       enctype="multipart/form-
data">
<input type="File"
name="upload1"><br>
<input type="File"
name="upload2"><br>
<input type="File"
name="upload3"><br>
<input type="submit">
</form>
Action page:
<cfloop list="#form.field-
names#" index="FormField">
 <cfif FormField contains
"upload">
  <cffile action="UPLOAD"

filefield="Form.#FormField#"
          destination="C:\temp"
          nameconflict="
MAKEUNIQUE">
 </cfif>
</cfloop>
```

CODE LISTING
The code listing for this article can also be located at www.ColdFusionJournal.com

Defending Socrates

Real masters *do* test

BY
HAL HELMS

"The untested life is not worth living."
—Socrates

So Socrates says, but I think if we were to pick out one trait that novice developers share, it might well be the unwillingness to test code. Why is this? Well, I can't speak for everyone, but I remember when I first began developing applications; after struggling so long to master the obscure syntax and baffling concepts of programming, I managed to get my first application to run. What a heady feeling!

After that first app, there came a second and third and more. I started feeling that perhaps I had licked this programming stuff. A friend of mine back then had a favorite phrase, "Humility does not become the truly great." That sure sounded a lot better than dour old Socrates with his testing of everything. No wonder the city fathers tired of him. Besides, testing apps was for people too timid to trust their code. Real masters didn't need to test.

The problem, you see, was that though I had read a little Socrates, I had missed the words of another Greek, the playwright Euripides: "Whom the gods would destroy, they first drive mad." If you're familiar with Greek tragedy, you know that much of it is built on the notion that the main character succumbs to hubris, an overweening pride that exhibits itself in many ways such as – well, such as in thinking that only losers test code, perhaps. All I lacked for my own personal production of "Fall of the Mighty" was a Greek chorus.

And fall, I did. In a public, high-profile demo, my righteous code broke! Well, it didn't exactly "break" – the thin tissue of code that was making up my program "dissolved" and with it went my assurance that I had programming even slightly mastered.

That was not an enjoyable experience but, ultimately, it was a very good one; it helped me understand that programming really is not just "chess for geeks." It's a craft – one that demands discipline of all who would master it – that ultimately rewards those who persevere with the quiet satisfaction of a job well done.

This month ***CFDJ*** invites you to examine the related fields of debugging and testing. They are related, of course, but quite distinct. In debugging the programmer exercises his or her best efforts to make the application run. In testing, the programmer tries to make his or her creation fail. The untested application is not worth deploying.

We know all this, of course. Yet many of us don't test our apps; we may give them a quick run-through, but do we try to break them? Maybe not. And maybe the reason is that too often we're afraid of all the shortcuts and kludges and stuff that we managed to make work, but aren't too sure we want to lean on it too hard, much less test.

Or maybe we're afraid to examine our applications lest we discover that our code has somehow been infested with insects. There's good news on that score: there's no such thing as buggy code. Code performs exactly as we wrote it. The problem is that we don't understand what we wrote. It follows then that debugging is primarily a matter of knowing our code. As the Oracle of Apollo at Delphi tells us, "Know thyself" – or thy code, in this case.

I've come up with some little notes gathered from painful experience that I hope may help you. These are principles I keep returning to, and they've never failed me. I've broken them into debugging (D) and testing (T) tips.

Debugging

- **D1. *Reduce the problem to its essentials.***

I often receive e-mail from people who are trying to make their code work. "What's the problem?" I ask. "It doesn't work," they say.

That's not debugging. When I'm having a problem with some code, I strip away everything that isn't part of the problem so that I don't distract myself or, worse, miss a simple error while assuming the problem is deep and needs major surgery. Do you need all that JavaScript, that HTML code, the various other code that surrounds the problem area? If not, get rid of it while you're struggling to understand your code. And have only one issue per page. The goal is to isolate the problem so you know exactly what you don't understand.

- **D2. *Test your ideas on scrap paper.***

Of course, you don't want to rip the original pages apart; presumably they were put together in that manner for a purpose and they'll need to eventually resume their initial form. Instead, take the pertinent portions of your code and put it on a separate page that you can manipulate at will without changing the original code. I have a separate "test" directory that I use to debug or simply test code to see how it will behave before using it in my code. Jeff Peters, who has the site grokfusebox.com, has an interesting twist on this: he uses my DevNotes in his test directory so he can easily make notes on particular bits of code. Nice touch. You can get DevNotes at www.halhelms.com.

Do you have a tiny bit of code that you want to try out without jumping to another directory? Try saving a file with an underscore as a prefix. When sorted, the under-

scores will rise to the top and you'll know that all these files can be deleted at will.

• **D3.** ***Don't count on luck.***
Let's say you find a nice piece of code that does what you want it to. At least you think it does. If you don't understand how the code works, you're counting on someone else knowing what they're doing. One writer refers to this as "programming with a rabbit's foot." There's nothing wrong with learning from others. That's a key to learning to write better code, only make sure you learn and don't simply lift the code.

• **D4.** ***Have some form of source control, even if it's primitive.***
You may not be outfitted with Microsoft's SourceSafe, but there are open-source alternatives such as CVS (available at www.cvs-home.org). Even something as simple as periodic back ups to a zip file can be invaluable when something really bad happens. And it will happen. Being able to roll back to a working version will prevent you from having to explain to your boss that the application won't be running anytime soon. Bosses tend to frown on this particular form of hubris.

• **D5.** ***Make code files short.***
It just stands to reason that it's easier to debug 20 lines of code than it is 200 or, shudder, 2,000 lines. This means writing modular code, which is much easier to do when you have a methodology. I recommend you look into Fusebox. Now in the third release of the specification, Fusebox has "growed up" and is being used with great success by many developers on everything from small Web sites to very complex applications, see www.fusebox.org.

• **D6.** ***Print out code and study it.***
If you've looked at code till you can recite it from memory, but still can't figure out what's wrong, print it out. Look at it. I've found that oftentimes the mistake seems to leap off the written page. I have no idea why this works, but it does.

"...one developer keeps a stuffed animal on his monitor that he does code reviews with"

• **D7.** ***Have code reviews with others.***
If you really get stuck, consider doing a code review with another person. By this I don't mean asking someone else to try to solve the problem. Instead, walk them through your code, explaining it. If you're like me, you'll find that very often this process will reveal the problem without any input from the other person. In fact, I know one developer who keeps a small stuffed animal on his monitor that he does code reviews with.

• **D8.** ***Use pretested code via code snippets and code templates.***
If you don't use two-dimensional arrays regularly and find that you occasionally have to loop over them, grab that code and put it in a snippet. Better yet, put it in a shared snippet directory so that others can use it too (only if they understand it, obviously!).

• **D9.** ***Use debugging tools.***
I'm not overly fond of Studio's built-in debugger, but I love Dan Switzer's excellent custom tags, Debug.cfm and ObjDump.cfm. These fantastic tools will provide you with a wealth of information and are free! They're available at www.pengoworks.com.

Testing

• **T1.** ***Duplicate the production server environment for your test environment.***
If your production code is going to run in a clustered environment, make sure you test it in one. The same holds true for databases. You need to recognize that the idea of using Access for a development database with the intention of "upsizing" to SQL Server is the song of sirens that want to dash your ship on the rocks of incompatibility. Other common inconsistencies include:
–Amount of server RAM
–Load balancing software/hardware
–Same version of ColdFusion server
–Same version of a Web server

• **T2.** ***Test your code on similar client machines.***
You spent months wearing your boss out until he or she agreed to buy you that fancy 2GHz machine with 512MB RAM and the 19" LCD flat panel. But users of your application may not be so persuasive. So test to see what experience they will have. That means you must know what client machines you're targeting. What is the minimum resolution? What if a user has a very high resolution screen? Will cookies be enabled? Will your user be running an AOL browser? Web TV?

• **T3.** ***Unit test your code.***
In the bad old days I used to wait until the end of a project to begin testing. Now I use a test harness to test each individual file (or fuse in Fusebox terminology) so that when I begin integrating my individual files, I have a high degree of confidence that each individual file is thoroughly tested.
Test harnesses can be as simple as a file that sets some variables needed by the file to be tested, and then includes that file.

```
<cfset productID = "15554">
<cfset productQuantity = "1">
<cfinclude
template="act_addItemToCart.cfm">
```

• **T4.** ***Test for extremes and for unexpected values.***
If you're testing some code to add an item to a shopping cart, what will happen if your user enters 2.55 for a quantity? What if he or she enters the word "one"? What will happen if the number is very large? What happens if the user enters a negative number? Test harnesses can protect you from finding out after the application is deployed.

• **T5.** ***Simulate an environment if you can't replicate it.***

While working on the code for an e-commerce site for one large client, I wanted to test the credit card validation for their own private-label credit card. Astonishingly, the company issuing the card had no ability to test the implementation. After unsuccessfully trying to convince them that this really was not a good idea, I wrote code to simulate their environment so that I could test my processing code.

- **T6. *Write code with testing in mind.***

I write most applications with the knowledge that I will have a Boolean variable called request.testing. When I'm testing the application I turn this on. What happens when this is set to TRUE depends on the application I'm writing. It may simply write to a log file, or it may call a file that simulates a credit card transaction. The point is that knowing that I will have this capability lets me write code to take advantage of it. Also, avoid hard coding values in individual code files. Consider including a file that dynamically sets values for these variables.

"...isolate the problem so you know exactly what you don't understand"

- **T7. *Do daily builds.***

Instead of waiting until all code is integrated, use the concept of stubs – placeholders for files that will actually perform some action. For instance, with a piece of code that includes a file:

```
<cfinclude
template="act_validateUser.cfm">
```

you may not be ready to write the code for act_validateUser.cfm, but you can certainly write this:

```
I am act_validateUser.cfm and
when implemented, I check to
see if the user seeking entry
has a matching record in the
database.
```

- **T8. *Test your returned recordsets with Query Sims.***

Just as you want to test for extremes and unexpected results in your variables, you should do the same with recordsets returned by a database. Query Sims are simulated queries that you can manipulate easily and that will work with any code expecting a recordset. You can download the code for these at www.halhelms.com.

This, at least, is what I have learned in my many years of making mistakes. I hope that you may profit from them. To quote from the father of medicine, Hippocrates, "Life is short and the art long," and we can all use a little help along the way.

ABOUT THE AUTHOR
Hal Helms (www.halhelms.com) is a Team Allaire member who provides both on-site and remote training in ColdFusion and Fusebox.

HAL.HELMS@TEAMALLAIRE.COM

CFXtras

www.cfxtras.com

Where to Go from Here

ColdFusion is not enough

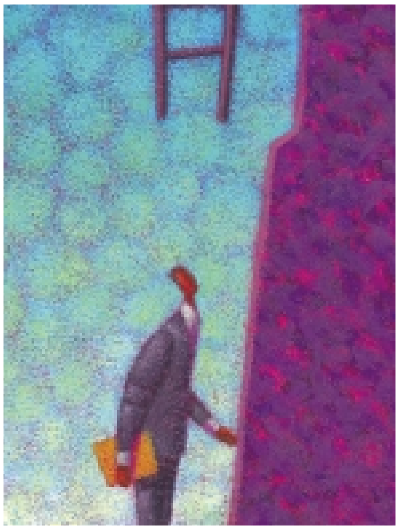
BY
BEN FORTA

ColdFusion 5 is a great product, so much so that I've dedicated six of my last seven columns to introducing and analyzing its new features and technologies. ColdFusion 5 is mature, fast, reliable, and robust, and with its release Macromedia has demonstrated a definite commitment to CF and the former Allaire community. And the community acknowledges this.

How do I know this? I spend a lot of time interacting with developers; between e-mails, meetings, tradeshows, and CFUGs, I have personally spoken to several thousand ColdFusion developers so far this year (and that's all before DevConf). What has hit me is the change in the questions I'm asked and the tones they're asked in. Sure, I still get asked very detailed usage questions all the time – that's not changing, nor would I want it to. What is changing is the higher-level questions and comments. I'm hearing far less concern about whether or not ColdFusion has a future, and far more interest in how ColdFusion can be part of the next wave of technologies and opportunities. I'm not asked as often to defend ColdFusion against competitors, and I am asked far more often how ColdFusion can play a role alongside other products.

The single most interesting question, however, has to be: "What else should I be learning?" I've been asked this question for years now, but the tone and context has changed. That question used to be: "I'm scared that ColdFusion will die and I need a backup plan"; now it's: "What should I concentrate on to become a better ColdFusion developer writing better ColdFusion code?" Same question, very different connotation.

As I'm being asked this very regularly now, I decided to share my list with you.



Master Databases

I don't even remember the last time I saw a CF app that didn't use a database in one form or another. As for the last time I saw an app with a well-designed database, well, I'm not sure I remember that one either.

And no, knowing how to write SQL statements, even good SQL statements, is not enough. I've seen way too many great apps die an ugly death because their underlying databases were a mess.

You need to understand relational databases (both the rules as well as when to break them) to know what your database can (and should) do, to master the more obscure SQL statements and capabilities, and to learn how to optimize and fine-tune your databases regularly. (See my column "Take Your Database Out of Retirement," ***CFDJ***, Vol. 1, issue 3).

Gaining solid database skills is about the most important investment you can make as a ColdFusion developer.

Say Hello to Java

It's no secret. We announced it at last year's conference, and many of us have written and talked about it since then – the future ColdFusion will be built on top of Java. That's a good thing. Jeremy Allaire listed some of the benefits of this new ColdFusion at last year's conference (and you'll hear more about it this year), so I won't repeat them here. You'll have access to all the benefits that the Java community has to offer, coupled with the simplicity and ease-of-use of ColdFusion – it doesn't get much better than that.

So will you have to learn Java? Nope. Not at all. You'll create CFM files, use <CFQUERY> and <CFSET>, write custom tags – do all the things you do now. The fact that Java is running things under the hood is transparent and irrelevant.

I've previously written about using ColdFusion in conjunction with other technologies so I won't go into that now (see "When Not to Use ColdFusion," Vol. 2, issue 3, and "Tiers, Not Tears," Vol. 3, issue 8). There are lots of ways to extend ColdFusion – custom tags and user-defined functions are great for encapsulating reusable CFML code, but to leverage other systems and technologies you need to use extensions that go beyond CFML. There are several choices:

- ***COM:*** Objects may be written in many languages including C/C++, Visual Basic, and Delphi
- ***CORBA***
- ***Java***

Knowing that the future of ColdFusion is built on Java, that last bulleted item becomes very compelling. No, you won't need to know Java to use ColdFusion, but you may want to learn some Java anyway. After all, as Java becomes a core part of ColdFusion, it will also become the obvious mechanism with which to extend ColdFusion.

In other words, brush up on Java basics. Learn what all those J acronyms mean, write some code to experience what's involved, understand what beans are and how they're used, and look at the available Java libraries to see what they can do – the more time spent understanding the world of Java the better. The exercise will pay off, even if you have no intention of becoming a Java programmer.

XML

I won't explain what XML is (or what it isn't). That has been overdone more times than I care to count. Frankly, and at the risk of upsetting some purists, I think XML is terribly overhyped – the way I see it, it's the ultimate data import/export engine, like CSV files on steroids.

XML is not that exciting. But XML is going to facilitate some very exciting things. Web services, distributed processing, intelligent server-to-server communication, seamless data sharing, and that's just the start of it.

"I've seen way too many great apps die an ugly death because their underlying databases were a mess"

ColdFusion doesn't have built-in support for XML processing yet, but there are plenty of third-party options that you can play with. When you start playing, you may even find uses for XML right now (although for most of us, the real fun is still in the not-so-distant future).

And Then...

So far I've listed server-side technologies, as these are the ones that impact CF development the most. But don't ignore the client-side; basic HTML will take you only so far, you should also pay attention to:

- Client-side scripting (JavaScript primarily)
- CSS
- DHTML
- Flash (with ActionScript)

You need them all; the more you know the better.

Conclusion

ColdFusion is middleware – the glue that binds all the bits and pieces that make up an application. To write good CF code, to be a great CF developer, to be able to take CF as far as it can go – you need more than just CF. If you were to ask me where to invest your precious time I'd tell you to master databases, experiment with Java, and become familiar with XML. ColdFusion has a bright future, and armed with this extra knowledge you'll not just be along for the ride, you'll be right at the front of the line.

ABOUT THE AUTHOR
Ben Forta is Macromedia's senior product evangelist and the author of numerous books including the recently published ColdFusion 5 Web Application Construction Kit *and its sequel,* Advanced ColdFusion 5 Development. *For more information on Ben's books visit www.forta.com.*

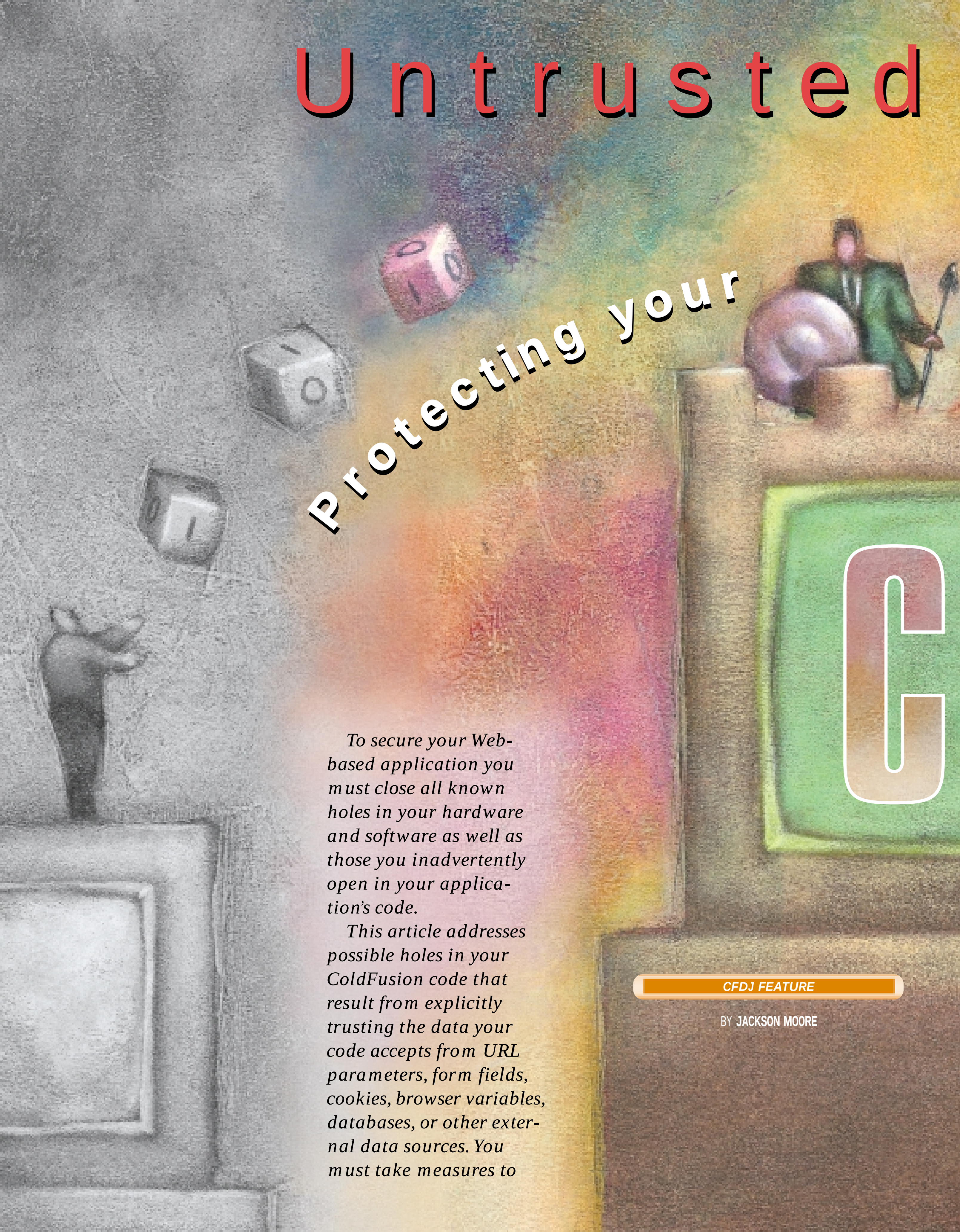
BEN@FORTA.COM

MINERVA
www.minerva.net

Untrusted

Protecting your

C

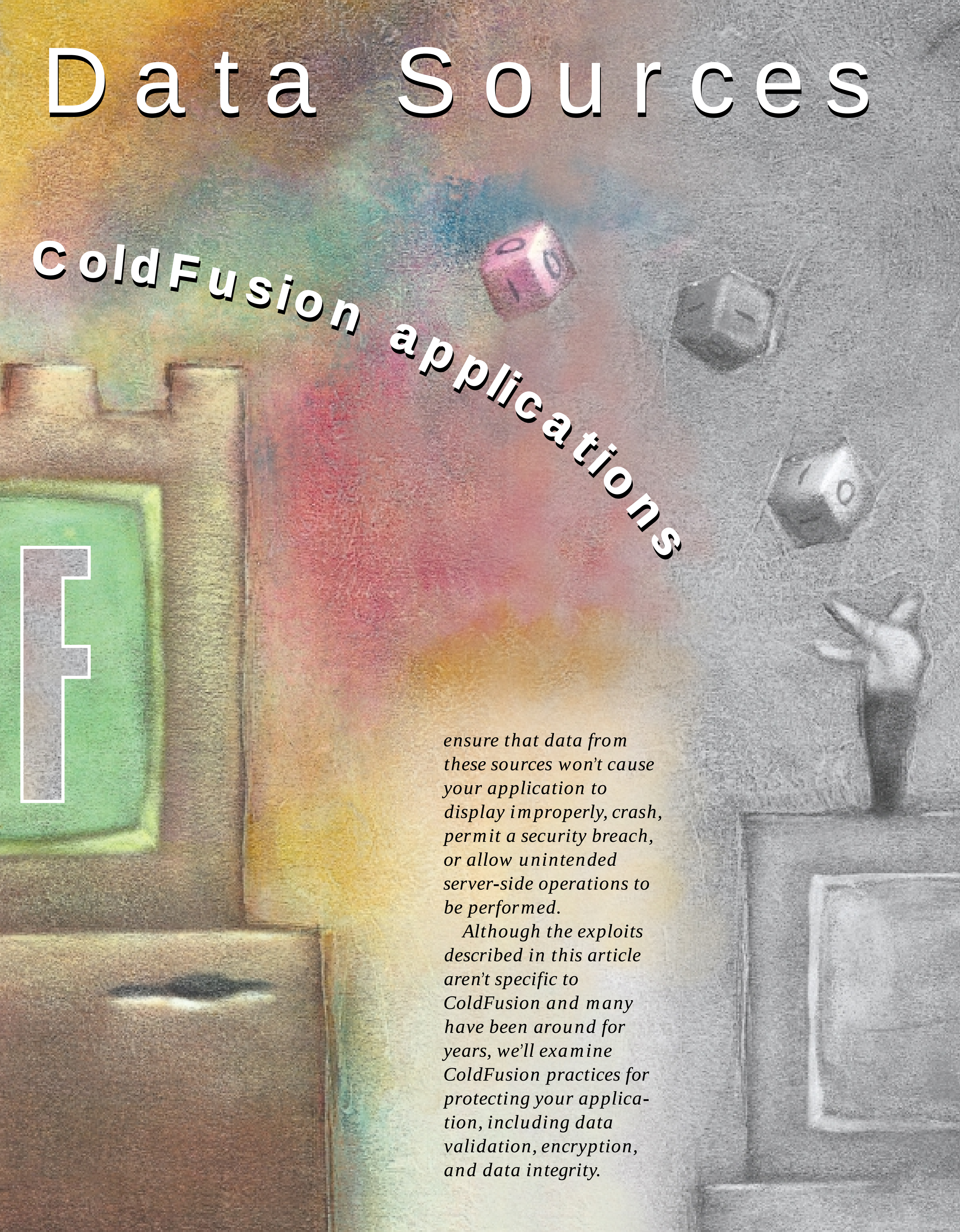


To secure your Web-based application you must close all known holes in your hardware and software as well as those you inadvertently open in your application's code.

This article addresses possible holes in your ColdFusion code that result from explicitly trusting the data your code accepts from URL parameters, form fields, cookies, browser variables, databases, or other external data sources. You must take measures to

CFDJ FEATURE

BY JACKSON MOORE



Data Sources

ColdFusion applications

ensure that data from these sources won't cause your application to display improperly, crash, permit a security breach, or allow unintended server-side operations to be performed.

Although the exploits described in this article aren't specific to ColdFusion and many have been around for years, we'll examine ColdFusion practices for protecting your application, including data validation, encryption, and data integrity.

Untrusted Data Sources

If you conceptualize your application as a black box, any data entering that box should be validated before being processed. For Web-based applications, common sources of input that should be validated include URL parameters, form fields, cookies, browser variables, and databases.

With URLs a user can simply add, change, or delete URL parameters in the query string. Form fields might not be as obvious, but it's a trivial exercise to save a form-based HTML page on your computer, change hidden form fields, or bypass client-side validation and resubmit the form.

Likewise, a cookie can be easily manipulated by using a text editor to make a few careful changes. CGI variables that originate in the browser ("cgi.http_referer") must also be treated as suspect since they can be spoofed or blocked.

Last, and maybe least considered, are the problems that can occur by not validating data retrieved from databases. Unless yours is the only database application, your application can't blindly trust the data in the database. What would happen if another application inserted a username containing a "<" character into the database?

These are the biggest sources of untrusted data, but you must audit your own situation carefully. Other sources of untrusted data are CFFILE, CFHTTP, COM objects, and any other means of accessing data outside the direct control of your application.

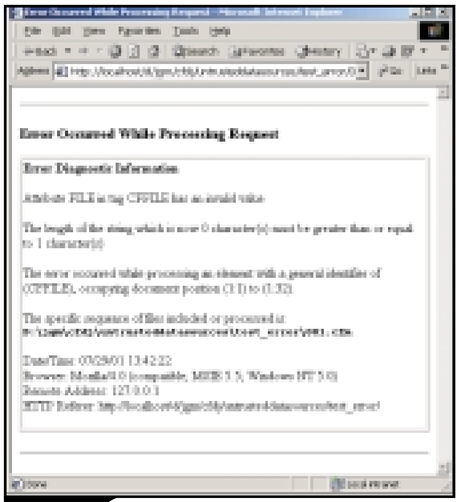
What Can Happen

Improper HTML Display

Perhaps the most benign of holes, allowing user input to affect the display of your page is not only unprofessional, but is a clue to serious hackers that little or no security precautions are in place.

If you have a search form that allows users to search your Web site, you might display a response to the user such as "Your search for 'ColdFusion' resulted in a gazillion matches." But what happens if your user searches for something like "</TD>" or "<B>" or simply a "<"? This may cause your page to display improperly – reformatting your text or omitting portions of your page altogether. Furthermore, some browsers are known to crash when parsing ill-formed HTML.

Improper display can also be caused when pulling data from a database that you don't exclusively control. A database of magazine articles may contain characters that cause display problems if you don't validate the data after you retrieve it from the database.


Error Occurred While Processing Request - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address
Go Links
Done
Local intranet

Error Occurred While Processing Request

Error Diagnostic Information

Attribute FILE in tag CFFILE has an invalid value

The length of the string which is now 0 character(s) must be greater than or equal to 1 character(s)

The error occurred while processing an element with a general identifier of (CFFILE), occupying document position (1:1) to (1:32).

The specific sequence of files included or processed is:
[illegible]

Date/Time: 07/29/01 13:42:22
Browser: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Remote Address: 127.0.0.1
HTTP Referer: [illegible]

FIGURE 1: Error message

Bypassed Client-Side Validation

Since all client-side validation, by definition, takes place on the client, a user can bypass validation code and submit the form with nonvalidated data. Users could have scripting disabled in their browser or bypass the validation code manually after saving the form to their computer.

As mentioned, "cgi.http_referer" (or any other client-side variable) can't be relied upon to verify the origin of a form submission. "cgi.http_referer" can be spoofed to make you believe that the form was submitted from your site or blocked (by a firewall or privacy software), preventing valid users from submitting forms.

Cross-Site Scripting

For this example let's assume you have a message board where users can read and post messages, and a user enters the following:

```
<script>window.open('http://someporn-
site/');</script>
```

If you're not validating this text field, every user that views your message board will be "treated" to a new window that originates from and appears to be authorized by your site. Other potentially damaging HTML tags include <OBJECT> and <APPLET>.

Error Messages

Serious attackers will look for any information about your server configuration that allows them to identify potential holes. Those unfriendly ColdFusion error messages (see Figure 1) that are so helpful when debugging an application are a gold mine to a hacker looking to have some fun.

As you can see from this figure, the user is presented with an error message revealing a physical path on the server.

The amount of critical information displayed will depend on the type of error, your ColdFusion server settings, server platform, and database server.

Unauthorized Access

Even if you employ an authentication framework, you must still validate data received from the client to prevent unauthorized access to restricted areas of your Web site. Consider a situation where you design a Web site for a magazine that wants to restrict some of the articles to subscribers only.

- ***Case 1:*** `<a href="display.cfm? article_id=14">Cover Story</a>`
- ***Case 2:*** `<a href="articles/2001/july/cover.htm">Cover Story</a>`

A curious user might try different values for "article_id" or different paths to see what will happen. Will your users gain unauthorized access by guessing the right value or path?

Unintended Server Operations

Holes that allow unauthorized server-side operations to be performed are the most disastrous. Some databases, for example, support multiple SQL statements inside a CFQUERY. Building on the magazine example we might construct the following query:

```
<cfquery name="qArticles" data-
source="testDB">
    SELECT article_copy FROM articles
    WHERE article_id =
#url.article_id#
</cfquery>
```

This works fine if the URLs submitted by the user have valid article numbers, but not URLs like this:

```
http://magazine/display.cfm?article_i
d=14%20delete%20from%20articles
```

With Microsoft SQL Server (among others) this user just deleted every record in the "articles" table. Granted, the user must know the name of your table, but through the error messages mentioned earlier or simply guessing common table names, this isn't difficult.

You need to analyze your other server software for similar "features." For example, Allaire Security Bulletin ASB99-09 (www.allaire.com/handlers/index.cfm?id=11069&method=full) warns of an issue with Microsoft Access that allows users to append VBA commands to a SQL string.

Any server software your application interfaces with that supports scripting needs to be audited for related patches. If you validate all user input, however, you're likely to avoid these vulnerabilities.

YES SOFTWARE

www.codecharge.com

What You Can Do

ColdFusion Validation

ColdFusion includes two built-in methods for validating data received from form data: server-side validation using hidden <INPUT> tags and client-side validation using CFINPUT. Both methods are designed for validating form data, but, as you know, form data is just one of the many untrusted data sources you must contend with.

The built-in validation options are somewhat limited, though, so you'll probably need to write some of your own validation code. If you utilize the server-side method, be sure to use CFERROR to specify a more presentable validation error-handling template. Last, since the validation instructions (server-side) and JavaScript functions (client-side) are sent to the client with the form, they can be bypassed.

That said, these techniques are useful in some cases and provide built-in validation for required fields, data types such as integer and date, as well as data values such as credit card and social security numbers. The online help provides documentation and examples for each method.

Use Server-Side Validation

Many of the techniques for data validation can be performed on the client with JavaScript. Client-side validation is unreliable, though, because this scripting may be disabled, bypassed, or the user's browser may not support the version of JavaScript you need.

Furthermore, client-side validation can't perform some data validation such as preventing duplicate database records or testing uploaded files. The advantage to client-side scripting is that it provides immediate feedback to the user without a round-trip to the server for validation.

I advocate using client-side scripting to check for required fields only, since you must validate the data on the server anyway. The advantages to this approach are that the required JavaScript implementation is small and you won't have to maintain two libraries of validation routines.

Checking for required fields can be easily implemented on the client-side through the CFINPUT tag without any knowledge of JavaScript. To make a field required, specify the "required" attribute:

```
<cfinput type="text" name="firstname"
required="yes">
```

ColdFusion will supply the necessary JavaScript with the page returned to the browser. If the user tries to submit the form and leaves the "firstname" field blank, a JavaScript alert box will pop up to inform the user that this field is required (note that spaces are considered valid characters). Again, this assumes that JavaScript is enabled on the user's browser and hasn't been bypassed.

Scope Variables

If you don't manually scope your variables, ColdFusion looks for them in various scopes in the following order: local variables, CGI, file, URL, form, cookie, and client variables. If you don't scope variables in your code, you won't know whether they're coming from the URL, a form, or a cookie. Scoping your variables will also make your code easier to follow and give you a slight performance boost.

Error Handling

To prevent the display of error messages that may reveal information about your server configuration, use structured error handling (with CFTRY/CFCATCH blocks) and/or site-wide error handlers (CFERROR). This will allow you to trap any errors and continue processing your page accordingly.

Disabling the "Display the template path in error messages" setting in the ColdFusion Administrator will prevent physical paths from being displayed in most, but not all (see Figure 1), error messages that do get through to the user.

For more information on handling errors look through your back issues of ***CFDJ*** for the "Toward Better Error Handling" series (Vol. 2, issues 10 and 12, and Vol. 3, issue 2) by Charles Arehart. These are good articles that will help even beginners get up to speed on error handling.

Data Type Validation

Perhaps the single most important validation to perform is to verify the data type of suspicious data – making sure, for example, that a variable that's supposed to be a number is actually a number. This ensures that all subsequent processing of this type of data won't throw an error by performing a date operation on a nondate value or by trying to insert a nonnumeric value into a numeric database column.

Two useful tags for this purpose are CFPARAM and CFQUERYPARAM. With CFPARAM, specify the "type" attribute with any of the following data types: array, binary, boolean, date, numeric, query, string, struct, or uuid.

```
<cfparam name="url.myValue"
type="numeric">
```

If the value of "url.myValue" is anything other than numeric, ColdFusion will throw an error. Using this in a meaningful way will require the use of CFTRY/CFCATCH blocks to catch the error and handle it properly.

"Those unfriendly ColdFusion error messages that are so helpful when debugging an application are a gold mine to a hacker looking to have some fun"

CFQUERYPARAM tags are nested within CFQUERY tags and will give you some control over data validation. Refer to the ColdFusion documentation for CFQUERYPARAM for a list of SQL data types that can be validated. Again, ColdFusion throws an error if the validation fails.

ColdFusion also has several built-in functions (IsDate(), IsNumeric(), etc.) that allow you to determine data types without using error handling. If you need more specific data-type validations (such as integer or hexadecimal), you'll have to write your own validation code.

Another helpful ColdFusion function for validating numeric data types is Val(). Val() will return a number from the beginning of a string or 0 if the string can't be converted to a number. This function is particularly handy when dealing with a peculiar aspect of ColdFusion's support for scientific notation.

ColdFusion treats the value "1D2" the same as "1E2" – both are scientific notations for "100". ColdFusion considers both values to be valid numeric values, but your database probably won't even if it supports scientific notation. Using the Val() function will ensure that the value is converted to the more recognized scientific notation format:

```
#val("1d2")#    <!--- 100 --->
#val("1d13")#   <!--- 1E+013 --->
```

For example, if your user enters his or her age into a form as "3D1" (30), you want to make sure that inserting this value into your database won't cause an error even though this is a valid number in ColdFusion.

Data Value Validation

While data-type validation should prevent errors in subsequent ColdFusion code, it doesn't ensure that the value of the data is acceptable. Consider again our example of the magazine Web site. Some articles are available to the general public, while others are available only to subscribers.

```
http://magazine/display.cfm?article_id=14
```

If the user manually changes “article_id” to a different (but valid) integer, should the user be presented with the article carte blanche? In this example the answer is no. For each article the user requests, we must authenticate and verify that he or she is allowed to view the requested article. You can’t assume that users won’t try to guess new paths or values for URL parameters, form fields, or cookies in order to access restricted portions of your Web site.

Another case where data value validation is critical is when the range of legal data values for a data type in ColdFusion differs from your database. ColdFusion, for example, considers dates as large as December 31, 9999, as valid dates. In SQL Server 7, the “datetime” data type will accept this value, but the “smalldatetime” data type won’t. To prevent database errors with date data types, you need to ensure that the date value is within an acceptable range for your database column.

There are many other reasons why you need to validate the value of a piece of data, not just its data type. Preventing duplicate records in a database or making sure the length of a string value is within acceptable minimum and maximum lengths are additional examples of when the value of the data must be validated. The bottom line is that (just like data types) you can’t make any assumptions about the value of data received from a client.

Defining Character Sets

Validating string data presents its own set of unique challenges since there are many special characters reserved by ColdFusion, URLs, HTML, and database software. Because ColdFusion doesn’t have any built-in validation functionality like Perl’s taint-checking mechanism, it’s the developer’s responsibility to perform all necessary string validations.

In nearly all cases you’ll protect yourself better by first defining which characters to allow and then making sure that the suspect string contains only those characters. This is preferable to defining which characters are illegal (though this may seem easier to implement), because it’s very easy to forget one and it reduces your vulnerability to (as yet) unknown exploits.

As an example, let’s consider a form field that accepts a username field and requires all usernames to contain only letters. It’s safer to verify that only letters are submitted as opposed to checking for all possible illegal characters. You can use a regular expression to verify that no other character is allowed:

```
<cfif refindnocase("[^a-z]",form.username) neq 0>
   <!--- invalid username --->
</cfif>
```

The validation becomes more complicated for string data that has to be more flexible. Our example of the cross-site scripting exploit was due to a developer allowing <SCRIPT> tags in message board posts. This particular example could have been avoided by defining a character set that excluded the “<” and “>” characters. But what if you want to allow users to use <B>, <I>, and <U> for formatting purposes within their message post? Now you have to allow some tags but not others.

As with defining single character sets, you’re far safer determining which HTML tags to allow as opposed to identifying the tags that may potentially cause problems. I take issue with some of the available custom tags that focus on which HTML tags to disallow because future additions to the HTML specification or browser-specific tags may require you to revisit your code.

If you want to allow bold, italic, and underline formatting in your message posts, allow only those and disallow all others. The following code will remove all tags except for <B></B>, <I></I>, and <U></U>:

```
<cfset form.message=rereplacenocase(form.mes-
sage,"(<[/]?[^biu]>)|(<[^/]([^>]+)>)|(</([^>]{2,})>)","",
"all")>
```

This, of course, won’t validate the HTML to ensure it’s well formed. Limiting the allowable HTML tags will minimize the amount of validation needed since you don’t want to have errors caused by a user who failed to provide an end tag or improperly nested certain tags. If the user leaves out a “</B>” tag, you won’t have a problem if the message post is encapsulated in a table cell. However, a misplaced <DIV> or </TD> can really wreak havoc on the display of your page or crash the browser.

PACIFIC ONLINE

www.paconline.net

Encryption

Another approach is to encrypt the data that appears in URLs, form fields, and cookies. Consider the earlier example of the magazine Web site: if we encrypt "article_id" our link might look like:

```
  http://magazine/display.cfm?arti-
cle_id=0101J442509E78541
```

The new value of "article_id" makes it less obvious to the user how "article_id" is used. Curious users may still tamper with the value, but after failing once or twice will give up.

ColdFusion offers two complementary functions, Encrypt() and Decrypt(), to perform encryption. They both take a string to encrypt as well as a key to use in the encryption process.

Encrypt()'s XOR-based encryption, though useful, is weak compared to encryption methods (such as that used in 128-bit SSL) considered secure by today's standards. Encrypt() shouldn't be used for sensitive information (credit card numbers or medical information), but it can be used to prevent most attempts to casually change "nonsensitive" values in URLs, form fields, and cookies.

Strings encrypted with Encrypt() can contain many potentially damaging characters (new lines, form feeds, backspaces, greater-than signs, carriage returns, etc.):

```
encrypt("Macromedia","key") = *=(IX
0B@5M%$V#\!
```

The last character (which you can't see) is a new line character. Practical use of the Encrypt() function will generally require another encoding step such as URLEncodedFormat() to escape all illegal characters in the string. Your decryption process then becomes a two-step process as well – decode and decrypt.

Another drawback to the Encrypt() function is that there's no data integrity check to ensure that the encrypted string hasn't been tampered with. Using the above example I decrypted the encrypted string, but changed the number 5 to 2:

```
decrypt("*=(IX 0B@2M%$V#\! ","key") =
Macromydia
```

As you can see, the decrypted string was successfully decrypted but resulted in a value different from the original.

At the risk of losing some objectivity, I'll mention a tag I wrote called CF_CRYP (available from the Developers Exchange, http://devex.allaire.com/developer/gallery/). CF_CRYP builds on the default encryption of the Encrypt() function, but encodes the resulting string and adds a checksum.

"You can't make any assumptions about the value of data received from a client"

The result is an encrypted string containing only numbers and letters with a checksum. Changing a single character in a CF_CRYP-encrypted string can be detected because the checksum of the decrypted value won't match the checksum of the original unencrypted value. CF_CRYP also provides a return structure with error information so you can detect tampering and, potentially, block that user from further access.

Using CF_CRYP to encrypt a value of "14" with a key of "somekey" produces "0101J442509E78541". To access the subscriber-only articles, the user would have to know the encoding scheme, key, and recalculate the checksum. This is a good solution for this example because if the encryption scheme was broken, the worst that can happen is an unauthorized user views some subscriber-only articles. If this happens, you should probably give the user a free subscription to your magazine anyway!

Use CFCONTENT

CFCONTENT will allow you to place content outside of your Web root directory and still send the content to a user. This is preferable when the files (PDF documents, video clips, etc.) require some type of user authentication to access. Since these files exist on your server outside of the Web root, users can't access the file with any URL. To view the file they have to access a page where you should authenticate them and use CFCONTENT to deliver the file.

Database Validation

Since many of the most damaging server-side operations are unauthorized database operations, your database authentication framework should be audited. Carefully review the operations that users need to be able to perform and then limit the SQL operations that can be performed under that user account. Generally speaking, regular users should be restricted to SELECT statements and, in some cases, UPDATE and INSERT.

Even if a user manages to get multiple commands through to your server, only those operations authorized for this user would be allowed. You must combine this type of security with the other guidelines in this article since even a benign SELECT operation can allow a user to view unauthorized content. If you use other server-side applications (including COM objects), take advantage of any built-in user authentication.

Most databases will also allow you to perform data validation at the database level. This is a good solution when you have numerous applications inserting data into a database. By moving some data validation to the database, you can remove redundant validation routines from each application.

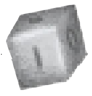
Conclusion

Most users are honest people using your application for the purpose it was designed for. However, a small group of users (some armed with homegrown HTTP clients) will attempt to exploit any holes that exist in your code. Form fields, cookies, and browser variables are just as likely as URL parameters (maybe more so since they're less obvious to inexperienced developers) to contain hack attempts.

Taken together, the techniques presented in this article will help you protect your application from any malicious data sent by a user. Whether the data is tampered with accidentally or on purpose, it's your job as the developer to close these code-level holes.

Resources

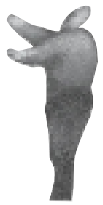
1. "Security Best Practice: Validating Browser Input": www.allaire.com/handlers/index.cfm?id=14558&method=full
2. "Cross-Site Scripting Vulnerability Information": www.allaire.com/handlers/index.cfm?id=14557&method=full
3. "Multiple SQL Statements in Dynamic Queries":www.allaire.com/handlers/index. cfm?id=8728&method=full
4. "Understanding Malicious Content Mitigation for Web Developers": www.cert.org/tech_tips/malicious_code_mitigation.html



About the Author

Jackson Moore is a self-employed Web application developer in Clearwater Beach, Florida. A Certified ColdFusion developer and engineering graduate of the University of Louisville, Jackson has been providing Web development services for the last five years and now specializes in ColdFusion and Java.

jackson@iology.com

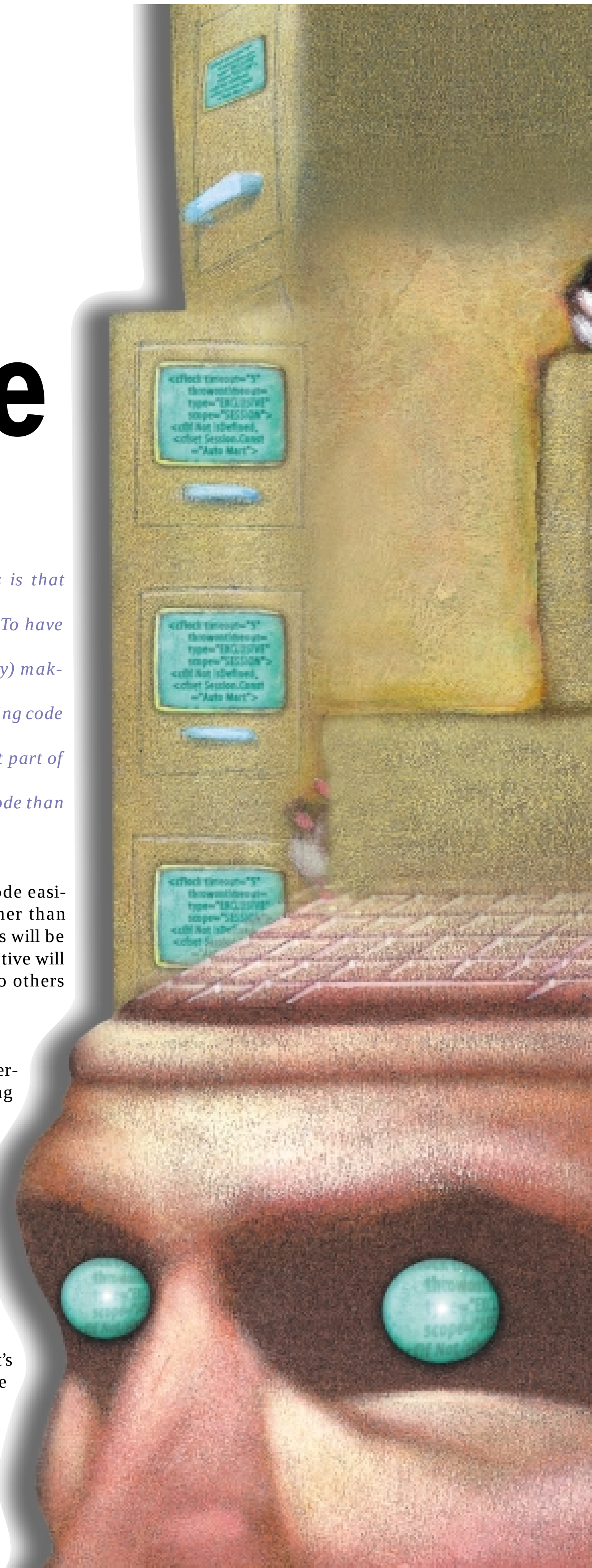


CFDJ FEATURE

BY **RAYMOND H. THOMPSON**

Coding for Maintenance

One of my philosophies when dealing with computers is that they're generally inexpensive in comparison to people. To have a warm body (and most developers fall into this category) making modifications to code can be expensive. I've always felt that making code easy to maintain, if not for myself at least for others, is an important part of any project. In most cases more time will be spent maintaining the code than in the initial development.



What I hope to show here are some examples of how to make code easier to maintain. Some of the ideas can be applied to languages other than ColdFusion, but since ***CFDJ*** is a ColdFusion magazine, the examples will be specific to that language. Writing code from a maintenance perspective will make life easier for the developer and will endear that developer to others who have to maintain the code.

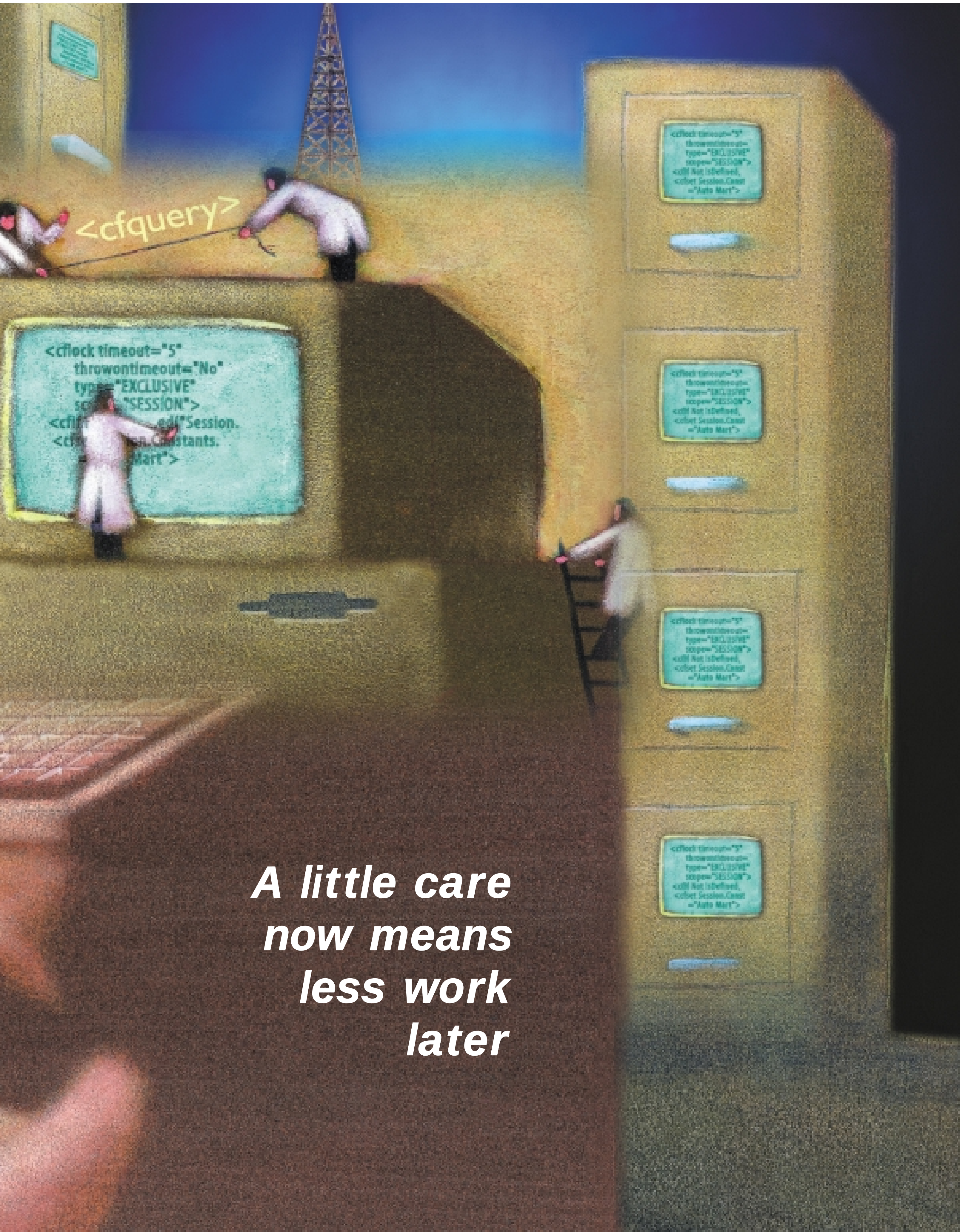
ColdFusion Limits

Since ColdFusion is an interpreted language there are some differences between what it offers and what other languages offer. One thing I miss is the ability to declare constants. Languages such as Delphi provide for an explicit constant section where values that can't be changed can be specified. This was a good thing as it provided information about values to the compiler during the compilation phase. The compiler used these values when needed and the constants didn't take up any memory in the resulting executable file.

Since ColdFusion is interpreted, the advantage of constants is lost from a memory and interpreter perspective. The information still has to be parsed, stored somewhere in memory, and then used within the script. So there's no real advantage to having a constant section within ColdFusion when dealing with performance.

In ColdFusion any variable can be used as a constant value. What's lost is that any variable used as a constant can have its value changed. That can make for some nasty bugs.

But that doesn't mean constants can't be used. Some care just has to be taken when dealing with values that will be used as constants. Prefacing the variable name with something like a "C" for constant value would allow these variables to be recognized easily.


<cfquery>
<cflock timeout="5"
throwontimeout="No"
type="EXCLUSIVE"
scope="SESSION">
<cfif ... ed("Session.
<cf ... n.Constants.
... Mart">
A little care
now means
less work
later

Suppose some variables were set in the following manner:

```
<cfset CYearlyInterest=0.0965>
<cfset CMonthsInYear=12>
<cfset CMonthyInterest =
CYearlyInterest / CmonthsInYear>
```

It would be immediately apparent that the values were constants and shouldn't be changed after the initial definition. Of course, there's nothing to prevent these values from being changed by some errant code. For that reason I'd like to see a constant section in ColdFusion. A better alternative would be a tag specific to constants. Once the value was set, it couldn't be changed. This "Set Once, Use Many" type of variable could be along the lines of the following:

```
<cfconstant CYearlyInterest=0.0965>
<cfconstant CMonthsInYear=12>
<cfconstant CMonthyInterest =
CYearlyInterest / CmonthsInYear>
```

But ColdFusion doesn't have this capability, so it's just something that has to be dealt with.

Working with Constants

Why would constants be used if there's really no advantage to performance or memory use? The answer is that using constants makes code easier to maintain.

As a general rule I like to use a constant when that value is used for the same purpose more than once within a script or set of scripts. If a change needs to be made to the scripts, it's sometimes difficult to remember where the value is used. Miss one, and extra time will be expended tracking the problem.

Let's look at some examples and the concept will become clearer.

The most obvious example for many scripts is the name of the data source used for accessing databases. The data source name is defined in the ColdFusion administrator and becomes a required parameter for the CFQUERY tag.

```
<cfquery Name="SomeQuery" DSN="DB1">
```

If this DSN is referenced in multiple places in the scripts, the value shouldn't be hard-coded in the CFQUERY tag. It's better to substitute a variable that contains the name of the data source.

```
<cfset CDSN="DB1">
.
.
.
<cfquery Name="SomeQuery"
DSN="#CDSN#">
```

"The thing to remember is to make the code 'self-aware' so the proper values can be set"

Notice that the variable name must be surrounded by pound signs because the data is contained within quotes. You can skip the quotes and the pound signs and achieve the same results. But the editor in the CFSTUDIO will place quotes around the value when the tag is modified using the tag edit function in CFSTUDIO.

What's the advantage of using a variable for the data source name? What if the data source name needs to be changed for any reason? The challenge of looking through the scripts and making the replacement would be tedious and one or more places that need to be changed might be missed.

Another advantage is that it would be possible to change the name of the data source depending on a number of variables. For example, if you have a development system and a production system, the data source name may not be the same in both environments. It's possible for the code to determine which data source to use and set the proper variable. In my opinion, this is best done in the "Application.cfm" script.

```
<cfif CGI.HostName EQ "DEVL">
  <cfset CDSN="DB1">
<cfelse>
  <cfset CDSN="DB2">
</cfif>
```

Any of a number of variables could be used to determine the environment, such as the script directory name and whether secure sockets are being used. The thing to remember is to make the code "self aware" so the proper values can be set.

This concept can be extended to screen colors, frame widths, screen widths, table width, and so on. Any value used more than once that represents the same purpose should be considered for use as a constant.

If a lot of constants are created, it would be best to store the information in some type of structure. It would also be advisable to create SESSION-scoped structures to store the constants between requests. If a SESSION structure is used, it should be copied to a REQUEST structure before being used in the scripts. This avoids contention and the need for locks when referencing the variables. If the SESSION scope is used, then CFLOCK would be required every place one of the constant values is used. Using the previous example for the DSN, Listing 1 illustrates this concept.

All code that needs to reference these constants would then use "Request.Constants.-CDSN" or whatever is used for the name of the variable. If any of these values are changed during the course of processing the script, all the REQUEST-scoped structure can be copied into a SESSION-scoped structure when the script completes. Here at last is a good use for the "OnRequestEnd.cfm" script that is run at the end of a page request.

CFINCLUDE and CFMODULE

Another method to make code easier to maintain is to use included code, that is, the CFINCLUDE and CFMODULE tags. The main difference between the tags is that CFINCLUDE cannot pass parameters where CFMODULE can pass them. There's no logical difference between CFMODULE and custom tags except how the location of the code is resolved. I prefer CFMODULE, as the script location can be controlled to a much finer level. There's extra overhead processing with CFMODULE and custom tags as a separate environment is established within the server. CFINCLUDE is probably a little faster.

A good use for CFMODULE and CFINCLUDE is in screen headings. As the look and feel of the site may well be the same from page to page, the heading information is probably constant for the most part. Parts that aren't constant can be passed as parameters.

For example, assume that the title, logo, and other elements are the same for the screen. The only thing that changes is the name of the screen. A call to a script using CFMODULE would look like the following:

```
<cfmodule template="Header.cfm"
Title="Entry Screen">
```

And the module referenced in the above may look like Listing 2.

Note Listing 2's use of constants for several of the parameters for screen formatting. If the width or font color needs to be adjusted, it's a simple matter to change the single script where the values are set.

Intelligent Code

The next thing to consider is how to make code intelligent. Basically, this means creating code that can determine as much as possible about items the code is working with so that changes become fairly simple.

MACROMEDIA

www.macromedia.com/go/devcon01

Let's consider a couple of fairly common examples of this concept.

The first example will deal with strings that need to be split. I've seen much code that requires some input from a user screen where the input from a single field must be split into two values. If you're asking for a credit card number and you want to do some host validation, you really need the first four digits to determine the card type. The last digits determine the account number.

Code in the following format will work for card numbers of 16 digits:

```
<cfset CC=Trim(Form.Card)>
<cfset Prefix=Left(CC,4)>
<cfset Account=Right(CC,12)>
```

That works really well as long as the length of the card number doesn't change. You could add code that would check the length and use different code for the different lengths, but there's an easier way, using standard functions in ColdFusion.

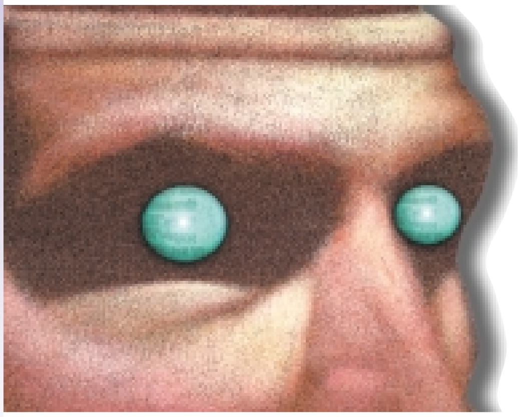
The following code will work for all lengths of card numbers. What's really key is that the code will work with future numbers if some card should ever use some oddball length.

```
<cfset PreFix=4>
<cfset CC=Trim(Form.Card)>
<cfset Prefix=Left(CC,PreFix)>
<cfset Account=Right(CC,Length(CC)-
PreFix)>
```

The changes are simple, but they make the code intelligent so any length card number will work.

Note also that the data returned from the form is trimmed of leading and trailing spaces before it's used. This should be standard on all form fields returned to the server. It makes the form more user-friendly and allows the user more freedom in entering data.

As another example, consider the process of color-coding alternating rows of a table. The standard method (or variations using CFIF) looks like the following:



```
<tr bgcolor="#IIf(RowCount Mod 2) EQ
0,DE("White"),DE("Yellow"))#"></tr>
```

This certainly works…and works well. But what happens if the colors have to be changed? Several places in the scripts may well have to be changed to support the different colors.

Where this really gets messy is if the number of alternating colors changes. What if the requirement is for three or four different colors? The coding to deal with more than two colors can be fairly unwieldy.

A simpler solution is to make the code intelligent and able to deal with any number of colors for the rows. This is easily done using standard ColdFusion functions. These functions extract information about the variables and use that information to build the rows properly, as seen in Listing 3.

What this code does is extract the length of the list, divide that into the row count, and then get the remainder. The remainder will be between zero and the length of the list minus one. One is added to this value to get a list element of one through the length of the list.

If a third color were added, the code that does the color-coding wouldn't have to be changed. In fact, code, such as this, would handle any number of colors without modification.

These examples show how it's possible to make code that works in changing environments without modification. Seeking ways to make code more "aware" of the processing parameters and to make decisions about that environment makes maintenance easier.

Intelligent Forms

Another area that sometimes causes maintenance problems arises when you're dealing with elements in forms. What can be so difficult in processing forms that coding for maintenance would be an issue? A couple of examples should help to illustrate this concept.

Let's start with the select list that allows multiple selections and the items that can be selected may contain commas. There's a conflict as multiple selections from a select list are returned as a single form field. Commas separate the selections in the list. But since the data could also have commas, errors could result when the list is processed.

```
<select name="Songs" multiple>
  <cfoutput query="GetSongs">
    <option value="#SongName#|">
      #SongName#
    </option>
  </cfoutput>
</select>
```

To avoid that problem an extra character (in this case the vertical bar "|") is added to the end of each item in the list. The bar can then become the list separator. All the receiving code has to do is remove the trailing vertical bar and process the items as a list, using a vertical bar as the list delimiter.

```
<cfset Songs=Trim(Form.Songs)>
<cfif Len(Songs) GT 0>
  <cfset Songs=
    Left(Songs,Len(Songs)-1)>
  <cfloop index="Song"
          list="#Form.Songs#"
          delimiters="|">
    #Song#<br>
  </cfloop>
</cfif>
```

So far this is fairly straightforward, simple processing. Now consider what would happen if the list delimiter needs to be changed. You'd have to find all the code that used that list and change the delimiter. It may become a tedious process and prone to errors if some code in one of the processing forms is missed.

The solution is to have the form that establishes the data also provide information about the delimiter used for the list. Defining a constant used for the delimiter, and then placing a hidden field in the form that contains the value of the constant, enables the value of the delimiter's to be passed to the receiving form (see Listing 4).

Once the receiving form gets the form fields, it has all the information to process the list. No hard-coded information about the delimiter is required. Changing the delimiter would require only one change in one form. Subsequent forms using the list wouldn't need any modifications.

```
<cfset Songs=Trim(Form.Songs)>
<cfif Len(Songs) GT 0>
  <cfset Songs=
    Left(Songs,Len(Songs)-1)>
  <cfloop index="Song"
          list="#Form.Songs#"

delimiters="#Form.ListDelim#">
    #Song#<br>
  </cfloop>
</cfif>
```

The minor changes involved made the code more "intelligent" and more aware of the data.

Another case in which the sending form should provide information is when dealing with multiple rows of data. Since the sending form is able to create the rows of data, the form knows the number of rows. Passing this data to the next form in

MACROMEDIA

www.macromedia.com/go/usergroups

NEW
Forte™ for Java™ 3.0 tools from Sun. Review: page 56
JAVA DEVELOPER'S JOURNAL
DoJa
DoCoMo Phones
FREE Download Forte™ for Java™ 3.0 tools
Sun

a hidden variable provides the receiving form with valuable information that can be used when processing the data.

The point is that the process, or form, that creates the data needs to provide enough information for subsequent forms to process the data.

Processing Forms

When processing forms, several things should be considered to make them as rugged as possible, which also reduces the maintenance involved.

Form fields with user input should be trimmed using the TRIM() function before being used. One exception is if a blank is a valid value for a field. Trimming will remove leading and trailing spaces from a form field, possibly preventing the user from entering invalid data. For example, when entering a person's name, a leading blank is probably invalid.

All date fields input by the user need to be validated (after trimming) to ascertain that the field contains a valid date. The form field should then be converted into a true date/time field using the ParseDateTime() function.

When processing user input for a query, always use a temporary variable if using a function (such as TRIM) against the field. This allows any single quotes entered by the user to be properly escaped by ColdFusion.

A problem with ColdFusion arises when you use a text field in a query where the data contains single quotes. If a function is used inside the query, single quotes are not escaped. If a field is used inside a query, then they are.

The following code won't work if a single quote is present in the value in the WHERE clause of the query.

```
<cfquery .......>
  select something
  from sometable
  where

somefield='#Trim(Form.Song)#'
</cfquery>
```

The following code will work. The only difference is the use of a temporary variable.

```
<cfset TSong=Trim(Form.Song)>
<cfquery .......>
  select something
  from sometable
  where somefield='#TSong#'
</cfquery>
```

This flaw has been in ColdFusion for a long time. Fixing it might break a lot of other code that does explicit single-quote escaping within a query using the Replace() function. The code's inconsistent behavior is a problem that needs to be addressed.

Nothing Is Free

An argument could be made that writing code that allows for multiple values and changes in the values, code that plants hidden variables to be passed to the next form, and so on, may cause some performance issues. There's probably some small overhead to handle the extra coding involved.

Look at this from a different perspective. Computers are supposed to be intelligent. The power of a computer lies in its decision-making process and in its ability to act on those decisions. Placing code in a script that is intelligent and can process variable data makes the computer do what it's designed to accomplish.

Overall, I suspect the additional overhead is very small. The impact on the script's operation probably wouldn't be measurable unless some type of loop were done hundreds of times.

Summary

These techniques can be applied to many different types of code and applications.

Using simulated constants in a script for values representing the same function simplifies the changing of those values. ColdFusion provides no explicit constant, so regular variables can be used.

Making the code intelligent in how it handles data allows the code to deal with changes readily. Using standard CF functions to obtain knowledge about data makes the code more intelligent.

Creating forms that provide all the necessary information about how to process the data isolates changes to singular locations.

A little thought and code intelligence results in code that's easier to maintain.

About the Author

Raymond Thompson is employed by Q Systems, a leading developer of Web-based applications in Oak Ridge, Tennessee. Ray has over 30 years of experience in information systems and has been involved in many aspects of systems design, deployment, and management, ranging from "big iron" to desktops.

rayt@qsystems.net

Listing 1

```
<cflock timeout="5"
        throwontimeout="No"
        type="EXCLUSIVE"
        scope="SESSION">
 <cfif Not IsDefined("Session.Constants")>
  <cfset Session.Constants.AppName
         ="Auto Mart">
  <cfif CGI.HostName EQ "DEVL">
   <cfset Session.Constants.CDSN
          ="DB1">
  <cfelse>
   <cfset Session.Constants.CDSN
          ="DB2">
  </cfif>
 </cfif>
 <cfset Request.Constants
    =StructCopy(Session.Constants)>
</cflock>
```

Listing 2

```
<cfparam name="Attributes.Title" default="">
<cfoutput>
<table width="#Request.Constants.ScreenWidth#">
  <tr>
    <td align="CENTER" valign="BOTTOM"><font size="5"
color="#Request.Constants.FontColor#">#Request.Constants.Ap
plicationName#</font></td>
  </tr>
  <tr>
    <td align="CENTER" valign="TOP"><font size="4"
color="#Request.Constants.FontColor#">#Attributes.Title#</f
ont></td>
  </tr>
</table>
<hr width="#Request.Constants.ScreenWidth#">
</cfoutput>
<br>
```

Listing 3

```
<cfset CRowColors="White,Yellow">
<cfset RowCount=0>
<table>
  <cfoutput Query="Q1">
    <cfset RowCount=RowCount+1>
    <tr bgcolor="#ListGetAt(CRowColors,RowCount MOD
Length(CRowColors) + 1)#">
      <td>Column1</td>
      <td>Column2</td>
    </tr>
  </cfoutput>
</table>
```

Listing 4

```
<cfset CListDelim="|">
<input type="hidden"
       name="ListDelim"
       value="#CListDelim#">
<select name="Songs" multiple>
  <cfoutput query="GetSongs">
    <option
     value="#SongName##CListDelim#">
      #SongName#
    </option>
  </cfoutput>
</select>
```

CODE LISTING

The code listing for this article is also located at

www.sys-con.com/coldfusion/sourcec.cfm

ASSETNOW

www.assetnow.com

Using MS-SQL Stored Procedures with ColdFusion Part 1 of 2

Easier than you've imagined – if you're meticulous

BY IAN RUTHERFORD

Stored procedures. Ah, yes. The dark side of database access. You've heard the rumors – faster processing, less network traffic – and have been tempted. But you've also heard those on the easy path whisper about how complex these procedures are...

...and how much of a pain it is to get them to work. So which is right? As with most day-to-day problems – both. However, I'll show you that stored procedures are worth the hassle, and in the end aren't difficult to write – if you're meticulous. Debug? That's a completely different matter. This article focuses on Microsoft SQL Server because Oracle is an even darker path that I dare not tread, and SQL Server has handy tools to help you in your efforts. (The principles, if not the exact code, will hold true no matter which database you use.)

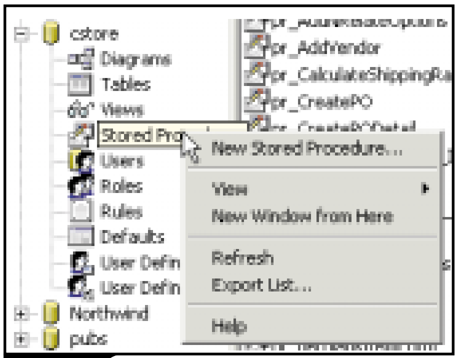
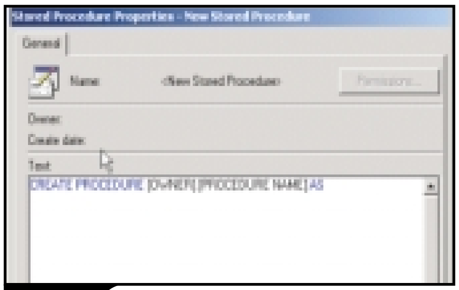


FIGURE 1: Enterprise Manager stored procedure section



```
CREATE PROCEDURE [OWNER].[PROCEDURE NAME] AS
```

FIGURE 2: Stored procedure properties

A stored procedure is a batch of SQL statements optimized on creation in the database. When you send a standard SQL statement to the database using CFQUERY, the database first has to determine how best to execute the statement before it actually runs it. The result? Longer processing time. The more complex the statement, the longer it will take. By creating a stored procedure, the code is optimized already.

Why Use Stored Procedures?

The first reason, faster processing time, has already been addressed. The second reason is that when stored procedures are used properly, they result in less network chatter between your Web server and the database (see Listing 1).

Let's assume that the initial query returned 10 rows. That query is one round-trip to the database. The loop results in 10 more round-trips to the database. It's possible to use a SELECT INTO command to do this in one step, but why return the recordset to the Web server at all? Wouldn't it be nice to send the value "Bill" to the database and only get a message back from the database saying the process was complete?

The third reason to use stored procedures is security. In the security settings for SQL Server you can turn off permissions for all commands and allow only stored procedures to access the database. This will keep malicious SQL code submitted in forms and URLs from threatening your database.

Okay, those reasons sound pretty good. So when *wouldn't* you want to use stored procedures?

- ***If you use cached queries***: They'll execute faster than a stored procedure because the result set is already in the Web server's memory.
- ***If you plan on porting your application to another brand of database***: Stored procedures don't transfer and will have to be rewritten.
- ***If you have a low tolerance for debugging:*** The debugging information that stored procedures return is cryptic and usually too vague to be of immediate assistance. (I'll address how to reduce debugging headaches in Part 2 of this article.)

Now that the basics are out of the way, let's look at the syntax for a stored procedure. (Yes, SQL Enterprise Manager has a stored procedure wizard, but telling prospective employers that you know how to use Microsoft wizards isn't going to get you the big bucks.)

I recommend installing Enterprise Manager and Query Analyzer from the SQL CDs (they're in the client tools option) so you can follow along. Open up Enterprise Manager and drill down into your database to the stored procedure section, shown in Figure 1. Right-click on the right side of the screen and select "New Stored Procedure...". You'll get the screen shown in Figure 2.

Unless you really want to give personal ownership to database objects (not recommended), replace everything in brackets with the name of the procedure you're creating. Enter your SQL code after the "AS" and click "Okay." If your code is syntactically correct, your procedure will be created. If not, you'll get an error message telling you which line number you need to fix.

The end result should look something like Listing 2.

But you don't want every title, of course, just the ones meeting specific criteria, so you need to create some variables. Variables passed into or out of the procedure need to be placed right before the "AS" portion of the procedure. Local variables always begin with "@" and must have a declared data type. If you have a varchar or char data type, you need to include the length of the field in parentheses. Once your variables are declared, you can use them in the procedure (see Listing 3).

Don't put single quotes around local variables or they'll be interpreted as text instead of as variables.

Accessing the Procedure

You now have a useful procedure. How do you get ColdFusion to use it? ColdFusion includes three tags that provide you with access to the procedures.

- ***<CFSTOREDPROC>*** works like <CFQUERY> and wraps the other two tags.
- ***<CFPROCPARAM>*** is where you provide a list of variables for the procedure to accept and give back.
- ***<CFPROCRESULT>*** is the tag to use if you're expecting a result set to be returned, as shown in Listing 4.

The <CFSTOREDPROC> can also take the attribute Debug="Yes|No". If your stored procedure is returning a result set, setting Debug="Yes" will print out the number of rows returned at the bottom of your page. That's all it does. This occurs even if you have debugging disabled in the CFadmin, so I don't recommend using this option in a production environment.

<CFPROCPARAM> needs a TYPE value of "In," "Out," or "InOut." "In" defines values that you're going to give to the procedure. "Out" defines returned values. "InOut" defines values that you provide and then get back. "InOut" is useful only if you want to provide a value and then get back a changed result.

DBVARNAME is supposed to allow you to feed your values into the stored procedure in any order instead of having to list them in the same order as in the stored procedure. This "feature" doesn't work with MSSQL. Even if you use this attribute, the stored procedure still demands that all your values be passed in from <CFSTOREDPROC> as they are found in the procedure. Even so, it can be helpful to use DBVARNAME so you know which CF variables you want associated with which procedure variables.

The <CFPROCRESULT> tag requires the name that you want associated with the result set and the number of the result set that you want. In our example we had only one result set so "RESULTSET" is going to be 1. If we had run a second select query, we'd need to refer to that result set as 2.

Timesaving Functions

Apart from the variables you can name using the @ symbol, a few functions (previously called *global variables* and designated by @@) are available in stored procedures. The most useful are @@Identity, @@Rowcount, and @@Error.

Using SQL2000, @@Identity was the safest way to get the identity value generated by the last INSERT WEO statement. This function is handy when you need to run a second SQL statement to insert the primary key from the first statement into another table. For example, when you create a customer order you may have two tables: Orders contains the customer information and an order number; OrderDetail contains item details. Use the primary key from the Orders table as the foreign key in the OrderDetail table.

When you don't need a variable to be passed either into or out of a procedure, use the DECLARE statement to create a local variable, as in Listing 5.

Integrating ColdFusion and Flash
COLDFUSION Developer's Journal
Creating a Custom XML Editor in ColdFusion
CFDJ Feature: ColdFusion-Driven Flash
Foundations: Fusedoc with XML Vision
CFDJ Feature: SQL Server 2000 Meets XML
CFs: Probe Your Servers

The @@Identity function will almost always be a safe tool but if you do some fancier processing using triggers, you could end up with the wrong identity. SQL2000 introduced a new function, SCOPE_IDENTITY, that returns the identity for the current scope only. @@Identity returns the last identity created for the current connection, which may contain triggers and nested stored procedures. So if you're using SQL-2000 I recommend replacing SET @NewIdent = @@Identity with SET @NewIdent = SCOPE_IDENTITY.

The second useful function, @@Rowcount, returns the number of rows affected by the last statement. @@Rowcount is extremely volatile, changing after any statement, even those that return zero rows, so you'll want to transfer the value to a local variable immediately to get accurate results (see Listing 6). Notice that I added the parameter OUTPUT to the @RowsAffected variable. This allows me to use the @RowsAffected value in my CF code after calling the procedure. To make use of this variable I need to include a CFPROCPARAM tag in the procedure call with a type set to "Out." I also need to provide a variable name using the VARIABLE parameter (see Listing 7).

@@Error is the function to use if you want to do some error handling within the stored procedure (not a bad idea). @@Error returns the error code of the most recent SQL statement issued. If the error code equals zero, the code executed without errors. If there's an error, you can use IF, ELSE statements within the stored procedure to fix it or you can return the error to CF for debugging purposes (see Listing 8).

• • •

I hope this brief introduction to stored procedures has convinced you that they really can be useful and aren't as difficult to work with as you might imagine. In Part 2 I'll show how to use some of T-SQL's programming constructs such as WHILE loops, IF, ELSE, and string manipulation.

ABOUT THE AUTHOR
Ian Rutherford has been working with CF and SQL for two years and is the designer of CatholicStore.com and Catholicliturgy.com, both built using Fusebox architecture. Ian has been working with SQL for two years and recently started working with stored procedures.

IRUTHERF@CATHOLICSTORE.COM

Listing 1

```
<CFQUERY NAME="gettotal">
  SELECT
    Total,
    LastName
  FROM
    Purchase
  WHERE
    Customer = 'Bill'
</CFQUERY>
<CFLOOP QUERY="total">
  <CFQUERY NAME="enterTotal">
    INSERT INTO
      History
      (total,name)
    VALUES
     (#gettotal.Total#,'#gettotal.LastName#')
  </CFQUERY>
</CFLOOP>
```

Listing 2

```
CREATE PROCEDURE sp_getBooks AS
  SELECT
    Titles
  FROM
    Books
  ORDER BY
    Titles
```

Listing 3

```
CREATE PROCEDURE sp_getBooks
  @Author [varchar](30),
  @Price [money]
AS
  SELECT
    Titles
  FROM
    Books
  WHERE
    Author = @Author
  AND
    Price < @Price
  ORDER BY
    Titles
```

Listing 4

```
<CFSTOREDPROC PROCEDURE="sp_getBooks"
    DataSource="Books">
  <CFPROCPARAM TYPE="In" CFSQLTYPE="varchar"
    DBVARNAME="@Author"
    VALUE="#Attributes.Author#">
  <CFPROCPARAM TYPE="In" CFSQLTYPE="money"
    DBVARNAME="@Price"
    VALUE="#Attributes.Price#">
  <CFPROCRESULT NAME="getTitles" RESULTSET="1">
</CFSTOREDPROC>
```

Listing 5

```
CREATE PROCEDURE sp_createOrder
  @Address [varchar](30),
  @Item [int],
  @Quantity [int]
AS

DECLARE
  @NewIdent [int]

INSERT INTO
  Orders
    (Address)
  VALUES
    (@Address)

SET @NewIdent = @@Identity

INSERT INTO
  OrderDetail
  (Orders_Fk,
  Item,
  Quantity)
VALUES
  (@NewIdent,
  @Item,
  @Quantity)
```

Listing 6

```
CREATE PROCEDURE sp_updateInventory
  @MinInventory  [int],
  @RowsAffected  [int]OUTPUT
AS

UPDATE
  Items
SET
  LowStock = 'Y'
WHERE
  Quantity < @MinInventory

SET @RowsAffected = @@Rowcountt
```

Listing 7

```
<CFSTOREDPROC PROCEDURE="sp_updateInventory"
    DataSource="Books">
  <CFPROCPARAM TYPE="In" CFSQLTYPE="integer"
    DBVARNAME="@MinInventory"
    VALUE="#Attributes.Min#">
  <CFPROCPARAM TYPE="Out" CFSQLTYPE="integer"
    DBVARNAME="@RowsAffected"
    VARIABLE="Variables.LowInvCount">
</CFSTOREDPROC>
```

Listing 8

```
CREATE PROCEDURE sp_updatePrice
  @NewPrice [money],
  @Item [int],
  @Error [int]OUTPUT
AS

UPDATE
  Items
SET
  Price = @NewPrice
WHERE
  ItemNumber = @Item

IF @@Error <> 0
  BEGIN
    SET @Error = @@Error
    (error handling code)
  END
ELSE
  BEGIN
    SET @Error = 0
  END
```

CODE
LISTING

The code listing for
this article is also located at
www.sys-con.com/coldfusion/sourcec.cfm

HOSTMYSITE.COM
www.hostmysite.com

Upgrading a ColdFusion Basic

The Macromedia ColdFusion 5 Web Application Construction Kit

By Ben Forta, et al.
Que/Macromedia Press
1,512 pages

REVIEWED BY
ANNETTE KUNOVIC

As a collector of all the Web application construction kit books (lovingly known as CFWACK), I figured *The Macromedia ColdFusion 5 Web Application Construction Kit* would just be a recycled version of previous books – with a dash of new features thrown in.

I was wrong. Yes, there's some of the same material (still good information), but much better written, with all new examples. And it's bigger than the previous books – with 39 chapters plus appendixes.

For those of you who have *ColdFusion Web Application Construction Kit 4.0*, it's time to retire it and get *ColdFusion 5 Web Application Construction Kit* instead. ColdFusion Application Server 5 does so much more and *CF5WACK* does an excellent tutorial job. It answers many questions you'll have while developing. If it doesn't, an advanced *CFWACK* is available.

If you're new to ColdFusion development, this is a great first book to have in your library. Overall, *ColdFusion 5 Web Application Construction Kit* flows better than the previous books, especially in its logical placement of chapters (although Chapter 3, "Building Databases," should be nearer to Chapter 6, "Introduction to SQL").

If you're a new developer, I suggest you read Chapters 1 through 15 in order. These chapters cover coding and how to get your templates working. Here's what you'll learn:

- ***Installation of ColdFusion Studio and ColdFusion Application Server:*** Only a few ColdFusion features are explained to get you started. The rest are saved for later chapters.
- ***Databases, ODBC, SQL:*** This is fundamental if you plan to have a dynamic Web site.
- ***ColdFusion basics:*** Variables, arrays and structures, using functions, conditional statements (CFIF, CFSWITCH), looping, reusing code, when to use pound signs, forms, etc.
- ***Debugging:*** Chapter 15 is a must read for every developer.

As an aside, look for an executable on the CD called *source.exe* that installs the example applications on your computer. The "ows" directory doesn't exist.

Plenty of notes and tips explain the how and why beginning developers run into. I'm glad to see a lot of function use in the example code. Sometimes developers reinvent the wheel because they forget to check the ColdFusion function list.

Beginners can now skip around the chapters after Chapter 15 to get information on specialized features and development needs. Hopefully, you won't need to read all the chapters right away to finish a project. There are chapters on UltraDev and integrating Flash with ColdFusion, so you can get a glimpse of other Macromedia products.

The book's midsection focuses on creating applications. This starts with planning. Chapter 17 does a thorough job of exploring the planning and design phase. Other chapters:

- ***Application framework and session management:*** Locking is explained in detail, making this chapter a must-read.
- ***Security:*** Create levels of granularity for your application.
- ***Custom tags:*** Create CFML custom tags. Previously, this topic appeared only in the advanced CFWACK.
- ***Usability and performance:*** Take advantage of the new CFFLUSH and CFSILENT tags, revamped graphing, and other Java-based tools.
- ***E-mail:*** Sending and receiving e-mail applications is covered.
- ***E-commerce:*** Create a shopping cart and checkout procedure. You might want to refer back to the previous book if you have it. It had more payment-processing examples. The experienced developer will probably be more interested in the advanced chapters.
- ***ColdFusion application server:*** Chapter 30 details all the features. This includes advanced security, sandboxes, and server management tools.
- ***Advanced SQL and stored procedures:*** This is useful for performance if your database supports it.
- ***Error handling:*** Catch errors and throw custom errors.
- ***NonHTML content:*** Create a Wireless Application Protocol (WAP) application.
- ***Operating system interaction and event scheduling:*** CFFILE, CFDIRECTORY, CFSCHEDULE, and so on.
- ***Verity:*** Create full-text search applications.
- ***Management and development methodologies:*** Implementing version control, coding using Fusebox and cfObjects are discussed.

The Macromedia ColdFusion 5 Web Application Construction Kit is just plain *more:* more information, more examples, more tips. Whether you're a beginner or advanced, a programmer or a designer, you'll find this book valuable.

ABOUT THE AUTHOR
Annette Kunovic is cofounder of Trilemetry, Inc. (www.trilemetry.com), a Colorado-based Web applications development company. She is a certified Allaire instructor and was a technical editor on the book, Mastering ColdFusion 4, from Sybex.

ANNETTE@TRILEMETRY.COM

Managing Application Development with **Macromedia Sitespring**

Streamline Web site design and development with new tool

REVIEWED BY
NEIL ROSS

Keeping a development project organized and on point is a common issue that Web designers and application developers alike often struggle with.

Managing your development team, their tasks, and the multiple versions of files they're working with are problems that many of us find ourselves scratching our heads over.

With the introduction of Sitespring, Macromedia joins the crusade to bring a little bit of sanity to our day and to help streamline the Web site design and application development processes.

Core Sitespring Management Concepts

Sitespring is an application designed to help manage Web development projects. It does this, in part, by fostering communication about that project within the development team and with the client. We all know that communication within the development team is vital to project success. For this reason Sitespring creates a personal home page for each team member that consists of a My Projects module, a My Tasks module, a My Discussions module, and a My Reports module.

Sitespring allows administrators, project managers, and Web developers to create projects, assign and organize tasks and team members, and participate in project-oriented discussion threads. Each project has its own Project Homepage within Sitespring, and any team member assigned to that project has access to vital project information. Each project consists of project details regarding client information as well as a list of tasks, discussions, team members, and project folders.

Sitespring has a browser-based user interface so it can be accessed from anywhere on your network or from the Web by any registered Sitespring user. Users can log in to their Sitespring home page using their Sitespring username and password.

Sitespring Manages Projects

A Web shop often has several projects going simultaneously. In the past there's always been a question about which project management tool is best suited to help make sense of all of the things going on in these types of projects. Sitespring, which fits into this niche, can make an immediate difference.

With so many development projects running at once, chances are that each team member is working on more than one. Some developers experience problems when they try to shift their focus from one project to another or try to keep track of task and project responsibilities. Commenting within your templates or Web pages helps to alleviate some of this "downtime," but that's another subject altogether.

The My Projects module is used to identify which projects each team member is assigned to. It allows you to see not only which projects you're assigned to, but which have the highest priority along with their current status.

If you click on one of your projects, you'll be able to see a description of the project and project details including all of the team members assigned to that project, project tasks and task status, all project discussions, and more. Each of the project tasks listed shows its own priority status, the name of the individual it's assigned to, and the due dates.

Sitespring also allows for the import and export of project information. A Microsoft Project project plan can be imported into Sitespring. You can also export project or task information in MPX format for Microsoft Project or in comma-separated values (CSV) format for more in-depth tracking and project reporting.

Sitespring Manages Tasks

The ability to track tasks across the full life of a project would be a helpful thing for project managers. So, to ensure that team members know exactly what they're doing and that project managers can stay current with project progress, Sitespring allows for the creation and assignment of tasks. Each project may consist of a number of tasks, each assigned to one or more team members.

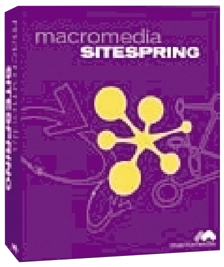
Any member of the team is allowed to create and assign tasks. Each task is assigned to a specific team member so that accountability is established. If more than one team member needs to work on a task, that task can be cloned and the second instance assigned to the other team member. Each task is added to the project task list and to the My Tasks module on the assigned team member's home page.

One of the deliverables of most projects is an estimation of time. An estimated time to completion can be assigned to each task. When the task is completed, the actual time should be entered to help project managers determine the efficiency of the project team. Since each task

is listed on the project page, its progress and status can be monitored by the team, providing incentive for team members.

Because all tasks related to a project are available to every project team member, the entire team can monitor task and project progress. This feature also allows them to stay informed regarding what will next appear on their plate.

Sitespring easily integrates its task management features with extensions for Dreamweaver and Dreamweaver UltraDev. The Sitespring Task Panel allows you to log in to Sitespring and to view all projects and tasks that you have open. You can access all of your files, but you can also edit your tasks within the Sitespring Task Panel and refresh for those edits to take effect.



Sitespring Manages Communication

Communication is the key to making sure that all members of the team know where they're supposed to be and when, what task is pushing its deadline, or even that they need to call a client contact. Sitespring works to ensure that the lines of communication are always wide open.

The My Tasks module is one of the ways that Sitespring enhances communication. But Sitespring also keeps you up to date on the ongoing discussion issues by presenting project-centered discussions on your project page. Built just like any other threaded discussion application, Sitespring lets you know what topics are being discussed, who started the thread, and how active it is.

Sitespring also ensures, via e-mail, that you're informed about your assignments. On the setup side you have to tell Sitespring what SMTP server to use, the SMTP port, and so on. The e-mail notification features, task assignment, and project discussions make the project team members more accountable for meeting deadlines and encourage them to keep in touch with each other.

Sitespring also provides for client communication through a project site that allows them to contribute files containing source material to the project. Clients can also view drafts of the work being performed, which keeps them up to date on their project's status.

Because Sitespring 1.0 lacks a project calendar, the Discussions section of the Projects page becomes a great place to post meeting notes, reminders, and directions to after-hours team-building activities.

Sitespring Manages Files

One very cool feature of Sitespring is its transparent versioning. Each project is set up in a shared project folder that Sitespring monitors. Any file that's modified within that shared folder will be versioned. The best thing about this feature is that you as the developer don't have to change your routine. You don't have to open a file to edit through the Sitespring explorer, but can use your same old methods of opening with ColdFusion Studio, Dreamweaver, or Ultradev.

It's as simple as creating a share for access to those folders and enabling File Versioning within Sitespring's Shared Folder Management. Any ongoing changes or updates to files within that share will be tracked by Sitespring's version management. What happens is this: when you enable file versioning for a share, Sitespring automatically creates a "_revisions" folder in the root directory of that share. Then, as you modify and save a file, Sitespring will save the old version of that file to the _revisions folder. A sequential number will be appended to each version of the file so that you can easily step back to the version that you want to restore. If the latest version of a page or application template isn't the greatest, this feature makes it easy to revert to the preferred version.

You can also take a look at the revision history of a file by accessing it through Sitespring's File Explorer. Within the Explorer you can clean up the revision history and clear out any obsolete revision files.

Another helpful feature for developers is the ability to link specific files to tasks. This assures that the file that's updated is the file that was intended.

Remote Collaboration

I recently read Hal Helms's ***CFDJ*** article "A Script for Teamwork" (Volume 3, issue 7). Hal talked about his experiences on a "distributed team." Sitespring could have alleviated some of the issues Hal recounted by fostering communication between team members and project managers and allowing client interaction. That interaction and communication helps keep everyone on task and on track.

Though Sitespring is neither a framework nor a development methodology, it's definitely a tool that fits into the concept of development via methodology, task assignment, and version management. In the same issue of ***CFDJ***, Sarge, an old buddy from my Allaire consulting days, discussed the benefits of frameworks and methodologies, and of team specialists, in "Optimal Development Environment: Working Together in CF." Sarge's assertion that "there's no substitute…" for specialists within your development team is supported by Sitespring's task-oriented approach to dividing up project responsibility and accountability.

Because Sitespring is a Web-based application and runs on your development environment, the project development files are accessible through your Web browser wherever you are. There's no check-in/checkout feature, but the task assignment and transparent versioning are great for remote developers. Remember, you can also access your project files by setting up a ColdFusion RDS mapping.

Leveraging ColdFusion

Okay, so where does ColdFusion fit into all of this? The best answer: anywhere. Sitespring and its "Project Sites" are run on Macromedia's JRun server. Thus, when a client interacts with the Project Site or when a development team member logs on to the Sitespring desktop, he or she is using a JRun application that's delivered in JSP pages.

Your Sitespring project folders can be made up of any number of file types. Create a folder in Windows Explorer or by opening up that project's directory. Your site templates are created within Sitespring's Projects folder rather than under the normal Web root (usually C:\InetPub\WWW Root\projectname).

ColdFusion doesn't interact with the Sitespring product, but as an application developer you'll appreciate the ability to assign and track development tasks and to interact with team members and clients. Many of ColdFusion's most popular development methodologies take a modular approach that encourages developers to work on independent blocks of code that can then be plugged in to the rest of the application.

The Future of Sitespring

The future of Macromedia seems bright, even in such an uncertain market. Many of us were really excited when we heard about the merger of Allaire and Macromedia simply because the upside of the mixture of the front-end and back-end technologies would lead to a more integrated design and development process. With the addition of Sitespring, we have a way to track and manage the activities that are involved in our projects on both sides of the process.

When I asked how Sitespring works into Macromedia's long-term strategy, Macromedia officials said, "We see Sitespring as a product that can help bring together the diverse workflow of all of the team members involved in delivering incredible Web applications. As such, you can expect us to improve the workflow between the tools themselves (e.g., how Flash and CF and Java and graphics folks tools interact) as well as with Sitespring as a common team production management platform."

That said, I think we can look for improvements to some of the…how did I put it?…"clunky" features of Sitespring, and the addition of some features that didn't make it into version 1.0. Like the much-needed project calendar.

Conclusion

I went into the writing of this article not knowing much about Macromedia Sitespring. I'd heard a bit of the hype, but had been too busy with a development project to stop and look at the product. After a period of some weeks I've become a believer. Yeah, it's only a version 1.0 product, but the potential is definitely there. I realize how useful a product like Sitespring could have been on several past projects, and how much time I could have saved through its version control and discussion features.

I've heard people in my office who've tried Sitespring say things like, "I can see where it might be good for a large development project, but it's not a good solution for the type of work we do around here." Au contraire, mon ami! Whether you're developing a brochure-ware type of Web site or a full-blown enterprise application or doing ad hoc updates to various files on an existing Web site, Sitespring can help.

To Macromedia's credit, Sitespring is built so you don't have to be a technical genius to figure it out. It takes advantage of an easy-to-use browser interface for almost all its interaction with project files, team members, and clients. Sitespring's inline help and Flash tutorials are a great place to start learning more about this product and how you can use it to your advantage.

ABOUT THE AUTHOR

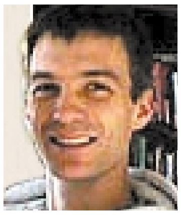
Neil Ross, an iConsultant with Ciber, Inc., is a partner in Codesweeper Consulting (www.codesweeper.com) and actively supports the ColdFusion developer community as a founding member of the Central Pennsylvania ColdFusion User Group.

NEIL.ROSS@CODESWEEPER.COM

REGISTER NOW. CONFERENCE FILLING FAST
SAVE $100
XML JOURNAL
SECURING AND AUTHENTICATING SOAP-BASED WEB SERVICES
XML Feature: The Brave New World of Web Services: SOAP, WSDL, and UDDI
XML.ORG: XML.org Expands as Online Resource for XML in Industry
Web Services: A Platform for XML Web Services
Q&A: Shedding a Little Light on XML

The **Logical Stack Trace**

Follow the rules of logic for faster code processing



BY
EBEN HEWITT

"Contrariwise," continued Tweedledee, " if it was so, it might be; and if it were so, it would be; but as it isn't, it ain't. That's logic."

— Lewis Carroll

To elucidate a manner in which ColdFusion programmers can write better performing, cleaner, and more sensible applications, this article will illuminate the noncoincidental relationship between the discipline of logic in philosophy and the writing of ColdFusion applications. Understanding this relationship will help you approach application flow management in a lean and efficient manner.

I've purposefully kept this article abstract, since it works at the conceptual level. The ultimate point is to apply the principles set down.

This is not an apology for a new framework; it's not my intention to propose competition for cfobjects or Fusebox. I use the term *objects* as a term of art, that is, I don't refer to a *COM object* or a *Java object*. I'm proposing a logical way to think about coding and to optimize your code in whatever given framework you currently enjoy.

I start with three major assertions:

1. A programmer's understanding of logical constructs and equations is the chief means of writing code that's consistent, easily understood, and quickly processed by ColdFusion.
2. Writing ColdFusion is a necessary generation of sets.
3. A ColdFusion application can be written and understood as a form of logical proof.

Each of the three assertions is inextricably related in this context; I'll therefore refer to them in various ways. Let me note first that I don't refer to *logical proof* as merely an instance of syllogism (which takes the form of premise, secondary premise, conclusion); I mean to invoke the sense that a logical proof requires something to be proved, though the auxiliary point isn't lost.

ColdFusion is a language and an application server conceived by a gentleman with a degree in philosophy. Just as it's no coincidence that ColdFusion contains commands shared with other programming languages (switch/case, try/catch), ColdFusion also shares much with philosophy – specifically set theory and logic. By extension, a ColdFusion application conceived as a mathematical or logical proof will follow the rules of those disciplines.

ColdFusion applications often operate within a series of if/else constructs. In general, ColdFusion programmers understand by conditional logic that it's what happens when they need to represent the following circumstance:

If A is true, execute B code; in all other cases, execute C code.

The tools available to construct conditional logic clauses within CFML are:

1. ***Variables:*** Test for existence or truth-value).
2. ***Decision functions:*** Route the actions.
3. ***Operators:*** Define possible states.

At the design stage the usefulness of conditional logic as the supposition of possible user interaction can't be stressed enough. You can create a structure known as a *use-case scenario*, in which all possible user interactions with the program, and all programmatic responses, are mapped. As the "world" of the program is circumscribed, by itself, this is in fact a finite construct.

To instantiate the above construct (3), we might write the following:

```
<cfif Request.Ses.LoggedIn is 1>
 <cfinclude template="welcome
.cfm">
<cfelse>
 <cflocation url="login.cfm">
</cfif>
```

It's a simple case in which the second event (the execution of either *B* or *C* code) is predicated on the truth-value of *A* within the given context. You'd quickly conclude that "within a given context" represents a set; that is, the set in which Request.Ses.LoggedIn is capable of truth-value. In ColdFusion you generate such a set by declaring a variable that this code (*A*) can test against. ColdFusion programmers often seem to leave things at that and move on.

If we allow for a more complete version of the meaning of the "given context," we can imagine an entire application. Think of the stack trace you use in debugging your applications. A stack in programming is a data structure in which items are removed in reverse of the order in which they were added. The trace is a genealogy of instances.

There's a logical stack trace for any given point in your application. A logical stack trace is a snapshot of the current structure, the complete definition of the logical structure contextualizing any given line of code. What's precisely locatable in this logical stack trace is the entire set of conditions that are true, false, initialized, or not yet initialized.

There is great value in knowing, even before you write, the logical stack trace of your application. Knowing it will help you plan your applications more quickly.

Perhaps the most important point to stress is that it'll prevent your writing unnecessary constructs and setting variables whose existence or value is redundant to test for. Build your application in the manner of a logical proof and you'll reduce writing time and write optimized code.

How then do you write an application in this manner?

To help us, let's refer to George Boole, the self-taught mathematician on whose work much current programming is based, and to the set theorist Bertrand Russell.

Paraphrasing Boole, we'll employ the symbol 1 to refer to everything within our ColdFusion application – every conceivable class of objects that may or may not exist. It is premised that the same individual object may be found in more than one class, as it may possess more than one quality in common with other objects. The usefulness of this is that it allows us to express our application in mathematical – and hence logical – terms. (We'll explore this later.)

However, we can see loopholes within Boole's conceptualizing. Bertrand Russell has a famous story that illustrates the problem; it's a story about a barber:

There was once a barber.

Some say he lived in Seville. Wherever he lived, all of the men in this town either shaved themselves or were shaved by the barber. And the barber only shaved the men who did not shave themselves.

You can immediately see the paradox when you attempt to answer the question: Did the barber shave himself? When we conceive of Boole's "1," or the set that contains all sets, we must conceive of a set of sets, that is, the set that contains all other sets, which must contain itself, ad infinitum. We must also conceive of {}, or the empty set, such as "The set of all living humans who do not have brains"– there are no members of this class. By extension we get the "set of all sets that do not contain themselves," which is an inconsistency.

I belabor the inconsistency within set theory (and consequently mathematics) to dispel irrelevant objections, and to prevent certain frustrations, and because it's an important conversation of the last hundred years. We're free to conceive of sets within sets within the set of our application. We select objects, such as Request.Ses.-LoggedIn, as the nodes of our structure, and then demark the structure as (1) the set over which this object has purview, and (2) the else-set, that is, all moments of the application wherein the object is not defined or not defined in equivalency.

Applications as Logical Proofs

An act of selection of an object from a class of objects is independent of the order in which successive objects are selected. I may select cats, and from cats I may select only gray cats. This is equivalent to my selection of all things that are gray and a subsequent selection of cats from within that class. Boole would write:

```
xy = yx
```

which we'd translate as "declare a class, every *y* of *x*. It is equivalent to the class of every *x* of *y*."

Perhaps you object, saying that while this is true for logic, where the matter is concluding a law of thought, it is not necessarily true within an application. So, we ask, What is the time continuum of a ColdFusion application? It takes place within a user-defined circumstance, which is in turn within a programmer-defined circumstance within a language-defined circumstance. Debugging output allows us the following conclusions:

- ColdFusion is aware of precisely its *entire* state at each moment, for every moment.
- ColdFusion outputs all queries run, all form variables passed.
- ColdFusion will accept or reject local and other variables based on availability.

It follows that an application that can be understood and written in this manner will optimize flow control and reduce redundancy. How are we to understand an application in this manner? By writing it as a logical proof.

To do this, we must consider the variables and expressions as contained within the structure of the logical stack trace. ColdFusion certainly does this. A number of corollaries immediately follow, and we can use these logical propositions to our advantage when designing flow control for our applications. We can therefore conclude the conversion of propositions:

1. Propositions are *convertible*. A proposition is said to be converted once its terms are transposed. Example: "No programmer is a tyrant" converts to "No tyrant is a programmer."

This is worth noting, given ColdFusion's "short circuit" evaluation method: when ColdFusion tests a concatenated statement such as:

```
<cfif Request.Ses.LoggedIn IS
Yes AND UserLevel is 12>
```

if Request.Ses.LoggedIn turns out to be false. ColdFusion never evaluates the second expression; if one expression is false, then the entire statement is necessarily false. You can use this to your advantage in writing: order concatenated expressions according to the likelihood of their truth or falsity within the given moment.

2. A conversion *per accidens* is a conversion by limitation, and this is allowable. Example: "All programmers are people" converts by limitation to "Some people are programmers."

You can take advantage of this in your applications. Locate the (necessarily existing) corresponding set to any given object and use it to optimize your code. For example, in a possible world where User-Level may exist, it may have a value of 12 or another number. You could test for existence and value in the usual manner and invoke such construction repeatedly as needed. However, the foregoing allows us to abstract away from variables at this level.

3. Negative conversions (or conversions by *contraposition*) accompany positive terms. Example: "Every programmer is a genius" converts by negation into "that which is not a genius is not a programmer."

Each object within a set can be conceived not only as itself, but it necessarily, concomitantly, defines a corresponding set of not-X. This is stated as a corollary of statement 2 above.

The class X	*x*	
The class not-X	1-*x* (where 1 is our application, or the entirety)	
All Xs are Ys All Ys are Xs	*x* = *y*	
No Xs are Ys	*xy* = *0*	
All Ys are Xs Some Xs are Ys	*y* = *vx* (*vx* = some *Xs*) *v* (1-*x*) = *0*	
Some Xs are Ys	*y* = *vx* (where vx = some x's; *v(1-x)* = *0*)	
No Ys are Xs Some not-Xs are Ys	*y* = *v* (1-*x*)	
Some Xs are Ys	*v* = *xy* OR *vx* = *vy* OR *vx(1-y)* = *0*	*v* = some *Xs or* some *Ys* *vx* = some *Xs, vy* = some *Ys* *v(1-x)* = *0, v(1-y)* = *0*
Some Xs are not Ys	*v = x(1-y)* OR *vx = v(1-y)* OR *vxy = 0*	*v = some Xs, or some not-Ys* *vx = some Xs, v(1-y) = some not-Ys* *v(1-x) = 0, vy = 0*

TABLE 1 A simple series of implied logical equations

There are three conclusions:

1. We can define within application.cfm an abstract set of sets, employable globally, and test against existence or value for nested set nodes.
2. By using the laws of logic, we can reduce the total number of variables required to manage application flow.
3. It follows that reduced variable creation will result in reduced expression testing, whether it's testing for truth-value, existence, quantitative value, or qualitative value.

Table 1 displays a series of implied equations you can use to express your logical stack trace. To bring this to life you must:

- *Consider* all of the logical conclusions in your applications once you begin to incorporate flow control, and consider variables in the design phase.
- *Choose* a scope to pass them through, based on availability.
- *Discover* the limit of use-case scenarios so that "1" represents something meaningful inasmuch as there are no user interactions that go unaccounted for.
- *Abstract* your variable and expression testing within a set of sets required. (After you implement the practices above, this should be apparent to you without your having to do much extra work.)

In this way we can conceivably write an abstract of our entire application – including all possible use-case scenarios – as a mathematical expression. If you follow the rules of logical thought, your code will process faster. It will be easier to write and quicker to debug. Your formalism will allow you to reuse code from application to application more easily and to write code to meet new challenges from within a sensible system.

Acknowledgment

I'm indebted to Alison Leigh Brown for her assistance in this article. For further reference please refer to *The Mathematical Analysis of Logic*, and *Investigation of the Laws of Thought*, both by George Boole.

ABOUT THE AUTHOR
Eben Hewitt is director of production at Cybertrails. He is a Macromedia-certified CF developer and a certified SQL Server 7 administrator. Eben is on the faculty of ColdFusion Edge and is the author of Core ColdFusion 5.

EBEN@CYBERTRAILS.COM

SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 **Fax:** 201 782-9600

president and ceo
Fuat Kircaali fuat@sys-con.com

advertising
senior vp, sales & marketing
Carmen Gonzalez carmen@sys-con.com
vp, sales & marketing
Miles Silverman miles@sys-con.com
advertising director
Robyn Forma robyn@sys-con.com
advertising account manager
Megan Ring megan@sys-con.com
associate sales manager
Carrie Gebert carrie@sys-con.com
associate sales manager
Alisa Catalano alisa@sys-con.com
associate sales manager
Kristen Kuhnle kristin@sys-con.com

editorial
executive editor
M'lou Pinkham mpinkham@sys-con.com
editor
Nancy Valentine nancy@sys-con.com
managing editor
Cheryl Van Sise cheryl@sys-con.com
associate editor
Jamie Matusow jamie@sys-con.com
associate editor
Gail Schultz gail@sys-con.com
associate editor
Brenda Greene brenda@sys-con.com
assistant editor
Lin Goetz lin@sys-con.com

production
vice president, production
Jim Morgan jim@sys-con.com
art director
Alex Botero alex@sys-con.com
assistant art director
Cathryn Burak cathyb@sys-con.com
assistant art director
Louis F. Cuffari louis@sys-con.com
assistant art director
Richard Silverberg richards@sys-con.com
graphic designer
Abraham Addo abraham@sys-con.com
graphic designer
Aarathi Venkataraman aarathi@sys-con.com

sys-con events
vice president, events
Cathy Walters cathyw@sys-con.com
sales executive, exhibits
Richard Anderson richard@sys-con.com
sales executive, exhibits
Michael Pesnick michael@sys-con.com
conference manager
Michael Lynch mike@sys-con.com
show assistant
Niki Panagopoulos niki@sys-con.com
JDJ store manager
Anthony D. Spitzer tony@sys-con.com

web services
web designer
Stephen Kilmurray stephen@sys-con.com
web designer
Purva Dave purva@sys-con.com
web design intern
Carol Auslander carol@sys-con.com

accounting
chief financial officer
Bruce Kanner bruce@sys-con.com
assistant controller
Judith Calnan judith@sys-con.com
accounts payable
Joan LaRose joan@sys-con.com
accounts receivable
Jan Braidech jan@sys-con.com

Debugging Tips and Tricks for Detecting Errors

But the best line of defense is preventitive

BY **KEVIN SCHMIDT**

Imagine, for a moment, a world in which we all wrote perfect code – a world in which we'd never have to spend precious time debugging our applications.

Okay, your moment's up. The world is, of course, far from perfect, and the reality is that we spend a fair amount of our time trying to chase down ColdFusion errors. After several hours of staring at the same lines of code, it's easy to become frustrated – especially if there's a deadline looming, or worse, a boss peering over your shoulder asking, "Is it ready yet? Can I demo it?"

My goal, then, is to keep you from throwing your monitor – or your boss – out the nearest window when you can't locate the problem. In this article we'll explore some of the tips, tricks, and general concepts for ferreting out those agonizing errors.

Relaaaaaax

The first thing I do if I'm having problems tracking down an error is to take a break. Frankly, it doesn't matter what you do: get a drink, smoke a cigarette, or just walk around outside. Sometimes a break is all you need. And when you come back, you'll often find the solution staring you in the face, causing you to wonder how you could've missed something so simple. So the next time you start to see red, take a few deep breaths and take some time away from code.

Enlist Help

Two heads – or three or four – are always better than one. If you're part of a team of developers and you've found yourself stuck, ask one of your co-workers to take a look; a fresh set of eyes always seems to help. Once again, when your colleague points out the problem, you'll probably find yourself wondering, "How did I miss that?" If you don't have anyone around to help, don't fret; there are still plenty of other avenues to explore.

In the age of e-mail and the Web, help is usually only a few clicks away. Think about it: you have numerous Web sites and mailing lists at your disposal, each of them with ColdFusion developers at the ready.

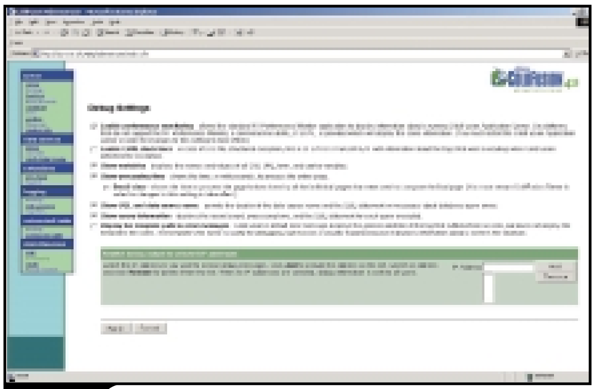


FIGURE 1: Debug settings

The forums in the developer section of Allaire's Web site (www.allaire.com) are an excellent resource as is the CF-talk mailing list available at the House of Fusion (www.houseoffusion.com). Both options usually lead to a flood of answers to your problem. The point is, whether you seek a co-worker's assistance or rely on a Web-based method, you're likely to find the solution.

Coding Tricks

Up until now, the tips we've explored have involved asking for help after the fact. But what about some preventive medicine? What about some pointers for when we're actually writing the code?

The first (and most fundamental) thing we can do is to use the debugging option on the ColdFusion server. This can be accessed by logging into the administrator and clicking on debugging (see Figure 1). We can select from a range of options, including Show Variables, Show Processing Time, Show SQL, and data source name. Each option gives us different information. The ability to see what form fields were passed in and what their values were can be of tremendous help. In addition, you can view what URL variables were passed in as well as their values. If you're having problems with queries, Show Query Information displays the record count, processing time, and the SQL statement for each query executed.

Use <cfabort>

I prefer to use <cfabort> if I'm having problems getting values into a database for one reason or another. And <cfabort> stops processing the page at the tag's location. When the tag is reached, ColdFusion outputs everything that was processed up until that time. When we want to check the values of variables right before they go into the database, we use a <cfoutput> block to output all their values, then use <cfabort> so the database query is never executed.

From here, we can examine what might be going wrong with our variables. The code in Listing 1 illustrates how this might work.

Now we can view our variables and make sure all their information is correct. If not, we can readily see what's going on and correct those variables before they're inserted into the database. Using <cfabort> works anytime we need to stop processing – before we enter a loop or while we're looping. No matter what, <cfabort> will stop the server from processing a page.

Using a Marker

Most of the time, the error page generated by ColdFusion gives us some excellent insight into where the error is and what's causing it, but this isn't always the case. I find that if I'm working on a page and I want to know where the processing is going before it fails, I use a marker.

A marker is nothing more than a <cfoutput> tag with a word or a letter to be output. When I view my page, I look for the marker; if I see it, I know that the processing made it at least to that point before it failed. By moving this marker around, it's easy to determine where our code is failing. The following simple example illustrates how this works.

```
<!---  CF code above --->

<!--- Output the marker --->
<cfoutput>
We made it to here
</cfoutput>

<!--- CF code below --->
```

By using the marker, we can determine what line of code was the last to process properly before the error occurred. Another variation of this method is to use two markers and position them around blocks of code. This way we'll know if a particular block of code executed successfully. See Listing 2 for an example of this.

Now we'll know whether that block of code is the culprit. By moving these markers around, we can discover what code is generating the errors. In short, either of these methods will achieve the same goal: finding the code that's causing the errors.

Debugging Dilemma

We've covered just a sampling of the ways in which we can address the debugging dilemma. Numerous other techniques and third-party software tools are at our disposal, all of them designed to do one thing – find our errors and help us fix them. Perhaps a co-worker holds the key, or perhaps you'll find your answer on the Web or by using your own techniques while writing the code.

In any case, you'll soon find yourself in a better place, with a better application. Thus, while we can all continue to pursue that mythical white whale we call perfect code, we're probably a lot better off simply honing our debugging skills.

ABOUT THE AUTHOR

Kevin Schmidt is a senior Web developer for GeoLearning, Inc., a developer of Online Learning Management Systems, in West Des Moines, Iowa. He also runs the Des Moines ColdFusion User Group.

KEVINS@GEOLEARNING.COM

Listing 1

```
 <!--- Output variables to be inserted into database --
->
 <cfoutput>
 #b_date#
 #b_name#
 #b_orderid#
 #b_items#
 </cfoutput>

 <!--- use <cfabort> to stop the processing --->
 <cfabort>


 <!--- When you use <cfabort> the sql query does not
get executed --->

<!--- sql query --->
 <cfquery datasource="orders" name="neworders">
 ……
 </cfquery>
```

Listing 2

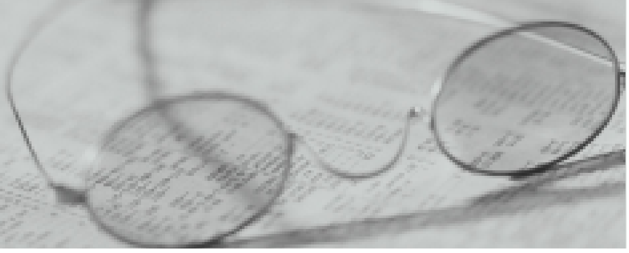
```
<!--- First Marker --->
 <cfoutput>
 Marker 1
 </cfoutput>

 <!--- block of CF Code --->
 …
 <!--- end block --->

 <!--- Second Marker --->
 <cfoutput>
 Marker 2
 </cfoutput>
```

CODE LISTING

The code listing for this article is also located at www.sys-con.com/coldfusion/sourcec.cfm

RED HORIZON

www.sourceaction.com



Letters to the Editor. . .

Locking Request Variables

Charles Arehart responds to a reader's query regarding his article, "Why It's Wrong to Use Application.DSN in Your Templates," [Vol. 3, issue 7]:

A reader asked me by e-mail: "Don't you also need to lock request vars?" The answer is no. They're not shared by any other user, and they're not shared by any other "thread."

The request scope, like the variables scope, is local to the running program and belongs only to that program. It's a single thread of execution, so there's no risk of conflict with other threads, as with the "shared scope" variables, server, application, and session.

Charles Arehart
carehart@systemanage.com

Making .vtm Files for Custom Tags

I just wanted to thank Christian Schneider for writing "VTML by Example", Vol. 3, issue 7. I've wanted to make .vtm files for my custom tags (as well as edit some of the CF tag completion menus), but have not had the time to figure out how. After only five minutes I have almost everything done. I never thought the tools I needed were already in Studio, and that it would be this easy!

Great article!

Dave West
dave@davewest.com

Modeling Correct Use of <cflock>

In the July issue of ***ColdFusion Developer's Journal*** [Vol. 3, issue 7], I was glad to see the article "Why It's Wrong to Use Application.DSN in Your Templates" by Charles Arehart. As a senior product support engineer for Macromedia, I find that improper use of shared-scope variables and <cflock> is the most common cause of server instability that we see in ColdFusion support. This article sets the record straight in advising against a common mistake made by many ColdFusion developers.

However, I was dismayed to see that in the same issue there's an article, "CFContent with Images" by Eron Cohen and Michael Smith, that ignores this advice and uses datasource=#application.dsn# in its code example on Page 43.

Unfortunately, this is not the first time I've seen code examples in ***CFDJ*** that use unlocked shared-scope variables.

I'm sure everyone in charge of a ColdFusion server, as well as everyone in ColdFusion tech support, would appreciate it if future articles were edited with an eye toward modeling the correct use of <cflock>.

David Stanten
dstanten@macromedia.com

Missed an issue?

We've got 'em all for you on CD!

JAVA DEVELOPER'S JOURNAL

The most complete library of exclusive *JDJ* articles on one CD!

Check out over 500 articles covering topics such as...
Java Fundamentals, Advanced Java, Object Orientation, Java Applets, AWT, Swing, Threads, JavaBeans, Java & Databases, Security, Client/Server, Java Servlets, Server Side, Enterprise Java, Java Native Interface, CORBA, Libraries, Embedded Java, XML, Wireless, IDEs, and much more!

***JDJ* The Complete Works**
~~Reg $119.99~~
Buy Online
Only $71.99

XML JOURNAL

The most complete library of exclusive *XML-J* articles on one CD!

Check out over 150 articles covering topics such as...
XML in Transit, XML B2B, Java & XML, The XML Files, XML & WML, Voice XML, SYS-CON Radio, XML & XSLT, XML & XSL, XML & XHTML, 2B or Not 2B, XML Industry Insider, XML Script, <e-BizML>, XML & Business, XML Demystified, XML & E-Commerce, XML Middleware, and much more!

***XML-J* The Complete Works**
~~Reg $59.99~~
Buy Online
Only $53.99

COLDFUSION Developer's Journal

The most complete library of exclusive *CFDJ* articles!

Check out over 250 articles covering topics such as...
Custom Tags, ColdFusion and Java, Finding a Web Host, Conference Reports, Server Stability, Site Performance, SYS-CON Radio, ColdFusion Tips and Techniques, Using XML and XSLT with ColdFusion, Fusebox, Building E-Business Apps, Applicaton Frameworks, Error Handling, and more!

***CFDJ* The Complete Works**
~~Reg $79.99~~
Buy Online
Only $71.99

SPECIAL OFFER:
Buy CFDJ & CFA The Complete Works
For Only $129.99

CF Advisor

The most complete library of exclusive *CFA* articles!

Check out over 200 articles covering topics such as...
E-Commerce, Interviews, Custom Tags, Fusebox, Editorials, Databases, News, CF & Java, CFBasics, Reviews, Scalability, Enterprise CF, CF Applications, CF Tips & Techniques, Programming Techniques, Forms, Object-Oriented CF, WDDX, Upgrading CF, Programming Tips, Wireless, Verity, Source Code, and more!

***CFA* The Complete Works**
~~Reg $79.99~~
Buy Online
Only $71.99

JDJStore.com

Order Online and Save 10% or More!

www.JDJSTORE.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

COLLECT ALL 4
FOR ONLY $229.99
JDJ, XML-J CFDJ & CFA

<the world's leading independent coldfusion magazine>

COLDFUSION Developer's Journal

home advertise subscribe contact SYS-CON MEDIA

<dot.com>
- buyer's guide
- coldfusion forums
- mailing list
- coldfusion jobs
- coldfusion store

<magazine>
- advertise
- authors
- customer service
- editorial board
- subscribe

<content>
- archives
- digital edition
- editorial
- features
- interviews
- product reviews

<conferences>
- coldfusion edge fast track new york, ny sept 23–26

<search cfdj>

Search

<cfdj specials>

JDJStore.com
$175.99

JDJ, XML, CFDJ - ALL THREE!!!-The Complete Works

<bestsellers>

1. Goetz's Net Tools Pro Kit from Lawrence Goetz $19.99
2. JRun Studio 3.0 from Macromedia $449.99
3. Homesite v4.5 Win 32 from Macromedia $89.99
4. ColdFusion 5.0 Server Enterprise $4,095.00
5. ColdFusion 5.0 Server Professional $1,295.00

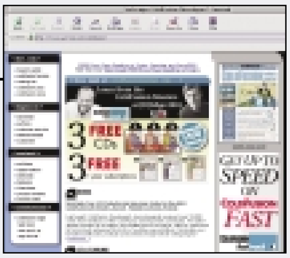
October **2001**

What's Online

www.sys-con.com/coldfusion

CFDJ Online

Check in every day for up-to-the-minute news, events, and developments in the ColdFusion industry. Visit www.sys-con.com/coldfusion and be the first to know what's happening.



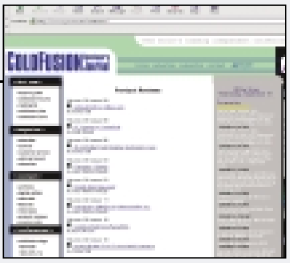
Readers' Choice Awards Vote Now

Make your voice heard...this is your chance to show what you think. Categories for this year's awards include Best Book, Best Consulting Service, Best Custom Tag, Best Database Tool, Best Design Service, Best E-Business Software, Best Education and Training, Best Testing Tool, Best Web Development Tool, Best Web Hosting, Best Web Site, Best Web Application, and Most Innovative CF Application. To vote in this year's awards visit www.sys-con.com/coldfusion/readerschoice2001.

Product Reviews

Considering a product upgrade? Want to know the ins and outs of a new product before you purchase it? Make sure you read our in-depth product reviews.

Our writers get behind the hype and give you the facts. All products are tested by experts and leaders in the information technology industry.



JDJ Store CD Special

The complete library of *CFDJ, JDJ,* and *XML-J* articles are available on CD at a special price for a limited time.

Order now and have more than 1,000 articles on hand for research and review. There are features, how-tos, product reviews, case studies, tips and tricks, interviews, IMHOs, and more!

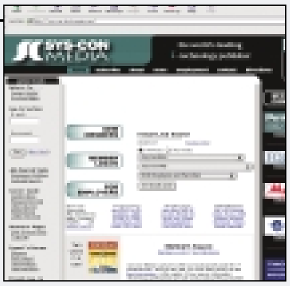
This exclusive package normally sells for $260, but it can now be yours for only $175.99. Order today and save; this is a limited offer!

Search ColdFusion Jobs

ColdFusion Developer's Journal is proud to offer our employment portal for IT professionals. Get direct access to the best companies in the nation. Learn about the "hidden job market" and how you can find it. As an IT professional curious about the market, this is the site you've been looking for.

Simply type in the keyword, job title, and location – and get instant results. You can search by salary, company, or industry.

Need more help? Our experts can assist you with retirement planning, putting together a résumé, immigration issues, and more.



QUICK POLL

What is Your Favorite New Feature of ColdFusion 5.0?
- User Defined Functions
- Query of Queries
- Application Deployment
- Integrated Charting
- Performance Improvements

Vote

Results

Click for a FREE 30-day trial
AltoWeb

click here to DOWNLOAD e-TEST suite
empirix

INTERMEDIA.NET NT web hosting FREE setup code CFWEB

AbleCommerce FREE TRIAL

CFDynamics QUALITY ColdFusion HOSTING

minerva

IS YOUR WEBSITE MISSING IN ACTION

Podium v4 can! Free demo at CFXtras.com

FREE Domain Names FREE ECommerce Software

Cold Fusion Hosting virtualscape

DOWNLOAD COLDFUSION

wireless EDGE conference&expo

Plan to Exhibit

Provide the Resources To Implement Wireless Strategy

The conference will motivate and educate. The expo is where attendees will want to turn ideas into reality. Be ready to offer solutions.

INTERNATIONAL WIRELESS BUSINESS & TECHNOLOGY CONFERENCE & EXPO

CONFERENCE & EXPO

CONFERENCE & EXPO

Shaping Wireless Strategy for the Enterprise

Santa Clara, CA | **May 7-9, 2002**

WirelessEdge will provide the depth and breadth of education and product resources to allow companies to shape and implement their wireless strategy. Developers, i-technology professionals and IT/IS management will eagerly attend.

Plan to Attend the 3-DAY Conference

WHO SHOULD ATTEND

Mobile & Wireless Application Professionals who are driving their enterprise's wireless initiatives:

- Program Developers
- Development Managers
- Project Managers
- Project Leaders
- Network Managers
- Senior IT and Business Executives

Conference Tracks

Track One: Development	Track Two: Connectivity	Track Three: Wireless Apps	Track Four: Hardware	Track Five: Business Futures
WAP	Smart Cards	Education	Cell Phones/	Wireless in Vertical Industries
i-Mode	Wireless LANs incl. Bluetooth	Health Care	WorldPhones	The WWWW
Bluetooth / 802.11	UMTS/3G Networks	Entertainment	PDAs	Unwired Management
Short Messaging	Satellite Broadband	Transport	Headphones/	From 3W to 4W: Issues and Trends
Interactive Gaming		Financial Services	Keyboards /	"Always-On" Management
GPS / Location-Based		Supply Chain Management	Peripherals	Exploiting the Bandwidth Edge
Wireless Java			Transmitters/	Unplugged Valueware
XML & Wireless Technologies			Base Stations	Wireless Sales & Marketing
			Tablets	

FOR INFORMATION CALL

201 802-3069

SPEAKER PROPOSALS INVITED

WWW.SYS-CON.COM

SHAPE YOUR WIRELESS STRATEGY SAVE THE DATES!

EXCLUSIVE SPONSORSHIPS AVAILABLE

Rise above the noise. Establish your company as a market leader. Deliver your message with the marketing support of

SYS-CON MEDIA

SYS-CON EVENTS

THE LARGEST WEST COAST WIRELESS

BUSINESS & TECHNOLOGY CONFERENCE OF THE YEAR!

CORDA Technologies Partners with SunGard

(*Lindon, UT*) – CORDA Technologies Inc. has partnered with SunGard Expert Solutions to embed PopChart Pro charting and graphing software into the SunGard Expert Solutions family of IT solutions for the financial services industry. PopChart Pro creates charts and graphs that enhance the user's ability to understand financial data. Using the pop-up and drill-down functionality, customers can increase their decision-making capacity through clear graphical presentation of the data.

www.corda.com
www.sungard.com

Call for IT World to Aid Businesses

(*New York City*) – INYC, a leading broadband solutions provider, is calling on ISPs, Web-hosting companies, and other tech firms to lend a helping hand to World Trade Center businesses that suffered loss of service.

INYC will focus its efforts on helping companies that lost Internet-related services resulting from the World Trade Center tragedy. INYC is providing free services such as e-mail, Web hosting, Web design, and other Internet-related products for at least until December 18 to any company affected by the network outage. INYC has reserved enough Web space to host 2,000 Web sites, and is asking other technology firms and Internet companies to post at www.silenthelp.com and create the September 11 Internet Fund. "We need all

the help we can get," said Justin A. Kiliszek, VP of business development for INYC. "We are asking any company out there that has the ability to help, please contact us and make a difference."
www.inyc.com/help.html

evolutionB Releases Synergy 2.2 Release

(*Vancouver*) – evolutionB has launched Synergy 2.2, a major maintenance release providing performance improvements, new system administration tools, and enhanced wireless device and database support. Synergy provides nine ready-to-use, collaborative applications, including a contact directory, scheduler, discussion forums, e-mail, and a powerful document manager with version control. In addition, it gives developers an application framework for rapidly developing and deploying robust custom ColdFusion applications.
www.evolutionb.com

Where2Net Becomes Macromedia Partner

(*San Mateo, CA*) – Where2Net, Inc., a leading provider of multichannel development tools and applications, has joined the Macromedia Alliance Partner program. Where2Net will integrate its daVinci Studio development tool with Macromedia products, including ColdFusion and JRun servers and studios. Where2Net daVinci Studio ColdFusion and JSP editions are available for a free download.
www.where2net.com

Macromedia Authorware 6, eLearning Studio Available

(San Francisco) – Macromedia, Inc., has announced the immediate availability of the Macromedia eLearning Studio, which combines the new Macromedia Authorware 6, a visual authoring product for creating interactive e-learning applications, with Macromedia Flash 5 and Dreamweaver 4 to provide a comprehensive authoring solution for e-learning.
www.macromedia.com

activePDF Version 3.5.2 Service Pack 3

(*Mission Viejo, CA*) – activePDF's version 3.5.2 Service Pack 3 has been released, and is available online. It contains over 30 major and minor bug fixes across all products. You can download the service pack directly at activePDF's Web site. www.activepdf.com

ColdFusion App of Choice for 192.com

(*London*) – For UK residents, dialing 192 for directory assistance is as common as 411 for U.S. citizens.

Capitalizing on the ability of the Web to present more information – such as street maps, aerial photographs, and driving directions – and to push that information to advanced mobile phones, i-CD Publishing Ltd. used ColdFusion to create 192.com, the UK's largest directory inquiry service and voted one of the world's top three people-finding sites.

192.com already boasts more than 500,000 registered users – with 25,000 more joining every week – and generates more than 5 million page impressions each month. Because ColdFusion is easy to learn and use, the company can push new services and features faster than ever while employing junior developers. Featuring prebuilt WML templates, ColdFusion enables developers to easily build and deploy wireless applications. www.192.com

Upcoming Events

CF_West
December 2001
Los Angeles
www.cfconf.org

Denver Macromedia Technology Conference
March 14–15, 2002
Denver
www.cfconf.com/denvercf

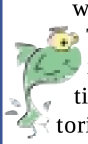
CF_North
May 4–5, 2002
Toronto
www.cfconf.org/CFNorth

CF_Cruise
May 18–25, 2002;
iCaribbean, departs from Miami
www.cfconf.org

CFDJnews

Freshwater Software Integrates with Topaz

(*Boulder, Co*) – Freshwater Software, a wholly owned subsidiary of Mercury Interactive Corporation, has announced an integration with Mercury Interactive's Topaz, thus delivering the industry's first applications to operations monitoring platform for enhanced e-business performance.



Freshwater's newest release of SiteScope, version 6.0, incorporates 10 new application monitors from Topaz and a new high-availability add-on module that allows operations groups to monitor and ensure the health of their applications' servers and gain access to automatic failover capabilities.

SiteScope can now monitor specific applications including Apache, Microsoft SQL Server, Ariba, Netscape Web Server, ColdFusion Real Server, Microsoft ASP, SilverStream, Microsoft IIS, and Windows Media Server. www.freshwater.com

TRG Design Releases TRG-SFA

(*Atlanta*) – TRG-SFA, a Web-based application for managing the sales pipeline, is now available from TRG Design.

TRG-SFA offers an integrated, server-based approach to sales team collaboration and client communications. The system tracks the individual sales pipelines for all sales associates and prompts users with reminders and assistance. Additionally, once the sale is closed, the system initiates the customer service–relationship process in order to ensure client satisfaction. Sales management can extract reports from the database by using a browser-based interface developed by TRG. www.trgdesign.com

CF and JRun Help GlobalSpec.com

(*Troy, NY*) – GlobalSpec.com uses Macromedia ColdFusion and JRun to provide engineers with immediate access to hundreds of thousands of electrical, mechanical, and optical components online.

Before the launch of GlobalSpec.com in 1996, design engineers could spend days thumbing through catalogs. Now they log on to one Web site to search a database of more than 590,000 parts.

GlobalSpec.com sprang out of the need for a specific kind of comprehensive database. Originally CGI based, GlobalSpec.com quickly outgrew that technology.

According to Mark Gaulin, chief software architect at GlobalSpec, the company decided to switch to Macromedia ColdFusion and JRun technologies because they offered both programming and cost efficiency.

Macromedia ColdFusion and JRun have provided a scalable and flexible infrastructure. GlobalSpec's success is tied directly to the performance of its Web site, and Gaulin credits Macromedia solutions for helping the company succeed. www.globalspec.com

CFDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
ABLECOMMERCE	WWW.ABLECOMMERCE.COM	888.801.1333	2
ACTIVEPDF	WWW.ACTIVEPDF.COM	888.389.1653	11
ASSETNOW	WWW.ASSETNOW.COM		39
CFDYNAMICS	WWW.CFDYNAMICS.COM	800.422.7957	13
CFXHOSTING	WWW.CFXHOSTING.COM	866.CFX-HOST	15
CFXtras	WWW.CFXTRAS.COM	704.408.6186	21
COLDFUSION DEVELOPER'S JOURNAL	WWW.SYS-CON.COM	800.513.7111	41
CORDA	WWW.CORDA.COM	801.805.9400	4
EMPIRIX	WWW.EMPIRIX.COM/DOUBLE/CFM	866.228.3781	9
EVOLUTIONB	WWW.SYNERGYANYWHERE.COM		63
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.HOST	19,43
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800.379.7729	64
JAVA DEVELOPER'S JOURNAL	WWW.SYS-CON.COM	800.513.7111	38
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	57
MACROMEDIA	WWW.MACROMEDIA.COM/GO/DEVCON01		35
MACROMEDIA	WWW.MACROMEDIA.COM/GO/USERGROUPS	888.939.2545	37
MACROMEDIA	WWW.MACROMEDIA.COM/GO/MASTERING	877.460.8679	47
MINERVA	WWW.MINERVA.NET	888.667.7231	23
PACIFIC ONLINE	WWW.PACONLINE.NET	877.503.9870	29
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	17
RACKSPACE	WWW.RACKSPACE.COM	888.850.1470	3
RED HORIZON	WWW.SOURCEACTION.COM		55
SYS-CON MEDIA REPRINTS	WWW.SYS-CON.COM	201.802.3026	45
VERTICAL SCOPE	WWW.TOPHOSTS.COM	306.546.3778	31
WEB SERVICES JOURNAL	W.SYS-CON.COM/WEBSERVICES	800.513.7111	51
WEBLOGIC DEVELOPERS JOURNAL	WWW.SYS-CON.COM/WEBLOGIC	800.513.7111	45
WEBSPHERE DEVELOPER'S JOURNAL	WWW.SYS-CON.COM/WEBSPHERE	800.513.7111	56
WIRELESS EDGE CONFERENCE & EXPO	WWW.SYS-CON.COM	201.802.3069	59
XML-JOURNAL	WWW.SYS-CON.COM	800.513.7111	49
YES SOFTWARE	WWW.CODECHARGE.COM	650.754.9810	27

EVOLUTIONB

www.synergyanywhere.com