ADVERTISERS INDEX

ADVERTISER	URL	PH	PG
ABLECOMMERCE	WWW.ABLECOMMERCE.COM	360.253.4142	2-3
ACTIVEPDF			4
AFFINITY	WWW.AFFINITY.COM	800.646.0662	17
ALLAIRE CORPORATION	WWW.VUE.COM/ALLAIRE	877.460.8679	41
ALLAIRE CORPORATION	WWW.ALLAIRE.COM	888.939.2545	35, 55
ANDREWS TECHNOLOGY	WWW.ANDREWSTECHNOLOGY.COM	888.393.0159	31
BIZNIZWEB		281.367.4016	57
CAREER OPPORTUNITIES	WWW.SYS-CON.COM	800.582.3089	52
CF HOSTING	WWW.CFHOSTING.COM	770.982.8382	25
CFXHOSTING	WWW.CFXHOSTING.COM	866.239.4678	13
COLDFUSIONDB.COM	WWW.COLDFUSIONDB.COM		57
CONCEPTWARE AG	WWW.CONCEPTWARE.COM	49(0)6196-4732-0	15
CORDA TECHNOLOGIES	WWW.POPCHART.COM	801.802.0800	23
CYSCAPE		800.932.6369	58
EVOLUTION B	WWW.EVOLUTIONB.COM/SYNERGYFORFREE		59
INTELIANT	WWW.INTELIANT.COM	800-815.5541	33
INTERLAND	WWW.INTERLAND.COM	877.887.3864	7
INTERMEDIA	WWW.INTERMEDIA.NET	800.379.7729	60
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	49
MACROMEDIA	WWW.MACROMEDIA.COM	415.252.2000	18-19
MISTRAL DESIGN GROUP	WWW.PURPLEHOSTING.COM		36
NETMORF	WWW.NETMORF.COM	877.533.MORF	9
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	37
SITEHOSTING.NET	WWW.WEARECFHOSTING.COM	877.NTHOSTING	43
THESHORTLIST.COM	WWW.SHORTLIST.COM		24, 54
UUNET	WWW.INFO.UU.NET/WEBHOSTING.COM	877.465.7107	11
VIRTUALSCAPE	WWW.VIRTUALSCAPE.COM	877.VSCAPE4	27
WIRELESS BUSINESS & TECH	WWW.WIRELESS-MAGAZINE.COM.		53
WOODBOURNE SOLUTIONS	WWW.WOODBOURNESOLUTIONS.COM	877.668.8575	51
XML DEVCON EUROPE	WWW.XMLDEVCONEUROPE2001.COM		47

Please visit the Web sites of our advertising partners who make it possible for us to bring you this Digital Edition (PDF) of *CFDJ*

EXCLUSIVE ALLAIRE CONFERENCE COVERAGE

U.S. $8.99 (Canada $9.99)

ColdFusion Developer's Journal

ColdFusionJournal.com

December 2000 Volume: 2 Issue: 12

December 3–5, 2000
wireless DevCon

Editorial
Live From the Show
Robert Diamond page 5

Foundations
A Gathering of Fuseboxers
Hal Helms page 14

Conference Report
The Allaire Developer Conference
Kevin Schmidt page 30

Book Review
CF Programming Methodology
Nancy Leiby page 40

Interview
Steve Drucker and Dave Gallerizzo
of Fig Leaf Software page 44

IMHO
Making the Case for ColdFusion
Rustin Ross page 56

CFDJ NEWS
page 58

RETAILERS PLEASE DISPLAY UNTIL FEBRUARY 28, 2001
$8.99US $9.99CAN

SYS-CON MEDIA

CFDJ Feature: Toward Better Error Handling Part 2: Site-Wide Error Handling 6
Explore the world of error handling templates *Charles Arehart*

CFDJ Feature: Using XML & XLST with ColdFusion 20
How to set up a working servlet that will read and process it using XSLT *David G. Halsted*

<BF> on <CF>: Java for CFers Part 2 28
Beyond JavaServer Pages, it's all about tags *Ben Forta*

CFDJ Feature: ColdFusion Techniques: Text & Excel ODBC 34
Working within the capabilities and limitations of ColdFusion Express *Raymond H. Thompson*

Q&A: Ask the Training Staff 42
A source for your CF-related questions *Bruce Van Horn*

CFDJ Feature: Matching ColdFusion with Server-Side Java 46
CFML with Java at the server side is easier than you might suspect… *Christian Schneider*

ColdFusion Developer's Journal

EDITORIAL ADVISORY BOARD

STEVEN D. DRUCKER, JIM ESTEN, BEN FORTA,
STEVE NELSON, RICHARD SCHULZE, PAUL UNDERWOOD

EDITOR-IN-CHIEF ROBERT DIAMOND
ART DIRECTOR JIM MORGAN
EXECUTIVE EDITOR M'LOU PINKHAM
MANAGING EDITOR CHERYL VAN SISE
EDITOR/COPY CHIEF NANCY VALENTINE
ASSOCIATE EDITOR BETTY LETIZIA
ASSOCIATE EDITOR JAMIE MATUSOW
EDITORIAL INTERN SUZANNE AUGELLO
PRODUCT REVIEW EDITOR TOM TAULLI
TIPS & TECHNIQUES EDITOR MATT NEWBERRY

WRITERS IN THIS ISSUE

CHARLES AREHART, ROBERT DIAMOND, BEN FORTA,
DAVID HALSTED, HAL HELMS, NANCY LEIBY,
RUSTIN ROSS, KEVIN SCHMIDT, CHRISTIAN SCHNEIDER,
RAY THOMPSON, BRUCE VAN HORN

SUBSCRIPTIONS

SUBSCRIBE @SYS-CON.COM
FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,
PLEASE SEND YOUR LETTERS TO
SUBSCRIPTION DEPARTMENT.

SUBSCRIPTION HOTLINE 800 513-7111
COVER PRICE $8.99/ISSUE
DOMESTIC $79/YR. (12 ISSUES)
CANADA/MEXICO $99/YR
OVERSEAS $129/YR
BACK ISSUES $12 EACH

PRESIDENT AND CEO FUAT A. KIRCAALI
VP, PRODUCTION JIM MORGAN
SENIOR VP, SALES & MARKETING CARMEN GONZALEZ
VP, SALES & MARKETING MILES SILVERMAN
ASSISTANT CONTROLLER JUDITH CALNAN
CREDIT & COLLECTIONS CYNTHIA OBIDZINSKI
ACCOUNTS PAYABLE JOAN LAROSE
ADVERTISING ACCOUNT MANAGER ROBYN FORMA
ADVERTISING ACCOUNT MANAGER MEGAN RING
ASSOCIATE SALES MANAGER CARRIE GEBERT
ASSOCIATE SALES MANAGER CHRISTINE RUSSELL
SALES ASSISTANT ALISA CATALANO
ART DIRECTOR ALEX BOTERO
ASSOCIATE ART DIRECTOR DINA ROMANO
ASSISTANT ART DIRECTOR CATHRYN BURAK
ASSISTANT ART DIRECTOR LOUIS F. CUFFARI
GRAPHIC DESIGNER ABRAHAM ADDO
GRAPHIC DESIGN INTERN AARATHI VENKATARAMAN
WEB DESIGNER STEPHEN KILMURRAY
WEB DESIGNER GINA ALAYYAN
WEB DESIGN INTERN PURVA DAVE
SYS-CON EVENTS MANAGER ANTHONY D. SPITZER

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 $79 BY SYS-CON PUBLICATIONS, INC.,
135 CHESTNUT RIDGE RD.,MONTVALE, NJ 07645

POSTMASTER

SEND ADDRESS CHANGES TO:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON MEDIA, INC.
135 CHESTNUT RIDGE RD.,MONTVALE, NJ 07645

© COPYRIGHT

COPYRIGHT © 2000 BY SYS-CON MEDIA, INC.
ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE
REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS,
ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPY OR ANY
INFORMATION STORAGE AND RETRIEVAL SYSTEM,
WITHOUT WRITTEN PERMISSION.

FOR PROMOTIONAL REPRINTS, CONTACT REPRINT COORDINATOR.

SYS-CON PUBLICATIONS, INC., RESERVES THE RIGHT TO REVISE,
REPUBLISH AND AUTHORIZE ITS READERS TO USE
THE ARTICLES SUBMITTED FOR PUBLICATION.

WORLDWIDE DISTRIBUTION

BY **CURTIS CIRCULATION COMPANY** 730 RIVER ROAD,
NEW MILFORD, NJ 07646-3048 PHONE: 201 634-7400

DISTRIBUTED IN USA

BY **INTERNATIONAL PERIODICAL DISTRIBUTORS**
674 VIA DE LA VALLE, SUITE 204, SOLANA BEACH, CA 92075
619 481-5928

ALL BRAND AND PRODUCT NAMES USED ON THESE PAGES
ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS
OF THEIR RESPECTIVE COMPANIES.

SYS-CON MEDIA

Live From the Show

BY **ROBERT DIAMOND**

I was unfortunately unable to make the Allaire Developer Conference in Boston last year, so I was extremely excited when I found my schedule free and clear for this year's event in Washington, DC. The conference was expanded from last year's, offering longer technical sessions and extending the overall length of the conference as well. Also of note this year was that attendees could take the new ColdFusion Certification Exam in one of several testing rooms on all of the conference days. Well hyped by Allaire to be a sellout for the second year in a row, the conference had me flying into town with high expectations. The question was whether it could live up to all the hype. The answer is a resounding Yes!

A packed show floor with outstanding companies, industry-leading speakers, and over 3,000 attendees from more than 500 companies made it the place to be for anyone developing in ColdFusion today (Spectra and JRun were represented too, of course!). Exhibiting companies were quite pleased with the enthusiasm of the developers in attendance, who were able to gather loads of information on the latest products and services.

Overall, the mood at the show was positively upbeat, which wasn't a surprise. Being able to put faces to names from online forums, and the wide variety of people representing thousands of companies around the globe, all coming together for one common interest, made the trip worthwhile. There was also some excitement about being in Washington during Election Day. This of course led to a few calls for CF guru Ben Forta for President – maybe in 2004?

Another highlight was the **SYS-CON Radio** interviews that I did with representatives from almost all the major companies in the industry attending the show. You can tune into all of these and our complete conference coverage at ColdFusionJournal.com. You'll also see some of the highlighted interviews in the magazine over the next few months.

The feedback on the technical sessions and the buzz on the show floor was that they were the best yet – with both 60- and 90-minute hands-on and workshop sessions on several different topics. The tracks offered covered a wide range of ColdFusion topics: Application Development, Solution Development, Server Management, Technology Integration, and Best Practices – almost everything that developers need to know today.

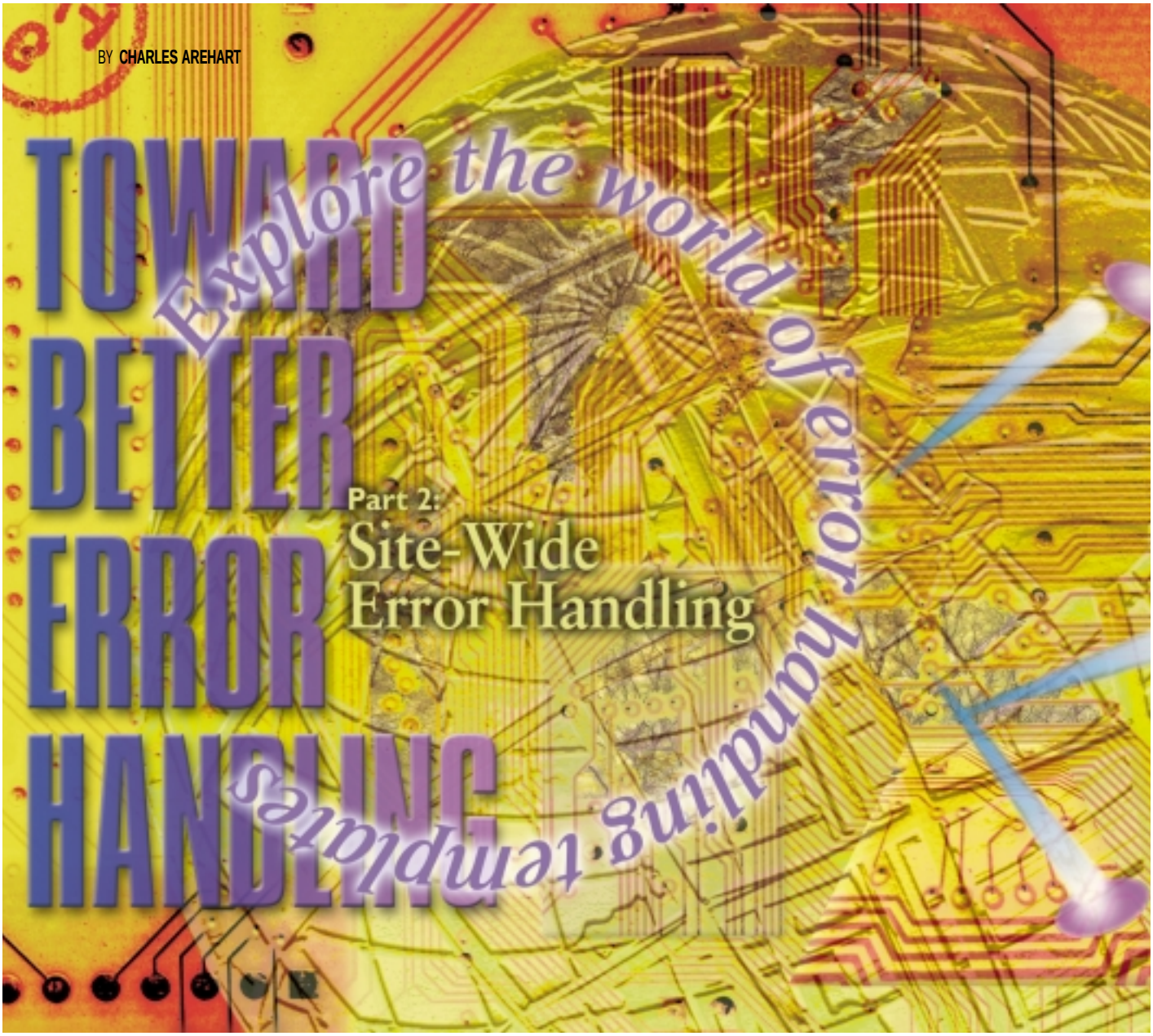
Perhaps the most exciting moments of the conference came during Jeremy Allaire's opening keynote on Monday morning during which he spoke about the course that ColdFusion would be taking throughout 2001 and beyond. Among the several new technologies previewed were ColdFusion 5.0 (to be released sometime in the first six months of 2001) and the next generation of CF (J2EE compatible) Neo coming toward the end of 2001. Look for extensive coverage of these new releases as they enter testing over the next few months. If the enthusiastically cheering audience response as several new (and often-asked-for features) were demonstrated was any indication, the best is certainly yet to come.

So if you didn't make it to the year 2000 event, I certainly hope to see you next year! And to sweeten the deal for all of us, some details about the 2001 Allaire Conference were announced during the long weekend. It will be held at Walt Disney World in Florida, October 21–24, 2001….Mickey and CF – what could be better?

ABOUT THE AUTHOR

Robert Diamond is editor-in-chief of **ColdFusion Developer's Journal** *as well as* **SYS-CON**'s *upcoming magazine,* **Wireless Business & Technology**. *Recently named one of the "Top thirty magazine industry executives under the age of 30" in* Folio *magazine's November issue, Robert is currently a senior in the School of Information Studies at Syracuse University.*

ROBERT@SYS-CON.COM



Did you know that when a CF error occurs, in many cases not only can you cause something other than the plain, black-on-white default CF error message to appear, but you can also log in a database or e-mail to support staff information about the error, including the error message, date, and time? Maybe you knew about that, but did you know that you can implement a change in this behavior for your entire site?

Did you know you can also easily log or e-mail all the session, application, client, and cookie variables in effect at the time of the error as well as the form fields and URL variables that were passed to the program?

Is all this too good to be true? Is it a dream? Not quite. This solution won't solve all your error handling needs, but it's a vast improvement that's been made even better in Release 4.5.

Introducing Error Handling Templates

In this second part of our series on better error handling we'll explore this new and recently expanded world of using error handling templates. These are CF templates that can better manage both the presentation of errors to the user and the back-end processing of errors for administrative and notification purposes.

They've been around in some form for quite a while, but have changed dramatically in recent releases. And they're no longer the only way to improve error handling.

We now have the choice of controlling this error handling at one of three levels of granularity (see Figure 1):

1. Site-wide error handling enabled via an Administrator setting
2. Application-wide error handling enabled using CFERROR

3. Template, or really tag-specific error handling, using CFTRY and CFCATCH

Here we'll focus on just the first approach, as it can be implemented in a way that has maximum impact on an entire site that hasn't implemented (or not fully implemented) the other choices yet. We'll cover CFERROR in the third installment, then CFTRY/CFCATCH in the final.

The Error Handling Hierarchy

It's useful to think of these three levels of error handling as a hierarchy. In most cases CF handles errors by looking up the hierarchy this way, and resolving an error by applying the error handling mechanism that's closest to the line of code causing the error.

(I realize that many readers don't yet know about – or perhaps don't fully understand – CFERROR and CFTRY, so it may seem premature to discuss it. But without at least some introduction, your first attempts to benefit from site-wide error handling may have some surprises.)

By *hierarchy* I mean that if a line or segment of code is enclosed with the CFTRY/CFCATCH tags, then any error occurring in that code will be handled by that CFTRY/CFCATCH processing. Such code could log the error, fire off an e-mail, provide a response to the user, or simply ignore the error.

But if there's no CFTRY/CFCATCH surrounding the code in error or if the CFTRY/CFCATCH doesn't trap the error (perhaps it traps only certain types of expected errors), then the error handling process will look for a CFERROR tag to be implemented for the template being executed. As we'll learn in the next article in this series, this is typically set in application.cfm. If there is one, it will handle the error (again, perhaps formatting a more attractive error message or, in some cases, perhaps even being able to fire off e-mail, log the error, etc.).

If there's no CFERROR or the CFERROR is of a type (discussed in Part 3) that fails to trap the error, the error handling process will next look (in Release 4.5 servers) for a site-wide error handling template, if defined in the CF Administrator. We'll see how to define one in a moment.

One reason to be aware of this hierarchy when learning about site-wide error handling is that it's possible some errors won't trigger your site-wide error handling routine. It may be that it's already handled by either a CFTRY surrounding the code or a CFERROR in place at the application level.

Conversely, it's important to note that not all CF errors follow this hierarchical error handling order. Some CF errors won't be caught by the lower-level routines. Compilation errors are an example. They will, however, be handled by the site-wide error handler.

This is another reason to consider the site-wide error handler approach: it nearly guarantees that the user will no longer see the old black-on-white CF error message. Of course, it's a site-wide solution, so it may not be appropriate in all situations. Let's see how we go about defining a site-wide error handler.

Site-Wide Error Handling

If your site serves just one application or several that are reasonably related to each other, you may want to implement the newest error handling feature introduced in 4.5, called *site-wide error handling*. This feature will enable a CF template of your own creation to be executed when an error occurs (one that's not been handled already by one of the other error handling templates – remember the hierarchy).

You'll see in the CF Administrator that there's a new option, under the server section's Settings page, called "Site-Wide Error Handler" (see Figure 2).

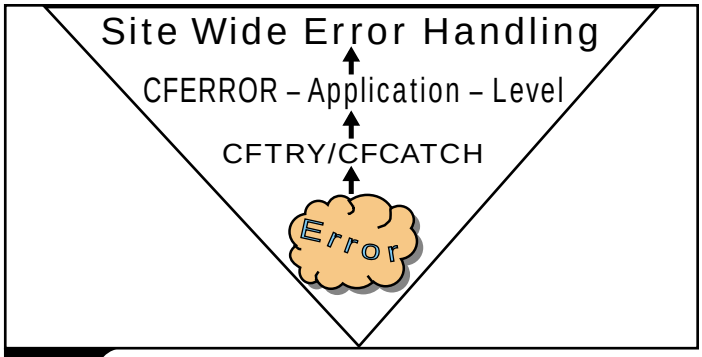


FIGURE 1: Error handling hierarchy

If you don't see an option to identify a Site-Wide Error Handler (and you're indeed looking at the Server>Settings page in the Administrator), you're probably not running Release 4.5 or greater of ColdFusion. This feature exists only as of that release.

Note that this setting expects a complete physical path to the template, not a Web-server relative path, so if the file is called site_error_handler.cfm and is located in your Webroot directory, c:\inetpub\wwwroot, you would identify it as c:\inetpub\wwwroot\site_error_handler.cfm, not \site_error_handler.cfm or simply error_handler.cfm. It's surprising that Allaire didn't include the usual Browse button found in other Administrator dialogs for naming templates. Perhaps it'll be added in the next release.

Try It Out

There's no better way to learn than by doing. If you have access to a testing server (or your own local workstation has ColdFusion Server installed), why not try out how this site-wide error handling works. We'll start with a very basic template that does nothing more than flag that an error has occurred. (We'll learn shortly how we can do much more.)

Create a template called site_error_handler.cfm. There's nothing magic about the name. You can use any name for the site-wide error handler, and where you place it is also up to you. It's not important that it be placed in a Web-accessible directory. Just keep in mind where you've placed it.

What you put in for display to the user is entirely up to you. For now, just put in something like this:

```
<h2>An Error Has Occurred</h2>
It has been trapped by the site-wide error handler in
c:\inetpub\wwwroot\site_error_handler.cfm.
```

It may seem needlessly wordy to indicate where the template is that's handling the error, but if you leave this set and forget it (or set it on a testing server where others are affected), it'll be helpful to know why the normal CF error message isn't showing and how to fix it. We'll leave it like this only momentarily. You may even want to browse whatever template you do create to make sure it appears as you'd expect (and make any corrections) before setting it to be the site-wide error handler.

Now, use the Administrator setting shown above to assign this to be your site-wide error handler. (Be careful not to do this on a production server. And if it's on a testing server used by others, you may want to wait until the end of the article to try it after we learn how to display more useful information, such as the error message itself.)

Finally, to test it out, create and execute a template with a ColdFusion error in it. It can be simply a compilation error, such as trying to use the tag <CFIF>, which is obviously incomplete on its own. This should generate an error that calls up the display not of the normal CF error message but your error handling page.

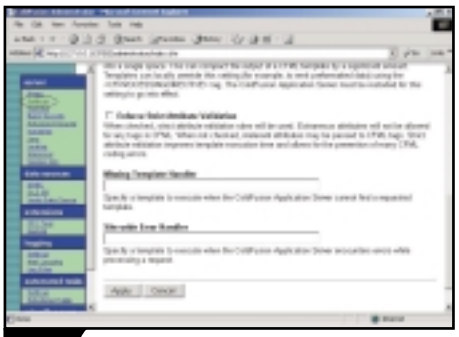


FIGURE 2 Administrator setting for site-wide error handler

If the error handler doesn't catch the error, make sure of a few things: Did you apply the change to the Administrator setting? Are you sure the error handling template exists and the path name is correct? If not, there won't be an error when you apply the Administrator setting; instead the following message will display whenever an error (such as our <CFIF> test here) occurs:

```
Cannot open CFML file

The requested file c:\inetpub\wwroot\
demo\site-wide-error.cfm cannot be found.
```

In this case, my mistake was in the misspelling of the directory name, wwwroot. The error message indicates I mistakenly left off a "w" in wwwroot. If all seems correct and the error isn't being caught by the site-wide error handler, make sure there's no CFERROR currently set in the application.cfm of the directory in which your testing code (the code having the sample error) has been placed. If this is the case, move the error testing code to another directory with no CFERROR set for it and try again.

If all still fails, you may want to remove the change in the Administrator identifying the site-wide error handler. As long as it's set but the process isn't working as expected, you may impact the ability of other developers or users to see expected error messages. Carefully retrace your steps to see why things aren't working as expected.

Now that we've confirmed that the site-wide error handler works, let's add more to it. We can choose to display the error message to the user, hide it, or do pretty much anything we want.

Greater Error Handler Capabilities

Indeed, one of the powerful new features of the site-wide error handler (as well as some new forms of CFERROR) is that the template can execute all CF tags and functions and has access to all CF variables.

You can now use CFMAIL within the template to send e-mail to support staff or use CFQUERY to store a database record. You're no longer reliant on the user to inform you of the error, and you need not rely on rummaging through CF's error logs to learn of one. You're finally able to know of the error *before* the user complains about it.

What's more, when you're able to access all the CF tags, functions, and variables, you can choose to display, e-mail, or log information that can greatly increase the ease with which you can diagnose errors. How wonderful it is to finally have access to any CF variables, such as session, cookie, form, and all the other types. We'll learn more about that in a moment. Let's first consider the simple prospect of improving the error message shown to the user.

Improving Error Appearance and Usefulness

Perhaps the most obvious drawback, from your site's perspective of the default CF error message, is its jarring effect on your Web application interface. It's bad enough that your code has failed (but hey, we all make mistakes). Isn't it a shame, though, that your otherwise colorful and well-designed interface is suddenly interrupted by a jarring page with white background and black Times Roman text that screams a less than useful message to your user?

Haven't you wished you could at least show the error message on a page with your site's normal background color and whatever font you use? Perhaps with your company logo and maybe the phone number for your help desk? How about continuing to show the user your navigational toolbars? You can do all this, quite easily in most cases.

Figure 3 shows an error message appearing at half.com, which happens to be a ColdFusion site. I experienced the error recently, and it's clear they're using error handling templates to present an attractive alternative to the standard CF error message.

Notice that the site's normal colors, background, navigational toolbar, and help links are still displayed to the user. Interestingly, they've chosen not to show the exact CF error message. Whether that's the course you take is up to you. The good news is that you can either show them the error or not, and/or (using the site-wide error handler and some forms of CFERROR) you could send the error in an e-mail to support staff.

The trick to accessing the actual CF error message is in knowing about and using special CF error variables available to such error handling templates.

Error Diagnostic Variables

In our sample, site_error_handler.cfm, as well as the half.com example, no details are shown about the error that triggered the error handler. The error itself is indeed available in a variable called #error.diagnostics#, and you can choose what to do with it, if anything.

This variable is only available within error handling templates. It contains the complete text of whatever error has occurred. There are several other variables available as well to report the date and time of the error, the template name in error (unless hidden from display by other Administrator options), the query string passed to the page, the page that called the one in error, and even the user's IP address and browser (user agent). We'll learn later about the mailto variable and how it works. See the sidebar on error handling variables.

Capturing Other CF Variables

As mentioned before, you may also be able to access all other CF variables, such as session, client, application, form, url, and cookie. These are always available in the site-wide error handler and in newer forms of the application-level error handlers (via CFERROR), discussed next month.

As always, you can show these variables (the error variables or any CF variables) to the user by way of CFOUTPUT. You can also refer to them within a CFMAIL or in a CFQUERY to store data in a database record.

Showing all the variables in effect at the time of an error is a phenomenal capability. How often have you had an

ERROR HANDLING VARIABLES

- Error.Diagnostics
- Error.DateTime
- Error.Template
- Error.QueryString
- Error.HTTPReferrer
- Error.Browser
- Error.RemoteAddress
- Error.MailTo

See the Allaire documentation for more information on these variables. You'll find them discussed under the CFERROR tag. They apply to the site-wide error handler as well.

error arise and been challenged to determine the reason knowing only the template name? Perhaps the error was caused by form variables passed (or not passed) to the program or session variables, not set as expected.

You may be thinking that a major drawback to this concept is that you'd need to know in advance what variables you'd want to process and therefore couldn't capture all the variables in effect at the time of the processing of the error handler. More great news: you can!

Looping Through All Variables of a Given Type

You may know that Allaire has been making (in each release) more variables available as CF "structures." Early on, we could process application and session variables this way. As of 4.5, you can also access url, cgi, form, and cookie variables as structures (server variables are still not available, partially for security reasons).

One of the great things about being able to access these variable scopes as structures is that we can use the CFLOOP tag, using its COLLECTION attribute, to identify the name and value of all variables in a given scope. The following shows an example of one way to loop through all session variables:

```
<b>SESSION VARIABLES:</b>
<cfif isdefined("session.cfid")>
 <ul>
 <cfloop collection="#session#" item="i">
  <li>#i#=<cfif
issimplevalue(evaluate("session."&i))>#evaluate("session."&i)#
<cfelse>Unprintable Value</cfif>
 </cfloop>
 </ul>
<cfelse>
 None
</cfif>
```

This first tests if session variables exist for the current user (if so, a session variable called CFID will exist), then loops through each existing session variable, printing its name and value. If the value is not printable, this is explained. This could occur if the variable is itself a structure or perhaps a query. One way to better handle this problem is to use the custom tag CF_OBJECTDUMP, available free at the Allaire Developers Exchange. It can recursively display structures within structures.

The code example above may be a bit tricky on first viewing, but it works. Putting this sort of code block for each of the available variable scopes would be a great addition to any error handler. Again, whether you choose to display those, send them to support in an e-mail, or store them in a database record is up to you.

Note: While client variables can't be processed as structures, there's a function getclientvariableslist() that can be used to achieve a similar objective. It will return the empty string if there are no client variables.

"Admin E-mail" Address Also Available

In the previous article I mentioned the possibility of defining an e-mail address in the CF Administrator that would be displayed in the default CF error message shown. Users could then click on a link to e-mail that person about the error.

Of course, with the site-wide error handler's ability to send e-mail on its own, it's not as important to show that sort of mail link to the user, but you may still prefer to use such a global setting to indicate who should receive the error notification.

The good news is that if this admin e-mail value is set in the Administrator, it will be available as an #error.mailto# variable in any site-wide error handler.

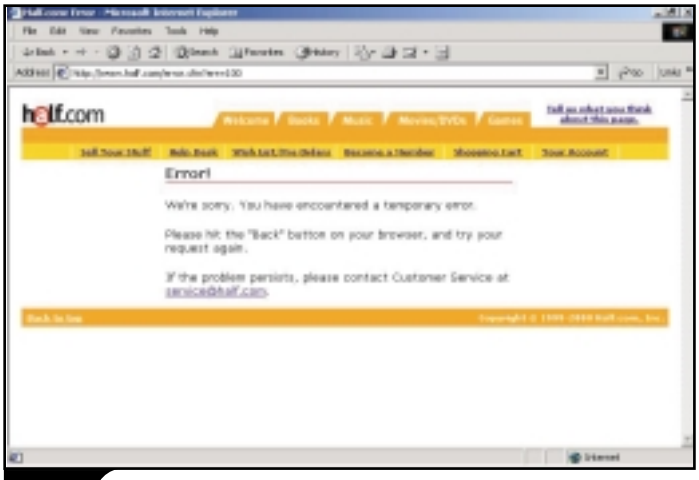


FIGURE 3: A more attractive error message at CF site half.com

Errors in the Error Handler

Finally, those who've been using CF for a while may remember that early forms of error handling templates, namely the CFERROR TYPE="request" and TYPE="validation", didn't allow you to execute CF tags or functions. You could only display the error variables mentioned above with a sort of presumed CFOUTPUT that allowed their use.

The thinking, it was said, was that enabling CF tags and functions (and other variables) in the error handler could raise the dilemma that the error handler could have an error itself, and the system might get stuck in a loop of the error handler trying to process the error handler. It never quite made sense to me (and may have been an apocryphal explanation for the lack of access to tags and functions), as it seemed Allaire could detect an error in the error handler and simply abort the error handling process altogether.

The good news is that this is indeed the tack Allaire takes when using the site-wide error handler. If there's an error in the handler itself, then the user will simply be shown the plain old CF error message in the black on white that we're all accustomed to.

Summary

There you have it. You should now have a better understanding as well as a working version of a site-wide error handling template. You've learned about the hierarchy of error handling mechanisms and how to catch errors that haven't been otherwise caught by error handling mechanisms at the template or application levels.

You've learned how to access a host of error diagnostic variables, and perhaps more valuable (for all your coding), you've learned how to loop through the various CF variable scopes to be able to report on all the session variables – or nearly any other scope – in effect at the time of the error. And you've learned what happens when the error handler itself has an error.

For all its capability, the site-wide error handling capability alone is not the only solution to implement to achieve "better error handling." Due to its global nature, it may not always be the appropriate (or only) solution. Next month we'll continue the discussion with the next step down the hierarchy: application-level error handling via CFERROR.

About the Author

Charles Arehart is a certified Allaire trainer and CTO of SysteManage, an Allaire partner. He contributes to several CF resources and is a frequent speaker at user groups throughout the country.

carehart@systemanage.com

A Gathering of Fuseboxers

Being there – almost as much FUN as getting there

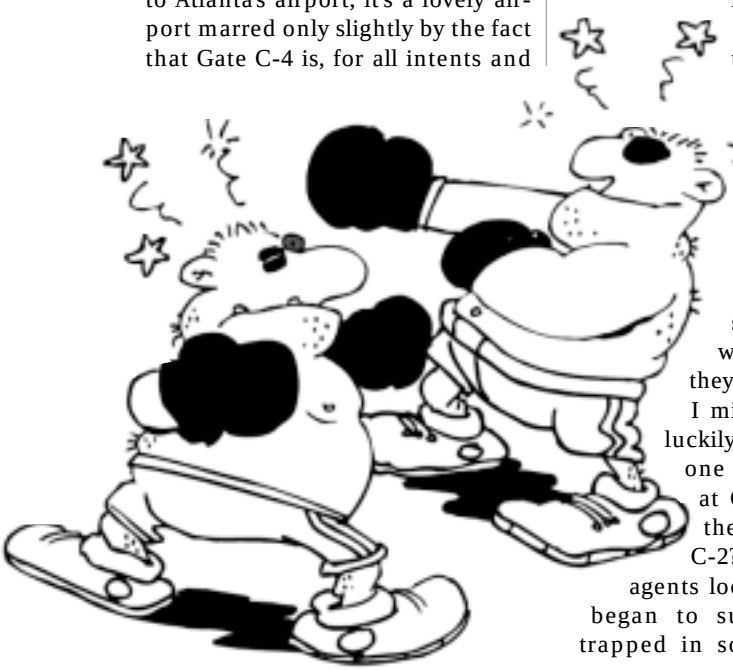
BY HAL HELMS

My troubles all began when my editor heard about the Fusebox FUN conference being held on September 16 in Charlottesville, Virginia. "You're the perfect person to report on it," she said convincingly. "After all, you're already going to be there doing a presentation."

There are times, I now know, when even the most trusted editor is forced, for the sake of a story, to make a pact with the forces of darkness. But when I agreed to her idea, I didn't realize this. In my defense there had never been any indications. Her eyes never glowed red and I've never seen her head spin around. And so I set forth – a trusting lamb – wending my way to Virginia.

Coming from Atlanta, it seemed that the shortest route would be to fly to Washington, DC, and drive to the conference. When I got to the airport – a little late, admittedly – I distinctly remember seeing a sign telling me that my flight was leaving from Gate C-4. If you've never been to Atlanta's airport, it's a lovely airport marred only slightly by the fact that Gate C-4 is, for all intents and purposes, at the other end of the Milky Way. When I finally got there, a friendly customer service representative asked me if this would be my first visit to Toronto. "Is that where I am?" I gasped. "I knew I must have passed the U.S. border."

"No," she giggled. "The flight to Toronto. Flight 1281. Your flight." My flight? My flight was 710 to Washington-Dulles. "Nope," she told me definitively. "Flight 710 is at Gate C-31. If that's where you're headed, you better hurry. It leaves in a couple of minutes." This was news akin to finding out that the Boston Marathon you've just completed in record time was actually held yesterday. Gazing toward the far end of the concourse, I could see the horizon but not my gate. Wondering how many time zones I'd be passing to get to my gate, I retraced my steps. I think that was when it first occurred to me that supernatural forces were at work – and they weren't on my side.



I missed that flight, but luckily there was another one in only 50 minutes at Gate C-2. "When is the next flight to Gate C-2?" I asked, but the agents looked at me oddly. I began to suspect that I was trapped in some Twilight Zone episode where for all eternity I would be a modern-day Sisyphus, forever pushing my laptop and carry-on along an endless corridor. So I was actually very happy when the plane left, only an hour and 20 minutes late.

When I got to Dulles, I went to pick up my car. "Don't have it," the less-than-friendly customer service representative informed me.

"Why not?" I half-demanded, half-pleaded.

"Your reservation was hours ago. We figured you were a no-show. Rented it to somebody else."

"Don't you have anything else?" I asked.

"Well…we do have one car left."

"I'll take it!" I fairly shouted.

"Yeah, uh, okay. Well, I can give you a deal on this car." I should have noted his hesitation. I do think I noticed his eyes had a distinct reddish tone to the pupils, but I passed that off to the semi-hallucinatory state I had lapsed into.

I never could identify the maker of the car I rented. I thought this odd at first. Later, I recognized it as a savvy marketing move – there are some things consumers don't really need to know. I had about a three-hour drive to Charlottesville and after an hour or so, I realized my arms were aching from trying to bend the car's natural tendency to drive at a 30-degree angle from the center stripes on the road. But I made it just fine and drove into Charlottesville just in time for the start of the conference.

Charlottesville is a beautiful place. It's the home of the University of Virginia, a college that owes its architecture to Thomas Jefferson. I drove into the hotel, tired but glad I'd made it. "You have a computer conference this weekend? A Fusebox conference?" I asked the customer service representative at the hotel. She looked at me oddly and asked me to wait a minute. She disappeared, leaving me to wonder if I'd accidentally stumbled onto a known code phrase used by terrorists.

The manager came out and after a few more minutes of me trying to look as if the ominous black bag I was holding was really just a laptop, he explained that there was no conference there that weekend. "You probably mean the Holiday Inn," he offered. "They have lots of conferences there." I thanked him profusely and headed for the Holiday Inn and the Fusebox FUN conference.

The customer service representative at the Holiday Inn was, if anything, even nicer than at the first hotel. "No…" he said. "No Fusebox conference here." Note to self: find out where you're going before you get there. I realized that in my hurry I'd never gotten to the point of actually looking on the actual site to see where I was supposed to be.

I called a friend. "Quick, get on the Internet," I barked. "Go to fusebox.org and tell me where I'm supposed to be." I had only 40 minutes until I was scheduled to speak. Wonderfully, miraculously, he found the hotel, and I was there in 10 minutes. I thought about calling my editor. I'd been practicing what I might say to her – "I'm on to you! I know about the pact you made with the devil!" – but I didn't want to miss the first speaker. Fred Sanders was speaking on Fusebox fundamentals. I could settle scores with the forces of darkness later; right now, I was a reporter.

Fred did a great job of explaining the theory and practice of Fusebox. He explained why it was created (cheaper, better, and faster code) and why it was catching on so quickly (it works).

I followed next with a talk on Fusedoc. As I talked with other developers about creating a standard that was both useful and usable, I was struck by what a unique thing I was witnessing: 150 people had traveled from all over the country, giving up their weekend to meet with other developers as passionate as they were about becoming better developers – sharing with and learning from each other. The logistics for this event were not handled by a company hoping to hawk products, but by Steve Nelson and Michael Smith who had arranged for the meeting space, box lunches, handouts, badges – even a beer bash! – and did so out of that same desire.

Next, Nat Papovich spoke on using Fusebox with frames. Nat demonstrated such virtuosity in his use of frames and had so many nested framesets that I believe one particularly complex set had a frame that contained itself. It was most impressive. More to the point, it was most instructive. I've seen people struggle with frames and Fusebox, and Nat's presentation demystified the process.

Lunch followed, and I was feeling considerably more sanguine about the experience. I was even feeling so charitable as to consider that my editor might only have made a horrible, horrible mistake rather than actively participating with demonic forces.

Steve Nelson was the next presenter and his topic was "The Need for Speed." Steve's idea of speed is a little different than what most of us are used to, as he related how he'd used Fusebox to build a complete app in five days. The trick, Steve said, was not to view an application deployment as a single monolithic event, but to think in terms of multiple deployments with each one incorporating more functionality and fewer bugs. I found this a very interesting topic, as I have long felt that adopting what I call the "Big Bang Deployment" model inevitably reveals vital feedback from users that would have been key had we found a way to solicit it earlier.

Gabe Roffman was the next speaker, and his topic was "The Fusebox Application Object Model." Gabe explained that he wanted a standardized way of creating application-level objects that would provide a standard template for all the global developers on a project. He pointed out that using request-scoped variables in lieu of application variables provided for global scope without persistence – a good choice for many application variables.

Steve Nelson was back next, explaining how Fusebox applications can be secured on multiple levels. For many developers, securing an application proves a daunting challenge. Steve's presentation laid out clearly how this could be accomplished easily and how the security framework could be reused on any Fusebox application.

Finally, Doug Nottage of Autobytel.com spoke on scaling ColdFusion applications. "Scalable ColdFusion applications scale in a scalable environment," Doug stated, then proceeded to explain how to create that environment as well as how to write code that would scale within it. His talk was very well received, as developers learned techniques for optimizing their applications.

A drawing for an enterprise edition of ColdFusion (graciously donated by Allaire) was held using – what else? – a Fusebox application written by Steve Nelson. I can vouch for the fairness of the drawing, as Steve resisted my suggestion that I could make it "worth his while" if the application happened to pick me as winner. Sadly, I must report that

ABOUT THE AUTHOR

Hal Helms is a Team Allaire member living in Atlanta, Georgia. A frequent writer on ColdFusion and Fusebox, he also offers training and mentoring on these subjects.

Steve was unable to see the wisdom in my proposal.

Steve's company, secretagents.com, also donated a Handspring Visor to another lucky winner. Let me take this opportunity to refute the reports that have arisen from the conference that I was "pouting in the corner" over not winning this drawing either. I may have been standing in the corner, but there are several excellent reasons why I might have been there aside from the slanderous ones advanced by my detractors. As soon as I can think of two or three good ones, I plan to write a separate article explaining them.

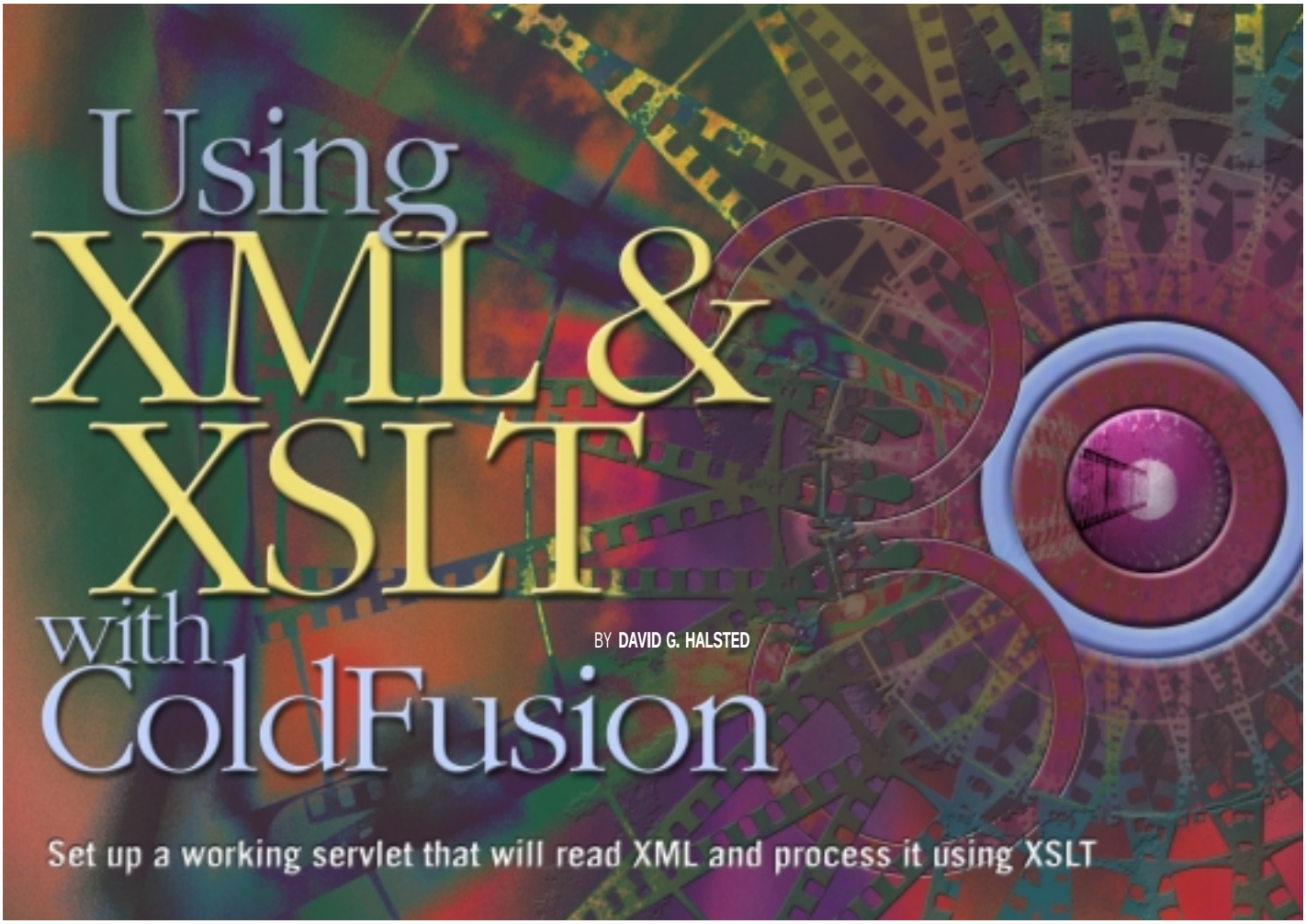
The hotel staff finally had to throw us out – we developers are a devoted lot – as they had to prepare for a large dinner to be held where we were meeting! The conference then adjourned to a local hangout where foosball and beer were the chief items of interest.

In all, it was a remarkable event. Kudos, again, to Michael Smith and Steve Nelson for their selfless efforts in organizing the conference. All the participants I spoke with were effusive in their praise for the presenters and the ability for like-minded developers to gather and talk shop. Even though this one was put together on fairly short notice, it sold out very quickly and many who wanted to come couldn't be accommodated. The best news to come out of the conference was that another, larger one is being planned. Look for news of it on fusebox.org.

As I headed back to Washington-Dulles, I reflected on just what an extraordinary event it had been. I felt privileged to be invited to speak. It began to occur to me that my difficulties getting there might have been simply a run of bad luck. The more I thought of it, the more likely this seemed. After all, I knew my editor to be a fine person and the thought of her conspiring with evil forces just to get a story – well, in the light of the conference, it seemed most unlikely.

As I arrived at the gate for the flight that would take me back to Atlanta, I was very glad that she had asked me to report on the conference. The customer service representative interrupted my thoughts. "Mr. Helms?" she said. "Umm….The flight left a half-hour ago." Shocked out of my reverie, I stared agog at her. Her smile darkened into an evil grin as her head began spinning around…

HAL.HELMS@TEAMALLAIRE.COM



Suppose you're approached to build a distributed Web back end for TechnicalNotes.com. TechnicalNotes.com wants its Web site to display brief pages summarizing recently published articles on various Web technologies. The notes are written by authors scattered over a wide area using many different platforms, so the company has chosen to store their submissions in XML. TechnicalNotes.com has contracted with another company to collect and store data about the articles themselves, which will be stored in XML on another server in a remote location.

Provide Code That's Easy to Maintain

Your task is to parse the XML pieces on the fly and combine the summaries on the TechnicalNotes.com server with the data on the remote server. You need to provide the customer with code that's easy to maintain and change. TechnicalNotes.com doesn't have a lot of Perl or Java programmers capable of manipulating XML with programming. And to top it off, they want the information displayed within a ColdFusion template because they already have a complex security system written in CF. How can you provide TechnicalNotes.com with a system that meets these requirements?

To meet the requirements for the TechnicalNotes system – or to integrate any ColdFusion system with XML – you need to bring data stored in XML into a format that ColdFusion can display or recognize and process. The CFServlet tag in ColdFusion 4.5, in combination with JRun, lets you integrate ColdFusion applications with servlets. One way to bring XML into a ColdFusion application for display or processing is to use a Java parser to process the XML and feed the results into the application using CFServlet.

Create a Powerful Environment

This article will show you how to set up a working servlet that will read XML and process it using XSLT. On the Java side, we'll use an open source XML/XSLT processor called *Saxon*. You'll want to have a development environment that lets you write and compile servlets, and you'll need JRun and ColdFusion 4.5. Once these elements are put together properly, they can be used to create a powerful environment that integrates CFML and XML.

One of the key pieces of the environment we'll build is XSLT, or eXtensible Stylesheet Language Transformations. XML documents can be queried using XSLT just as relational databases can be queried using SQL; XSLT can sort, run conditional queries, create key structures, and add identifying numbers to rows of output. XSLT can also be used to transform an XML file into many different kinds of output, from plain text to WML. In conjunction with FOP, now a part of the Apache project, it can even be used to create PDF files. (More information is available at http://xml.apache.org/fop/index.html.)

XSLT – Easier to Learn Than Java or Perl

You can use XSLT on the server to batch-convert a set of files, or you can process XML through one or more XSLTs via a servlet and return the results to a Web browser or a wireless device. Once you realize how many things XSLT can do, you'll begin to find lots of ways to apply it. You don't have to use XSLT to process XML, but it's easier to learn XSLT than Java or Perl, so you can let nonprogrammers manipulate output more easily if you filter it through XSLT. For overviews of XSLT, see the W3C spec at

www.w3.org/ TR/ xslt.html or G. Ken Holman's introductory essays on xml.com (www.xmlcom/pub/2000/08/holman/index.html). (For more in-depth information, see Michael Kay's *XSLT Programmer's Reference*, published by Wrox.)

To meet the requirements set out by TechnicalNotes.com, you settle on Saxon to process the XML sources using XSLT stylesheets. Saxon is available at http://users.iclway.co.uk mhkay/saxon. A command-line version of the program called *InstantSaxon* is available for Windows. InstantSaxon is a useful way to get started with XML and XSLT, but if you want to deliver Web pages on the fly with Saxon, you'll want the full version.

Downloading Saxon

Once you've downloaded Saxon, decompress the zip file and follow the installation instructions. You'll also need a Java development environment that lets you write and compile servlets. An easy way to get started is to download recent versions of Sun's JDK and JSWDK from http://java.sun.com; Sun provides full installation instructions for various platforms, and the site includes helpful online tutorials. Everything is in place when you can type "java.com.icl.saxon.- StyleSheet" at a command prompt and you get an error message from Saxon itself. If you get a Bad Command or File Name error, you need to reset your path – the JDK installation instructions help with this – or type the full path name to java.exe in the command itself. If you get an error indicating that Java can't find a class, you need to fix the classpath, or use the -cp option when you invoke Saxon from the command line (see Java documentation). It may take a little trial and error to get all the pieces working properly in your environment.

To see what Saxon does, try some of the samples that come with it. Instructions are in the doc subdirectory under the directory in which you installed Saxon. You can also experiment with the files included in the code listings at the end of this article. First, save the XML and XSLT sources in Listing 1 in the same directory using the suggested filenames. Then cd to that directory, and type

```
java com.icl.saxon.StyleSheet javascriptarticle.xml simple_article.xsl
```

at the command prompt. You should see an example of how XSLT transforms XML on your screen. *Hint:* If you're doing this on a machine that can't remember previous command-line input, you may want to put the command into a .bat file so you don't have to type it over and over.

Using Saxon with ColdFusion

To use Saxon with ColdFusion, you also need to install JRun on your machine. If you're only interested in displaying the results of an XML/XSLT transformation, JRun 2.3.3 is fine, but if you want to write servlets that change the values of CF variables, you'll need to upgrade to JRun 3.0 (a developer's version is available for free on the Allaire Web site). Once JRun is installed, you should ensure that it's operating properly. If you're using 2.3.3, you can check the HelloWorldServlet that ships with JRun by hitting http://<hostname>/servlet/HelloWorldServlet. 3.0 is a little more complex, but if you look in the QuickStart documentation you'll find a sample JSP page you can use to confirm that the JRun servers are running properly. The JRun documentation will also explain JRun's structure and tell you where to place new servlets; that's where you want to put the .class files you compile from the source in the code listings.

Once you've got JRun configured the way you want it, make sure it's communicating properly with servlets you add. Listing 2 is a simple HelloWorld style servlet that can accept an optional parameter from a CFServlet tag. To test Listing 2, type it into a text file called "CFTestServlet.java." At a command prompt in the same directory type "javac CFTestServlet.java" and hit enter. Then look at the files in that directory; you'll see that javac has created a file called CFTestServlet.- class. If you're using JRun 2.3.3, copy CFTestServlet.class to the "servlets" directory under the JRun root; if you're using JRun 3.0, copy it to <jrun-root>/servers/ default/default-app/ WEB-INF/classes. If compilation is successful, you'll get a UNIX-style response – nothing. If you get an error indicating that the command wasn't found, type the full path to javac at the command prompt or edit the path variable (how you do this depends on the OS you're running). You may get an error indicating that the compiler can't find a package starting with "javax." If this happens, you need to edit the classpath to include the appropriate classes. With Sun's JSWDK, for example, you need to have the line <JSWDK root>/lib/servlet.jar on the classpath. You may also invoke the proper classpath at compile time using the –cp option.

Now check to make sure you can see the servlet. If you set up JRun to communicate with your Web server, you can do this by typing http://localhost/servlet/ CFTestServlet; otherwise you'll have to include the port number. You should see a message that identifies itself as coming from CFTestServlet. That tells you that CFTestServlet, at least, is working fine.

Ensure Good Communication

The next step is to ensure that servlets and ColdFusion are communicating properly. This means writing a page that calls the servlet using the CFServlet tag. Simply place

```
<cfservlet code="CFTestServlet">
 <cfservletparam name="testParameter"
value="Greetings from ColdFusion">
</cfservlet>
```

in a ColdFusion template and open the .cfm with your browser. In this snippet, the CFServlet tag invokes CFTestServlet and passes it a name-value pair, just as you might pass a name-value pair in a URL request. Use CFServletParam to pass values to the servlet. The servlet catches the parameter's value, processes it, and returns a string; the CF template delivers the string to the browser. If you have set JRun up to work on a remote server, you'll need to use CFServlet's JRunProxy attribute. The servlets in this article merely return strings. With JRun 3.0, however, you can use a name-variable pair instead of a name-value pair in CFServletParam. This makes it possible to manipulate CF variables from the servlet itself. See the JRun documentation for an extended example of this powerful new feature.

CFServlet gives CF developers the potential to handle XML in several different ways. Data stored in XML might need to be read and edited by a user before it's entered into a database on a record-by-record basis. In this case, a servlet could be created that would read the XML and use XSLT to place the appropriate data into an online form for editing. A regular XML feed might need to be dumped wholesale into a database, but only on user request. To accomplish this, a developer could write a servlet that would do the data dump itself using JDBC, but set up the front end using ColdFusion. It would also be possible to write a servlet that would look for particular values in the XML, assign those values to ColdFusion variables, and pass the variables back to the calling template for processing and entry into the database. This approach might be useful in a shop with few XML/Java developers but lots of ColdFusion people; the servlet could be treated as an object and called from many different contexts.

Now that we've got servlets running and talking to ColdFusion, all we need to do to get our XML into a ColdFusion template is run Saxon as a servlet. Listing 3 is a simplified version of the servlet that's included with the Saxon package. For production

work you'll want to use something with the functionality of the original servlet. Most important in the original is the feature that caches the XSLT once it's been used; this behavior can lead to dramatic increases in performance. Of course, nothing prevents you from using another XSLT processing program if you wish.

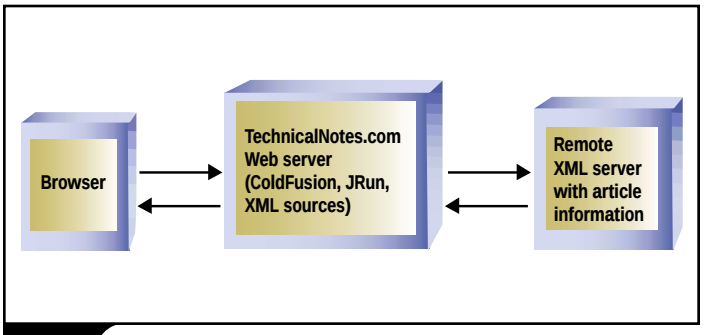


FIGURE 1: The TechnicalNotes.com production system

Set Classpath for Servlets

To experiment with the code in Listing 3, copy the code into a file called MiniSaxonServlet.java and compile it. The Java compilation program will create another file called MiniSaxonServlet.class. Copy MiniSaxonServlet.class to the same directory as CFTestServlet. You need to make sure that both saxon.jar and the directory containing saxon.properties file are on the classpath for the JRun server you're using. As a security measure, the programming that actually delivers servlets to the browser doesn't read your system classpath or the classpath for any particular user. Instead, you must explicitly set the classpath for the servlets. In JRun 2.3.3, the classpath can be set using the general Java settings in the Administrator. In JRun 3.0, edit the classpath for each server in the "Java settings" section for that server in the Administrator; you need to include saxon.jar and the directory containing saxon.properties in the classpath of each server that will use Saxon. You must also compile MiniSaxonServlet from an account that has these on the classpath, or use the –cp option on the javac command line. If you make changes to MiniSaxonServlet.java, remember that you have to recompile the source and copy the resulting class to the appropriate directory or the changes won't be reflected in your output.

If you haven't yet created the XML and XSLT files in Listing 1 to another directory, do so now. Make sure that the directory in which you place these files can be read by MiniSaxonServlet. MiniSaxonServlet is designed to be invoked with three parameters:

- ***rootpath***: The directory in which MiniSaxonServlet will look to find the XML and XSLT files
- ***source***: The XML file MiniSaxonServlet will transform
- ***style***: The XSLT file (with the extension .xsl) MiniSaxonServlet will use to perform the transformation

The ColdFusion code in Listings 4 and 5 assumes that you'll place these files in a directory called "C:/cfdj" and that you'll use the suggested file names. You're welcome to use another directory and different names for the files, but you'll need to adjust the "value" attribute of the corresponding CFServletParam elements or you'll get an error.

If you can run a successful ColdFusion template containing Listing 4, you've created an environment in which XML, XSLT, and ColdFusion can work together. You can now process and display XML on the fly from within a ColdFusion application. You should also be able to run MiniSaxonServlet directly by including the parameters in the command line:

```
http://<hostname>/servlet/MiniSaxonServlet?rootpath=C:/cfdj/&source
=javascriptarticle.xml&style=simple_article.xsl
```

The HTML that displays in the browser in response to this URL is generated via the XSLT transformation performed on javascriptarticle.xml. If you only plan to call MiniSaxonServlet from within a CF page, you'll get better HTML if you delete some of the HTML tags in the .xsl files.

At this point, the bulk of the system needed to meet the requirements of the TechnicalNotes.com Web site is in place, but one element is still missing. In the TechnicalNotes.com scenario, some of the XML we wanted to read was located on a server remote from the systems running ColdFusion and JRun. XSLT lets us accomplish this easily. article_style.xsl invokes the same XML source document as simple_article.xsl, but it includes a variable called "articleList" that's initialized using the document() function. In XSLT, document() can be used to read XML from another file and process it, even if that file is on another server. article_style.xsl assumes that the file "articles.xml" is in the same directory as your other XML and XSLT files. However, if you put a copy of articles.xml on another server, you can substitute:

```
<xsl:variable name="articleList"
select="document('http://<remoteserver[/path]>/articles.xml')/arti-
cles" />
for
<xsl:variable name="articleList"
select="document('articles.xml')/articles" />
```

If you do this, you'll find that the code still works, though there may be a delay as the XSLT processor reaches out to find the XML source. This example shows how XSLT can be used to extend the functionality of ColdFusion. With this last step we've met the requirements set out by TechnicalNotes.com. We've created an environment that uses XML/XSLT within a ColdFusion application framework to read local and remote XML, combine the sources, and display the results within a ColdFusion template. Changes to the XML can be handled without requiring developers to touch Java code.

Conclusion

XML and XSLT open the door to a whole new way of thinking about structured data. By integrating XML/XSLT data processing with ColdFusion, your ColdFusion applications can interact with local and remote data sources as long as that data is encoded in, or can be delivered as, XML. The approach outlined in this article will be useful in some circumstances, particularly if you need to read relatively small amounts of data on the fly. ColdFusion developers needing to integrate their applications with other Web apps may also take advantage of CF's WDDX support. For an example that uses WDDX to process XML with the CFObject tag, see the Allaire Technical Note by Anne Sandstrom, "Transforming XML to Application Data Using WDDX" (www.allaire.com/Handlers/index.cfm?ID=14244). If you have to deal with very large XML data sets you may want to use SAX rather than XSLT to process the XML. A tutorial is available at http://java.sun.com/xml/docs/tutorial/sax/. By finding the right tools for your situation, you can build an environment that will let you integrate XML into your ColdFusion applications.

About the Author

David G. Halsted, PhD, is an independent consultant. He creates Web applications that integrate XML/XSLT, databases, Java, and ColdFusion.

halstedd@hotmail.com

Listing 1

javascriptarticle.xml

```
<?xml version='1.0'?>
<page title="JavaScript in the News">
<authorln>Jones</authorln>
<authorfn>Jeremy</authorfn>

<text>
<p type="text">
JavaScript has become one of the most powerful tools a Web
application
developer can deploy.  By delivering data structures to a
JavaScript object
as a page loads, developers can provide faster response
times for users
and reduce processing load on servers.
</p>
<p type="text">
A number of recent articles describe how to handle
JavaScript in your Web application.
</p>
<p type="text">
Here's a sample:
</p>
<p type="samples" />
</text>

</page>
```

simple_article.xsl

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  >

<xsl:output method="html"/>

<xsl:template match="/">
 <html>
  <head>
   <title><xsl:value-of select="/page/@title" /></title>
  </head>
 <body>
 Author:  
<xsl:value-of select="/page/authorln" />,  <xsl:value-of
select="/page/authorfn" /><br />
 </body>
 </html>
</xsl:template>

</xsl:stylesheet>
```

article_style.xsl

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  >

<xsl:output method="html" />
<xsl:variable name="articleList"
select="document('articles.xml')/articles" />
 <xsl:template match="/">
<html>
<head>
 <title>
  <xsl:value-of select="/page/@title" />
 </title>
 <style type="text/css">
 body {font: 12pt Arial}
 .bibInfo{text-align:right}
 .author{font: 10pt}
 .title{font: 10pt;font-style:italic;text-align:right}
 </style>
</head>
<body>
 <div class="bibInfo">
<span class="author">Author: <xsl:value-of
select="//authorln" />, <xsl:value-of select="//authorfn"
/></span><br />
 <span class="title">Title: <xsl:value-of
select="/page/@title" /></span>
 </div>
```

```
 <xsl:apply-templates />
</body>
</html>
</xsl:template>

<xsl:template match="authorln|authorfn" />

<xsl:template match="text/p">
 <xsl:choose>
  <xsl:when test="@type='text'">
   <p><xsl:value-of select="text()" /></p>
  </xsl:when>
  <xsl:when test="@type='samples'">
   <p>
   <ul>
   <xsl:for-each select="$articleList/article">
    <xsl:sort select="authorln" />
    <li>
<xsl:value-of select="authorln" />, <xsl:value-of
select="authorfn" />. <span class="title" style="font-
size:12pt"><xsl:value-of select="title" /></span>
(<xsl:value-of select="length" /> pp.)
</li>
   </xsl:for-each>
   </ul>
   </p>
  </xsl:when>
 </xsl:choose>
</xsl:template>

</xsl:stylesheet>
```

articles.xml

```
<?xml version='1.0'?>
<articles>
 <article>
  <authorln>Lopez</authorln>
  <authorfn>Mary</authorfn>
  <title>Scripting Languages from A-Z: Alternatives to
JavaScript</title>
  <length>20</length>
 </article>
 <article>
  <authorln>Brown</authorln>
  <authorfn>Anne</authorfn>
  <title>Using JavaScript in JSP Pages</title>
  <length>35</length>
 </article>
 <article>
  <authorln>Smith</authorln>
  <authorfn>Jonas</authorfn>
  <title>JavaScript Dreams; A Programmer's Tale</title>
  <length>10</length>
 </article>

 <article>
  <authorln>Thompson</authorln>
  <authorfn>Samuel</authorfn>
  <title>Using JavaScript for Statisticians</title>
  <length>15</length>
 </article>

</articles>
```

Listing 2

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

/**
 * CFTestServlet is designed to test installations of Cold-
Fusion 4.5's
 * cfservlet tag.  It responds when called with or without a
passed parameter.
*/

public class CFTestServlet extends HttpServlet {

  public void service(HttpServletRequest req, HttpServlet-
Response res)
```

```
  throws ServletException, IOException
{
 res.setContentType("text/html");
 ServletOutputStream out = res.getOutputStream();

 out.println("<html><title>CFTestServlet
Results</title><body>");

 // return data, whether the page is called with or without
parameters

 out.println("<center><h3>CFTestServlet</h3></center>");
 out.println("<p><b>Greetings from CFTestServlet</b><p>");

 // read all the parameters with which the servlet was
called
 // and display any you find

 Enumeration p = req.getParameterNames();

 if (p.hasMoreElements()){
   out.println("<b>Parameters and values:</b><p>");
  }

 while (p.hasMoreElements()){
  String name = (String)p.nextElement();
  out.println(name + " = " + req.getParameter(name) +
"<br>");
  }

 out.println("</body></html>");

 }

}
```

Listing 3

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

import com.icl.saxon.*;
import com.icl.saxon.output.*;
import com.icl.saxon.expr.*;
import org.xml.sax.*;

/**
 * MiniSaxonServlet. Transforms a supplied input document
using a supplied stylesheet.
 * Based on Michael Kay's SaxonServlet.
 */

public class MiniSaxonServlet extends HttpServlet {

      public void service(HttpServletRequest req,
HttpServletResponse res)
 throws ServletException, IOException
    {

  // rootpath = directories in which you place XML and XSL
files
  // source = XML file to transform
  // style = XSLT to use for the transformation

  String rootpath = req.getParameter("rootpath");

  String source = rootpath +  req.getParameter("source");
  String style =  rootpath +  req.getParameter("style");

     try {

  // grab the output stream and set the content type

  ServletOutputStream out = res.getOutputStream();
  res.setContentType("text/html");

  // call apply(), which applies the XSLT to the XML;
  // this version of apply() returns a string

  String outStr = apply(style, source);
  out.println(outStr);

         } catch (SAXException err) {
res.getOutputStream().println("Error applying stylesheet: " +
err.getMessage());
         }
   }

/**
* Apply stylesheet to source document
*/

    private String apply(String style, String source)
              throws SAXException, java.io.IOException
    {
            try {

 // prepare a style sheet and create an instance of it

PreparedStyleSheet pss = new PreparedStyleSheet();
 pss.prepare(new ExtendedInputSource(new File(style)));
             StyleSheetInstance ssi =
pss.makeStyleSheetInstance();

 // use a string writer to collect the results of the
transformation as a string

             OutputDetails details = pss.getOutputDe-
tails();
         StringWriter spWriter = new StringWriter();
 details.setWriter(spWriter);
  ssi.setOutputDetails(details);

 // apply the XSLT to the XML and return the resulting
string

             ssi.transform(new ExtendedInputSource(new
File(source)));
 return spWriter.toString();

         } catch (SAXException err) {
return err.getMessage();
         }
    }
}
```

Listing 4

```
<cfservlet code="MiniSaxonServlet">
 <cfservletparam name="rootpath" value="C:/cfdj/">
 <cfservletparam name="source"
value="javascriptarticle.xml">
 <cfservletparam name="style" value="simple_article.xsl">
</cfservlet>
```

Listing 5

```
<table cellpadding="10" width="100%">
    <tr valign="top">
 <td style="background-color:wheat;color:maroon;width:80">
  TechnicalNotes.com  Site Directory
 </td>
 <td>
  <cfservlet code="MiniSaxonServlet">
   <cfservletparam name="rootpath" value="C:/cfdj/">
<cfservletparam name="source" value="javascriptarticle.xml">
   <cfservletparam name="style" value="article_style.xsl">
  </cfservlet>
 </td>
    </tr>
</table>
```

CODE LISTING

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

Java for CFers Part 2 of 3

Beyond JSP, it's all about tags

BY BEN FORTA

As I explained last month (and in several columns earlier this year), Java is here to stay, and Allaire is fully embracing the technology. For ColdFusion developers this is a scary proposition. The thing we love most about CF is that it's easy, simple, and rapid – and those aren't adjectives usually used to describe Java.

Or, rather, it was a scary proposition. As I showed you last month, server-side Java can actually be relatively painless. The trick is to use JSP – JavaServer Pages, a scripting-style interface that gives you access to all the benefits of Java without the pain.

As promised, this month we'll continue our look at JSP (using Allaire's JRun application server, which I strongly recommend that you download and play with).

Note: **Those of you who attended the Allaire Developer Conference last month were fortunate enough to see Java-based CF in action. You also learned that CF5 is not based on Java, and that all initial tests show that Java-based CF (the version after CF5) executes faster than the current C-based CF.**

URL Parameters and Form Fields

Continuing last month's analysis of comparable CFML and JSP code, let's look at what it would take to access URL parameters or FORM fields. The following is a simple link with two parameters passed to a second page:

```
<A
HREF="page2.cfm?fname=Ben&lname=Forta">Click</A>
```

In ColdFusion you could display the passed data as simply as this:

```
<CFOUTPUT>
Hello #fname# #lname#
</CFOUTPUT>
```

The JSP equivalent is just as simple. Within JSP pages a set of objects are always available to you. Now before you panic, don't let the term *object* scare you – you don't need to understand object-oriented development to use these objects, as you're about to see. There is a whole range of objects you can use to perform all sorts of operations. For example, the "application" object is used to read and write application-wide data (much like ColdFusion APPLICATION variables), the "session" object is used to work with session variables (much like ColdFusion SESSION variables), and the "response" object is used to send data (for example, cookies) back to the browser (much like using the <CFCOOKIE> tag, for example). One of the most important objects is the "request" object, which is used to retrieve information that is passed from the browser during a request. This is the object you'd use to return URL parameters (and form fields too).

The following is the JSP equivalent of the above CFML code:

```
Hello <%=request.getParameter("fname")%>
<%=request.getParameter("lname")%>
```

JSP objects have "methods" associated with them – which can be thought of as functions pertaining to an object. Any interaction with an object occurs via the appropriate methods; "request" has a method named *getParameter*, which is used to retrieve a specified parameter's value. request.getParameter("fname"), for example, retrieves the fname parameter.

Of course, if you prefer, you could retrieve the values and save them to variables for later use (as you could do in ColdFusion). Here's a commented example of this:

```
<%--- Get URL parameters ---%>
<% String fname=request.getParameter("fname"); %>
<% String lname=request.getParameter("lname"); %>
<%--- Display name ---%>
Hello <%=fname%> <%=lname%>
```

JSP doesn't distinguish between parameter types, so if fname and lname had been FORM fields, the code would have still worked as is. And unlike ColdFusion, JSP won't throw an error if the specified parameter doesn't exist. Instead it'll return an empty string.

As you can see, access user data is only slightly more involved than it is in ColdFusion.

Managing Sessions

ColdFusion programmers use the SESSION scope for any data that needs to persist between requests. Session state management in JSP (and JRun in particular) is very similar to ColdFusion. As already mentioned, the session object is used to manage session data. In JRun, as in ColdFusion, client sessions are automatically created when needed if session state management is enabled. And like ColdFusion, session state management in JRun is turned on per application (although in JRun this is done in the application definition in the administrative tool and not in a tag). And again, as in ColdFusion, cookies are used to send an ID back and forth.

Here's how you'd set a SESSION variable in CF:

```
<CFSET SESSION.name="Ben">
```

And here's the JSP equivalent:

```
<% session.setAttribute("name", "Ben") %>
```

Reading session variables is just as easy. Here's the CFML syntax:

```
SESSION.name
```

And here's the JSP equivalent:

```
session.getAttribute("name");
```

Simple as that. Oh, and you'll like this one – you don't have to lock variable access in JSP.

Working with Databases

So far, so good. But now it gets a little ugly. As you've seen thus far,

JSP provides language elements and objects for all sorts of basic operations. But when you need to interact with external systems (like databases), things aren't as simple.

There is no JSP tag or language element for database access. We ColdFusion developers take tags like <CFQUERY> (and <CFMAIL>, <CFPOP>, etc.) for granted sometimes. Developers using other languages often find themselves jumping through hoops to achieve the same functionality. And as database access is such an important part of application development, this warrants special attention.

So, to set the stage, here's some simple CFML code, the kind you and I write every day:

```
<!--- Retrieve data --->
<CFQUERY DATASOURCE="forta.com" NAME="articles">
 SELECT title FROM articles
</CFQUERY>
<!--- Loop through and display results --->
<CFOUTPUT QUERY="articles">
 #title#<BR>
</CFQUERY>
```

Now you're really going to appreciate CFML. As JSP has no special database support, developers have to use lots of embedded Java. The equivalent of the above code (take a deep breath and make sure you're sitting down) could look a lot like this:

```
<%@ page
import="java.sql.*,javax.sql.*,javax.naming.*,java.io.*"%>
<%
// Set DB access code.
String dsName = "Forta.com";
    Connection dbConnection = null;
    Statement dbStatement = null;
   String sqlStatement = "SELECT title FROM articles";
    ResultSet dbResultSet = null;

try
{
    // Define JNDI InitialContext object.
    InitialContext ctx = new InitialContext();
    // Look up data source in InitialContext.
    DataSource ds =
(DataSource)ctx.lookup("java:comp/env/jdbc/" + dsName);
    dbConnection = ds.getConnection();
    // Create Statement object.
    dbStatement = dbConnection.createStatement();
    // Execute the query.
   dbResultSet = dbStatement.executeQuery(sqlStatement);
    // Print the result set.
     int rows = 0;
     //Step through the result set.
    while (dbResultSet.next())
    {
        rows++;
        // Display row contents.
        thisLine = "";
        for (int i = 0; i < colCount; i++)
        {
            // Column index is based on 1.
            thisLine += dbResultSet.get
             String(i +1);
            thisLine += "<br>";
        }
    }
}
catch (Exception e)
{
    out.println("<p>Exception in main try block");
    e.printStackTrace();
}
    // Do this no matter what.
    finally
    {
        // Clean up.
        try
        {
            if (dbResultSet != null) {
                dbResultSet.close();
            }
            if (dbStatement != null) {
                dbStatement.close();
            }
            if (dbConnection != null) {
                dbConnection.close();
            }
        }
        catch (SQLException sqlex)
        {
            out.println("<p>SQL exception in
             finally block");
            sqlex.printStackTrace();
        }
    }
%>
```

Okay, so calling it a little ugly was a bit of an understatement – truth is, it's downright horrid. In fact, it's not even JSP, it's Java, plain and simple. JSP is a simplified interface to Java, but when the appropriate simplification doesn't exist, JSP developers must resort to using Java. Or must they?

Note: Like ColdFusion developers, JSP developers use database drivers to interact with databases. But instead of ODBC, they use a JDBC driver (similar to JSP, but Java based). ODBC drivers actually can be used with JSP (via a special piece of software called a *bridge*), but for performance and scalability reasons you won't want to use ODBC drivers this way on a production site.

Once Again, It's All About Tags

The most significant recent enhancement to the JSP spec is support for Tag Libraries – or, as we CFers would call them, *Custom Tags*. JSP provides a mechanism to encapsulate and black-box underlying code in much the same way as ColdFusion does – using tags and tag families, and by passing attributes to them. Your JSP code will still use Java (like the code above), but you don't have to know or care about it – you just use tags and they do all the work.

Which is great, except that standardized Tag Libraries for the most part don't exist yet – you'll have to create your own. Unless, of course, you happen to be using JRun as your JSP server (why on earth would you use any other product anyway?). JRun comes with a set of Tag Libraries, which (and this should come as no surprise) include all sorts of tags that closely mimic the behavior of CFML tags. You can use the sql tag (which is similar to <CFQUERY>) and the foreach tag (which is similar to <CFOUTPUT>) and pass the appropriate attributes to them – the tags do all the heavy lifting.

The following is a block of JSP code (using JRun Tag Libraries) that accomplishes the same thing as the previous Java code:

```
<%@ page import="allaire.taglib.*,java.sql.*" %>
<%@ taglib uri="jruntags" prefix="jrun"%>
<%--- Execute SQL ---%>
<jrun:sql datasrc="forta.com" id="articles">
 SELECT title FROM articles
</jrun:sql>
<%--- Loop through results ---%>
<jrun:foreach item="i" group="<%= articles %>">
 <%= i.title %><BR>
</jrun:foreach>
```

Okay, so it isn't CFML, but it is orders of magnitude better than straight Java. Once again (as we CFers knew years ago), the future of application development is tags.

Summary

We've covered a lot of ground by now, and I hope that Java (via JSP) is a lot less scary, and maybe even somewhat intriguing. But there's more, and in the next issue we'll conclude this series with a discussion on using Java and ColdFusion together. Stay tuned.

ABOUT THE AUTHOR

Ben Forta is Allaire Corporation's product evangelist for the ColdFusion product line. He is the author of the best-selling ColdFusion 4.0 Web Application Construction Kit *and its sequel,* Advanced ColdFusion 4.0 Development, *as well as* Allaire Spectra E-Business Construction Kit *and* Sams Teach Yourself SQL in 10 Minutes. *He recently released* WAP Development with WML and WML Script.

BEN@FORTA.COM

The Allaire Developer Conference: A Developer's Perspective

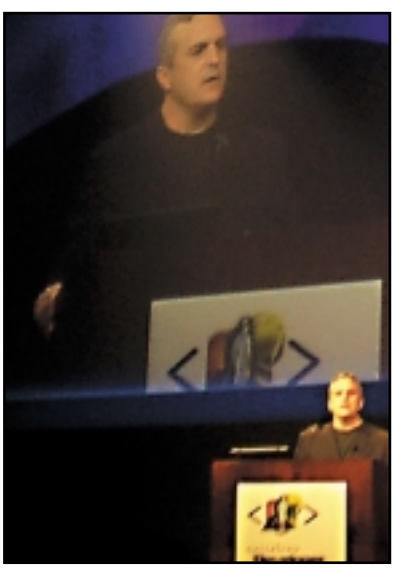
Well worth the trip

BY KEVIN SCHMIDT

Sitting in the lounge at Detroit Metro, waiting to board my flight to DC, I began to wonder what to expect at the Allaire Developer Conference. Would I really learn anything? Meet anyone interesting? Actually enjoy myself? After all, this would be my first developer conference of any kind, and all I had to go on were a schedule of events and a few bits of information from Allaire.

I began to conjure up images in my mind....Would this be nothing more than a geekfest, with a bunch of geeks sitting around reveling in their geekdom? I snapped out of it when I finally heard my flight number.

After a pleasantly uneventful flight, I arrived in Washington late Saturday evening. I'd opted to stay off-site of the convention and get a rental car, which in retrospect was a bad idea; anyone who knows the Beltway's traffic and parking issues knows what I mean. No matter though – here I was ready to take the plunge. Into what, though, I still wasn't sure.



David Orfao presents keynote.

Show floor

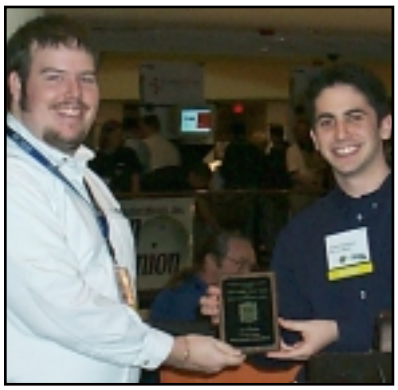
<CF_Underground>

The convention didn't get started until Sunday afternoon, so I signed up for one of the preconference events. I'd get my first taste of what was to come at <CF_Underground>. It was an informal gathering of members of the ColdFusion community organized by TeraTech (www.teratech.com); other sponsors included fusionauthority.com, Granularity, and Marketing Matrix. The agenda included discussions about ColdFusion, presentations by some well-known members of the CF community, and a rousing game of charades in which participants had to act out ColdFusion tags and functions. The atmosphere was relaxed and enjoyable; the event was held in a local tavern.

The most useful presentation – as far as I was concerned – was by Hal Helms, a fellow ***CFDJ*** author. Helms's presentation was on the shortage of ColdFusion programmers and how adopting a programming methodology such as Fusebox (www.fusebox.org) would help alleviate it. He made his case and backed it up with keen insight. When he finished, I found myself wondering why I hadn't been using a methodology like Fusebox in all my projects.

As it turned out, I was presented with a copy of a new book at the gathering. Its topic? You guessed it – Fusebox. The book *Fusebox: Methodology and Techniques*, was authored by Steve Nelson, Craig Girard, and Nat Papovich, with a little help from Helms as well, and was published by Fusion Authority. The book takes us step-by-step through the development of an application with the Fusebox methodology. Although I haven't been able to read the entire book just yet (hey, I'm on deadline),

CFDEV.COM receives *CFDJ* Readers' Choice Award.



Macromedia receives *CFDJ* Readers' Choice Award.

it's certainly one of the better tools I'll be taking back with me. (*Ed. Note:* A review of the book appears in this issue of ***CFDJ***.)

Overall, <CF_Underground> was well worth the time and was well organized for a usergroup event. After a few more presentations, I was off to the convention to register.

Welcome to the Developer Conference

By the time I pulled into the parking lot of the Wardman Park Marriott where the conference was being held, I had a little better understanding of what I was getting into. I found my way to the registration desk through a sea of people, all with looks of anticipation and excitement on their faces. I gave the attendant my name, and in return got the requisite name tag and conference freebies.

That's the main reason I came to the conference, of course – the free stuff. (Funny, my employer thinks I'm here to get certified.) My first freebie was the welcome package from Allaire. This included my conference schedule and materials, various and sundry other items,

Interview with Alexa Raad from CyberCash.

and a nice notebook for whatever I might find...well...noteworthy.

I decided to head up to the exhibitors' level for a little while. Once there, I found a large assortment of refreshments, including an open bar and enough cheese to feed the French army. After sampling this bounty, I decided the vendor area was a little too crowded, so I decided to go out for the evening. After driving around the nation's capital for a while and eyeballing some of the more well-known landmarks, I headed back to my hotel. A good night's rest, I thought, would have me more than ready to tackle the next day's events.

Ready for Anything

"We're sorry, Mr. Schmidt."

This was the response I got from the front desk when I asked them what happened to my wakeup call. Fortunately, I was up in time to make my first session but I missed Jeremy Allaire's keynote. I got the *Reader's Digest* version from my fellow convention-goers, who said it was chockful of really cool buzzwords and code names for new Allaire products. Everyone, it seemed, was reengergized about the Allaire product line.

My first of three sessions for the day was the hands-on variety. The purpose of "Unwiring the Web: Developing WAPplications with CF" was pretty obvious – how to integrate ColdFusion with WML and WAP. Allaire provided an excellent environment in which to learn, with every person having a dedicated computer with all the necessary software. The instructor gave an overall presentation on CF, WAP, and WML, and we then proceeded to tinker under the hood and try it for ourselves. My only complaint was that we didn't have enough time to cover all the material. On the upside, I left the session with a new skill in its infancy, and I'll have something to play around with when I get back home.

After uploading my brain with these nifty new technologies, it was time for lunch and I really like to eat. I was expecting something simple, something minimalist, something like sandwiches and chips. I was wrong. Once again, Allaire put forth an excellent effort. After lunch it was back to the sessions.

My second session was on EJB – Enterprise JavaBeans. Unlike the previous session, this wasn't hands-on. The presentation was made by Bushan Byragani of Allaire, and it was quite informative. I don't have a lot of experience with Java or EJB, but I was able to follow his presentation and once again learn something new. There were plenty of questions from the audience, and Byragani answered them all. He was cordial and receptive, and he finished by throwing T-shirts into the audience, stating that "As long as developers rule the world, everything will be fine." Of the day's sessions, this was the one I enjoyed most.

Off to my next session, "Designing J2EE Apps: When to Use JSP, Servlets, EJB, and JMS," which left me a bit disappointed. The material was interesting, but the presenter lacked the enthusiasm of his predecessor. As I looked around, I saw that many others had the same semi-uninterested look on their faces. After the session I decided I needed a break, so I grabbed a Coke and headed to the CyberCafé.

SYS-CON Radio interviews J. Todd Peters from PaperThin.

The CyberCafé

The CyberCafé offered attendees a chance to break away from the convention and relax. Here I could check my e-mail, surf the Web, play a little pinball or air hockey, and talk with other members of the CF community. This was new to the convention this year, and it looked as though everyone appreciated the effort made by Allaire to keep us Web denizens somewhat connected. After checking my e-mail and doing a little surfing, it was time to head up to the vendor area. Hopefully, I'd land some more free goods

SYS-CON Radio interviews Sanjeer Abrol from Impact-Inteliant.

and maybe find some useful tools in the process.

The Vendor Area

Still awhirl with people, the vendor area provided an excellent opportunity to learn what products were being developed vis-à-vis ColdFusion. The majority of the vendors were offering various software products and hardware, and the rest were offering services such as hosting and consulting. Having no shame, I spent the lion's share of my time at the booths with the best freebies. Overall, most of the vendors I spoke to were fairly knowledgeable about the technical aspects of their products, but my most unique experience by far was at Macromedia's booth.

When I walked up, the rep was in the middle of a demo showcasing their latest product. As I spoke to a few people around me, I offered (for what it was worth) my less-than-favorable opinion of GUI development tools in general. One of these people pressed me to elaborate further, then countered that she used the Macromedia product, enjoyed it, and thought it worked well. No shrinking violet myself, I continued to argue my case. I explained to her that I thought it didn't integrate well with ColdFusion's data sources, and that the code it generated was filled with stuff I'd never write.

Steve Drucker from Fig Leaf receives *CFDJ* Readers' Choice Award.

After I'd run out of breath, the woman informed me that she was the product's product manager, worked with Macromedia, and was only posing as a developer to get my true opinion – which, I dare say, she got. Undaunted, she invited me to a sneak preview of the revision currently in development. Although I can't disclose most of what I saw (lest she have me killed), I can say that what I saw seemed like a huge improvement over the current version. It added a number of new tools and features, and everyone around me seemed quite enthusiastic. I, however, will stick with Studio. After this excitement I decided to partake in the Allaire Alliance Partner event in the atrium.

Time to Call It a Night

After taking some time to eat and network with my fellow ColdFusion developers, I decided to head back to my hotel and get some rest. I still had two days of sessions ahead of me, including the Certified ColdFusion Developer exam on the final day (you remember – the reason my company sent me here in the first place).

Conclusion

As I mentioned at the outset, when I arrived for the conference I wasn't sure what to expect. After my first few days, though, I'm pleased to report that I enjoyed myself immensely. And not just because of the fine food and freebies. For anyone who's wondering whether to attend the conference, I recommend it highly. Not only have I had the opportunity to interact with some of the major players in the ColdFusion community, I've learned about some new tools and had a good time in the process.

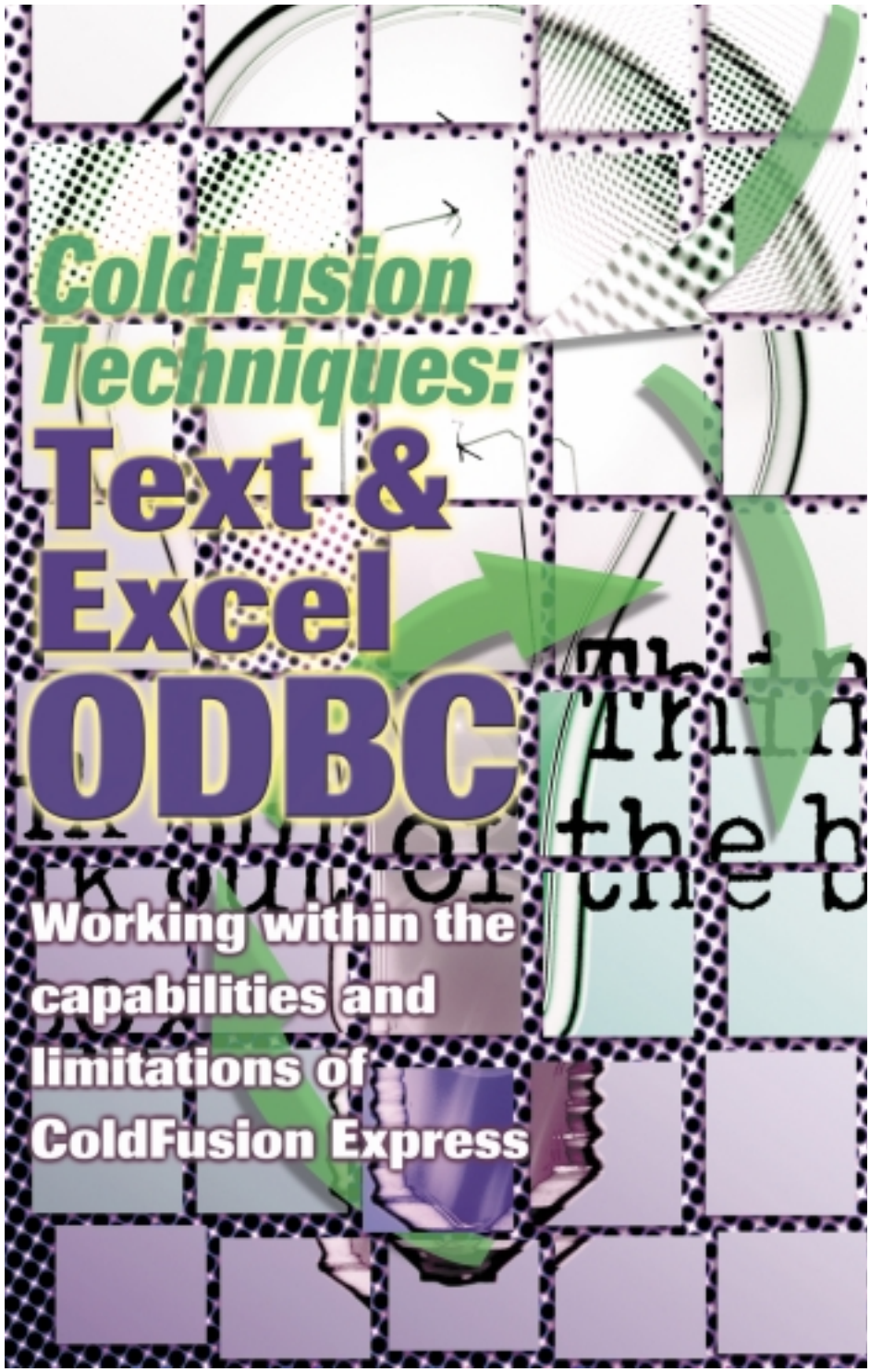
Next year's convention, by the way, is at Disney World.

ABOUT THE AUTHOR

Kevin Schmidt is an Allaire Certified ColdFusion Developer (I passed the exam!) and is Internet Services Manager for PWb, an integrated marketing and information design firm in Ann Arbor, Michigan. He spends most of his time developing Web applications for clients using ColdFusion and Oracle.

SCHMIDT@PWB.COM

Recently I needed to create an ODBC source that would allow an application to read and write from a text file. An obvious question is: Why would I use ODBC to read from and write to a text file? After all, ColdFusion supports this easily using the <CFFILE> tag. You'd be right if you were talking about a purchased copy of ColdFusion. My requirements involved ColdFusion Express, which doesn't support the <CFFILE> tag.



BY
RAYMOND H. THOMPSON

Unfortunately, there's little, dare I say no, documentation on how to use ODBC against text files. There was no information in the CF manuals, nothing on Allaire's Web site, and nothing in the ODBC documentation from Microsoft.

On the surface it seemed simple. But it wasn't. There were some surprises along the way. Eventually the mystery was solved, and I was able to read from and write to a text file. In the process I also decided to explore reading from and writing to an Excel spreadsheet directly using the Microsoft ODBC driver.

The Situation Defined

As I explained, the application required the use of ColdFusion Express. This is the free version from Allaire that can be downloaded from their Web site. It has many of the capabilities of the full version. However, there are limits, and one was the lack of support for directly reading from and writing to text files. And, as the <CFFILE> tag wasn't available, I needed a method to access information in a text file and output information into another text file.

This prompted the search for documentation. It's probably somewhere on the Web, but I couldn't find it easily. Even a question posted in the developer conference failed to produce a timely answer to my needs. So I set off on my own.

Defining the Source

The first thing to do was define an ODBC source for the text files. I created a directory and placed a text file in it. The contents of the file are shown in Figure 1. Then I brought up the CF Administrator and created an ODBC source selecting the text file. The connection wouldn't verify. Okay, so step back, think, and perhaps look more closely at the screen in the CF Administrator.

The CF Administrator screen clearly says "database directory." And there's a box on the CF Administrator screen that says "Extensions List" and has the values "asc,csv,tab,txt" listed in the box. I changed the data source to point to the directory, then put "txt" in the Extensions List box. The connection now verified correctly. This indicates that the ODBC connection must point to a DIRECTORY that contains the text file or files. The "Extensions List" needs to contain the extension of the file or files desired. Adding additional extensions separated by commas can process multiple types of files.

Accessing the Data

Now it was time to try to access the data in the text file that was placed in the directory. I was clueless about how the SQL would be constructed to access the data, so some trial and error was needed.

I guessed correctly that the table name would be the name of the text file by virtue of the ODBC setup in the CF Administrator pointing to a directory instead of a file. I'd placed two records in the table with the data comma-delimited as that seems to be a common method of separating text data. I wrote a small script to access the file, which did nothing but select data from the file using only the file name as the table name. It returned an ODBC error and failed to work. I then added the extension of the file to the table name, making the SQL Select * from address.txt. That was successful, so it was obvious that the fully qualified file name is required even though "txt" was specified in the ODBC setup.

The next thing was to see the names of the columns that were returned from the query. I knew that the variable queryname.ColumnList, where queryname is the name of the query, contains a list of column names returned from the query. Since I'd selected all columns, this variable would show me all the columns that were returned.

```
Name,Address,City,State,Zip,Phone
Ray,105 Mitchell,Oak Ridge,TN,78233,865-481-6832
```

FIGURE 1: A text file

I was pleasantly surprised to find that the column names consisted of the comma-delimited values in the first record. A little experimentation showed that column names could have spaces in them as long as they had brackets around them in the query. For example, a column name of First Name would be [First Name] in the SQL for the query.

I then extended the script to output the data that was in the text file. It appeared as expected, with data from the text file starting at the second row as data. A comma was needed between each data field in the text file. I also discovered that if your data contains commas, then simply surround each field with double quotation marks. If your data contains double quote marks, then escape the double quote marks by placing two double quote marks in a row.

The problem of reading in data from a text file was solved. The first row is the column name, commas separate the data, and the table name is the fully qualified name of the file with the extension. It was even possible to select only specific rows by using a WHERE clause in the query.

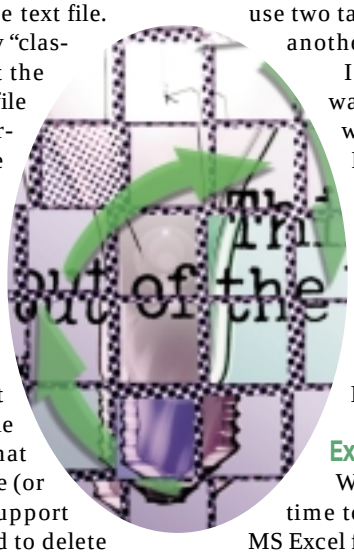
Outputting the Data

Now it was time to create data in a text file using the ODBC driver. Since I already knew that the column names would be placed in the first row of the table, I felt it should be easy to create a file.

The first thing was to create a file with the proper headings, which I could have done by creating the file with just the column names using any standard text editor. But the idea was to make the

entire process automatic. I looked for clues in "Help" for MS Access and found some information on creating a table. The code for creating a table is shown in Listing 1. For this example I created a file with text, numeric, and date fields. I don't know if this really matters for a text file, but it certainly helps to document what you're trying to create.

Next, I wrote a small script to place data in the text file. The query was the same as would be used for any "classic" database. After running the script I looked at the file with a text editor, and the data was in the file almost as expected. The text data was always surrounded by double quotes even though they were not specified when the data was written. For the numeric fields I was never able to figure out how to specify the number of decimal places. You can use NUMBER for a field type and get a field with two decimal places. The result of the code is shown in Listing 2.



At this point I had all the information I needed to create a text file using an ODBC driver. The last piece of information was how to eliminate the file if it existed. Any attempt to run the SQL code that created the file would cause an exception if the file (or table) already existed. Since CF Express doesn't support the CFFILE tag, another method had to be devised to delete a file. I'd already used ODBC to create a table, so why not use ODBC to delete one? It was certainly worth a try.

I used the CF tag <CFIF FileExists()> to look for the presence of the file (or table) and, if it existed, then use some SQL to delete it. The result was the code in Listing 3. If the file (or table) existed, it would be deleted using the DROP SQL command; it could then be re-created. *Note:* When using the FileExists() function, it's necessary to fully qualify the file (or table) name with the drive and directory.

Notable Limitations

One thing you can't do with ODBC is update the data in the file. You can't use the UPDATE SQL or DELETE SQL commands; all you can do is INSERT data. And any inserts are placed at the end of the existing data. So you can add to a text file, but that's about it. If you really needed to update the data, you'd have to use two tables – reading from one table, then inserting into another. This would be tedious.

I also observed that the process of writing the file was extremely slow: it took almost 45 seconds to write 100 records to the file. This was done on a Pentium 450 with ColdFusion and the ODBC source on the same system. (This was a top-notch system two years ago; now it's low end.) This isn't something you'd want to use for large volumes of data. Reading the data was reasonably quick, but writing was painfully slow.

This did accomplish the goal of allowing the reading and writing of text files with ColdFusion Express.

Excel Data Source

With the problems of text files out of the way it was time to turn my attention to discovering how to use an MS Excel file as a data source. Fortunately, I had some documentation on how to use MS Excel as an ODBC data source, so this should have been easy. Unfortunately, the documentation was incorrect – the examples it gave wouldn't work.

Now, this may be due to the version of Excel I'm using. The Administrator clearly only allows files created with Excel version 3.0, 4.0, or 5.0. The only version I had was Excel 2000, and I was using the "Save As" option to save the file as Excel 5.0. Regardless, I couldn't get CF to work according to the example.

Setting up the data source was no problem, and unlike a text source that points to a directory, the data source for Excel points to a file. This implies that the data to be used as tables is contained within the Excel file.

The example I had says that you must have a data sheet with a name that contains a dollar sign ($) as the last character. You then use this name as the table name in the query and place it in quotes. Well, it just flat out wouldn't work. I tried removing the trailing dollar sign from the sheet name. I tried using single quotes and double quotes. I removed the trailing dollar in the query. Every combination I tried failed for syntax error in the query or not being able to find the table.

I then highlighted the data in the sheet and gave it a name by using the menu items "Insert, Name, Define" and giving the area a name called "Staff." That didn't work either. I then changed the name of the sheet to "Staff." That worked. Apparently you have to have the sheet named what you want your table named, and you have to have the area defined in your spreadsheet that defines the area. I was able to run a simple query selecting all the data. I used the same data as was used for the text ODBC (see Figure 1) and the same query (changing the table name) and output (see Listing 4). The column names were properly retrieved. An interesting side effect is that the column names were returned in alphabetical order.

Formatting on the cells doesn't seem to matter, as the ODBC driver makes its own decisions about the data. In the case of the zip code, Excel was informed that it contained text data. Yet when the data was returned from the query it clearly considered the zip code to be a number.

I then expanded the data by one column but didn't change the area defined by the name "Staff." The query still worked, but the additional data didn't appear. Only when I expanded the defined area to include the new column did the new information appear.

From all of this it was obvious that the sheet name is the table name, and you must have an area defined on your sheet that encompasses all the required data. As in the text ODBC, the first row is considered to contain the names of the columns for the data.

When defining the data source for Excel in the CF Administrator there's a field called "Rows to Scan." I have no idea what this field represents. I originally thought it was the maximum rows to be returned, but that's not what it does.

I should also point out that if the defined area contains blank rows, then they are also returned. The named area on the spreadsheet is what's always returned when doing the query.

> "The need to read from and write to text files using ColdFusion Express made me explore how to do so within its limits

Writing to Excel

Now it was time to see how to write data to an Excel spreadsheet. I used the code shown in Listing 1 that was used in the text example. I changed the table name and removed the .txt extension. The code ran with no errors. Time to open the spreadsheet and see what happened.

The data appeared in the Excel spreadsheet as a sheet within the file. There was a new page in the file called "MyTable," which is the table name I created. There was also a named area in the spreadsheet called "MyTable" that exactly matched the amount of data that would have been created. This made sense, as that was the case when reading. The sheet name and the named area on the sheet need to be the same. The data that appeared is shown in Listing 2.

I ran the code again and an error was generated when creating the table. Since the table existed, this was reasonable. I removed the create table code to see what would happen and ran the script.

Another error was produced, informing me that I couldn't expand the named range. It seems that once the data has been inserted into the spreadsheet it can no longer be changed. You can't add to or update the data, and you can't delete the data unless you delete the table using the code shown in Listing 5.

It took 1.8 seconds to insert 500 rows into the Excel spreadsheet. This is considerably faster than writing to a text file.

What Good Is It?

By being able to create a text file I was able to create the data needed in ColdFusion Express, even though it doesn't support the <CFFILE> tag. It was slow and should be used in extreme cases, which this was. Using a text file as an ODBC source would allow the reading of data with CF Express. For the "real" version of ColdFusion, the <CFFILE> tag is the obvious choice.

The capability to write to an Excel spreadsheet could be useful. You could create data for a client in Excel format, then deliver the file using the <CFCONTENT> tag. By wrapping the table delete with a simple <CFTRY> and <CFCATCH> tag, the attempt to delete a nonexistent table could be easily handled.

Even then I'd think it would be better to just create a text file with comma-delimited values, name the file with .CSV extension, and deliver it to the client if you wanted to create a single file only.

But consider if you need to deliver multiple files that are going to be imported into Excel. I think it would be better to create the multiple files using an Excel ODBC connection and create multiple tables (sheets) in a single Excel file, then transfer that file using <CFCONTENT>. The client will receive only a single file but with multiple sheets. This could be useful in some circumstances.

Wrapping It Up

The need to read from and write to text files using ColdFusion Express made me explore how to do so within its limits. The ability to use an ODBC source pointing to the text file seemed to be the answer. The process does work; however, performance is fairly poor. But it can fill a need.

After finding out about the text ODBC capabilities, I decided to explore using Excel as a data source. It was possible and did work. The documentation I had for Excel was incorrect and some experimentation was required to find the solution. The information for text files as a source was not to be found. Now that I've written about this, I suppose someone will come forward and point me to it.

The process is so slow when writing the data, making it painful except for the smallest files and when no other solution is available. Even so, it was an interesting exploration into the capabilities and limitations of using text files and Excel as ODBC sources for ColdFusion.

About the Author

Ray Thompson is employed by Q Systems in Oak Ridge, Tennessee, a leading developer of Web-based applications. 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

```
<cfquery name="Q1" datasource="CFDJ" dbtype="ODBC">
  create table mytable.txt
    (Field1 char(10),
     Field2 date,
     Field3 integer)
</cfquery>
<cfloop index="Indx" from="1" to="5">
  <cfquery name="Q2" datasource="CFDJ" dbtype="ODBC">
    insert into mytable.txt
      (Field1, Field2, Field3)
    values
      ('Ray',#CreateODBCDate(Now())#,#Indx#)
  </cfquery>
</cfloop>
```

Listing 2

```
"Field1","Field2","Field3"
"Ray",9/20/2000 0:00:00,1
"Ray",9/20/2000 0:00:00,2
"Ray",9/20/2000 0:00:00,3
"Ray",9/20/2000 0:00:00,4
"Ray",9/20/2000 0:00:00,5
```

Listing 3

```
<cfif FileExists("c:\inetpub\wwwroot\cfdj\mytable.txt")>
  <cfquery name="Q0" datasource="CFDJ" dbtype="ODBC">
    drop table mytable.txt
  </cfquery>
</cfif>
```

Listing 4

```
<cfquery name="Q1" datasource="CFDJ" dbtype="ODBC">
  select *
  from address.txt
</cfquery>
<cfoutput>#Q1.ColumnList#</cfoutput><br>
<cfoutput query="Q1">
  #Q1.Name#,
  #Q1.Address#,
  #Q1.City#,
  #Q1.State#,
  #Q1.Zip#,
  #Q1.Phone#<br>
</cfoutput>
```

Listing 5

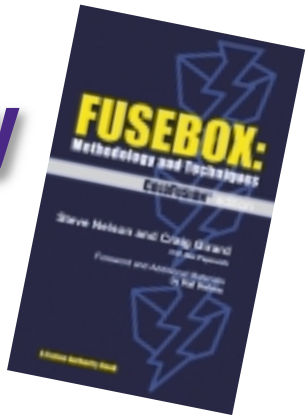
```
<cfquery name="Q0" datasource="CFDJ" dbtype="ODBC">
  drop table mytable
</cfquery>
```

CODE LISTING

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

CF Programming Methodology

REVIEWED BY NANCY LEIBY



***FUSEBOX:** Methodology and Techniques*
ColdFusion Edition
By Steve Nelson and Craig Girard with Nat Papovich
Foreword and additional materials by Hal Helms

Design, build, and test version 1 of any Web application in just five days! Wow! That's what coauthor Steve Nelson theorizes can be achieved. The case study that describes the project in which he did this, and the step-by-step instructions to achieve these results, are included in one of the final chapters of Nelson and Girard's book, *Fusebox: Methodology and Techniques, ColdFusion Edition*. The rest of the book describes the Fusebox methodology and related techniques to help you reach this goal.

For those of you unfamiliar with the name, Fusebox is a methodology for developing Web applications. Currently it's been defined for ColdFusion application development, but it's applicable to other Web application development languages as well. Fusebox methodology describes a body of practices, procedures, and rules that a ColdFusion developer uses to organize a Web application. Web application developers who use Fusebox to write CF applications create apps that are organized, structured, modular, and understandable by other Web application developers.

Fusebox includes a well-defined way to split a Web application into several defined file types, with file-naming conventions and specified types of contents. It also includes guidelines about the directory structure and which files go in each directory and subdirectory. All this puts a method to the madness of developing a Web application, making it easier for another Fusebox developer to jump in and work on a site. Conversely, a developer can create a site that's easily understood and maintained by someone else who knows Fusebox.

I come from a traditional programming background and used a structure approach to design and develop traditional and client/server applications. When I made the conversion to Web applications, I applied this traditional structured approach to design and development to Web application development with ColdFusion. Unfortunately, this was my structure, not a standard one, and I had to teach it to everyone else on the project to keep the site consistent.

Fusebox provides this structure through a well-defined methodology. It provides what I'd been trying to achieve, but it's a structure that's been refined by a community of users who've also created structure for things I hadn't even thought of. This book contains the meat and gold of Fusebox. In fact, it contains a lot of generic ColdFusion information that's useful even if you don't follow the Fusebox methodology.

Now let me explain that I'm not an expert in Fusebox. For one reason or another, I haven't had the opportunity to use Fusebox to develop a Web site. On the other hand, I've followed the evolution of Fusebox through conversations with other ColdFusion developers and through the Fusebox mailing list. When I first heard of Fusebox several years ago, I read a white paper on it. That was the only resource I knew of. This book fills in the gaps and makes your Web application development process less painful and more productive.

The Fusebox community is based around their Web site (www.fusebox.org) and mailing list hosted by House of Fusion (www.houseoffusion.com). The Fusebox Web site contains two white papers along with ColdFusion Fusebox sample code and files to download. This book does more than just repeat the information on the Web site. It elaborates on it and incorporates many of the threads that have been discussed on the mailing list. It also includes many code samples of the different file types along with code walkthroughs that explain the nuances of the code.

Along with the coding aspects of Fusebox, the book describes a methodology for the Software Development Life Cycle (SDLC) as it applies to developing Web applications with Fusebox. In fact, the SDLC described in the book applies to Web application development in general and would be useful for any CF developer or Web application project manager. Useful chapters on testing, deploying, and securing Fusebox applications are also included. Although it's helpful, you don't have to use Fusebox to reap the benefits of most of the information provided here.

Basic Fusebox concepts are presented, then built on with more advanced concepts and related material, making it a good resource for those who want to learn Fusebox, as well as experienced users.

I'm basing this review on a prepress version of the Fusebox e-book; some of the sections hadn't been completed. Additional case studies and chapters on FuseDoc and fUseML will be included in the final release. FuseDoc defines a standard set of documentation for each ColdFusion Fusebox file. fUseML is the Fusebox usage of the Unified Modeling Language (UML).

The book is available directly through the Web as an e-book at the Secret Agents and Fusion Authority Web sites (www.secret-agents.com and www.fusionauthority. com). It is also available as a print book from the same sources.

ABOUT THE AUTHOR
Nancy Leiby, a Web application consultant at Syntegra, has been working with Allaire ColdFusion for two years. Prior to that she designed and implemented Windows and client/server database applications. Nancy is currently developing Web applications using Allaire's Spectra and ColdFusion.

NANCY.J.LEIBY@SYNTEGRA.COM

Ask the Training Staff

A source for your CF-related questions

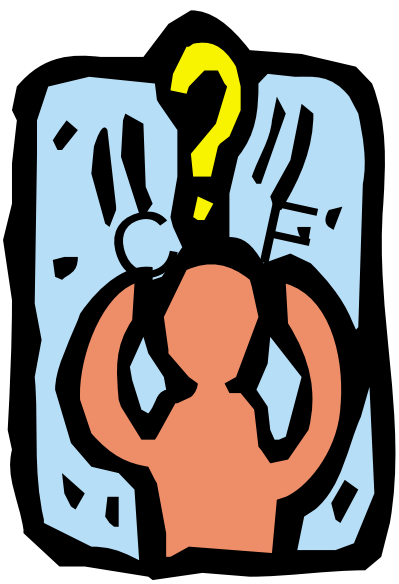
BY
BRUCE
VAN HORN

Thanks to those of you who sent in questions recently. I hope the answers you received from me or other members of the Allaire training team were helpful. For our readers, here are a few of the questions that were submitted:

Q: *I understand the importance of using CFLOCK with the Scope attribute around code that references session or application variables. Could you give me an example of why I'd ever want to use CFLOCK using the Name attribute?*

A: Let's start with the bottom line: CFLOCK makes sure that a block of code isn't executed by more than one request at a time. As you know, the SCOPE attribute is used when locking code that reads or writes variables that use shared memory spaces (variables in the server, application, and session scopes). The NAME attribute is used for locking any other kind of code that could cause problems if more than one instance of that same code were executed simultaneously.

There are a lot of places you might want to do this, but here's a simple example: suppose you have a page that inserts records into an MS Access database. Immediately after you insert a record you need to know its primary key value. In Access you need to run a subsequent query to get that value. However, if more than one person can insert records at the same time, you could get the wrong value back for your record. Using CFLOCK with the NAME attribute protects this process.



```
<CFLOCK NAME="LockInsert" TIMEOUT="20" TYPE="EXCLUSIVE">
 <CFINSERT DATASOURCE="MyDSN" TABLENAME="Customers">
 <CFQUERY NAME="GetID" DATASOURCE="MyDSN">
   SELECT Max(CustID) AS NewID
   FROM Customers
 </CFQUERY>
</CFLOCK>
```

The trick to using the NAME attribute is to make sure you give each instance of CFLOCK a unique name, or CF will single thread any code that has the same name (regardless of the code being executed!).

Most of the time we're thankful for the multithreaded capabilities of ColdFusion, but at times we need to make certain tasks single threaded. This is where you should use CFLOCK.

Q: *How can I retrieve and display to my users a .pdf (Adobe Acrobat) file from another server using CFHTTP? I get an error that says I can't output the value of CFHTTP.FileContent.*

A: The reason for the error when outputting the value of CFHTTP.FileContent after retrieving a .pdf file is that the contents are binary rather than plain text (as an HTML page would be); you can't output binary data using the CFOUTPUT tag. Use CFHTTP to retrieve the file, but physically store it on your server before showing it to the user. Your CFHTTP tag should use the PATH attribute to store the retrieved file to your server, then simply create a link or CFLOCATION to that file. The example tag below will write a file called "somefile.pdf" to the specified subdirectory.

```
<CFHTTP
URL="http://someserver.com/somefile.pdf"
        METHOD="GET"

PATH="c:\inetpub\wwwroot\pdf_files"
        RESOLVEURL="false"></CFHTTP>
```

Q: *Is there a way to programmatically flush a CFQUERY result set that's been cached before it naturally expires using the CACHEDWITHIN attribute?*

A: Yes. All you need to do is run the identical query with new values in the CACHEDWITHIN attribute. For example, if you created the cached query using CACHEDWITH="#CreateTimeSpan(0,0,30,0)#", you'd simply run the same query using CACHEDWITH="#CreateTimeSpan(0,0,0,0)#" to clear that query from the cache.

Q: *I've seen references in this journal and in some forum posts to the "Request" variable scope, but the ColdFusion help files don't say much about it. Could you please tell me what it is and why I might want to use it?*

A: The Request scope is a very useful but often overlooked variable scope. Use the Request scope for variables that you want to make available to all templates that make up a single user request. For example, a call to MyPage.cfm may result in several other pages being called (Application.cfm, cfincludes, and custom tags). If you created a local variable in Application.cfm, for example, that variable would be available to MyPage.cfm but it wouldn't be available to custom tags called by MyPage.cfm. However, if you set that variable into the Request scope (<CFSET Request.MyVar = "Some Value">), it would be available to any custom tags called as a result of this "request." The Request scope, however, isn't a persistent (memory resident) variable like the Application or Session scopes. It exists only for the life of the request from that particular user. Because it's something of an "extended life" local variable, you don't have to worry, as you would for a Session variable, about using CFLOCK to protect the memory space it occupies.

> Most of the time we're thankful for the multithreaded capabilities of ColdFusion, but at times we need to make certain tasks single threaded

• • •

Please send your questions about ColdFusion (CFML, CF Server, or CF Studio) to AskCFDJ@sys-con.com.

ABOUT THE AUTHOR

Bruce Van Horn, a regular contributor to **CFDJ,** *is president of Netsite Dynamics, LLC (an Allaire Alliance Partner), and an Allaire certified instructor.*

BRUCE@NETSITEDYNAMICS.COM

Interview... with Steve Drucker and Dave Gallerizzo of Fig Leaf Software

INTERVIEWED BY **ROBERT DIAMOND, EDITOR-IN-CHIEF** OF ***COLDFUSION DEVELOPER'S JOURNAL***

CFDJ: Fig Leaf Software is one of the top e-business consulting and training organizations in the world, specializing in the application of technology from Allaire and Macromedia. Steve Drucker, president and CEO, and Dave Gallerizzo, vice president of consulting services, will tell us what they're currently working on.

Steve Drucker: We're contributing to a new initiative for Macromedia and Allaire called *Harpoon*. Harpoon is a new toolkit – Macromedia Flash objects with Invocation methods for ColdFusion, JRun, all the Macromedia tools. You're going to be able to use Dreamweaver and UltraDev Dreamweaver, as well as HomeSite and ColdFusion Studio. There's going to be a lot of really cool advantages to it in that these objects are going to be cross-browser or cross-platform. They're also going to be dynamically resizable and can be used with JRun and ColdFusion to pass information in and out.

CFDJ: When is that going to be available?

SD: Right now we're not at liberty to say what the actual release date on the toolkit is.

CFDJ: Well, we'll keep our eyes open for it. What other stuff are you working on these days?

Dave Gallerizzo: We're actually doing quite a bit with the full-fledged suite of both Allaire and Macromedia technologies. A lot of what we're doing is Spectra-based. We're building an intranet for T. Rowe Price, a large financial company. We're driving all their internal content with Spectra and ColdFusion. A lot of the interface pieces are built using Macromedia, Flash, and Java. We're also using Spectra pretty heavily in the marketplace to drive dynamic content. We're building a pretty advanced system to do dynamic Flash content through a generator using Spectra. This is a pretty big deal and the Macromedia guys have actually approached us to do a Spectra Generator white paper. We're training all over the world right now. As a matter of fact, we just got back from South Africa where we ran a one-week Spectra class. We're preparing to help Allaire update their Spectra materials, anticipating more than five coming out very soon. So we're moving down the road with the technology as quickly as we can.

CFDJ: Can you tell us a little about Fig Leaf as a company, general things you guys are involved with?

SD: Fig Leaf is really centered around relationships with two of our vendors: Allaire and Macromedia. We're currently a 65-person company headquartered in Washington, DC, with an office in Atlanta, Georgia. We'll probably expand into a third market in 2001.

Basically, the company is broken into three major pieces: a consulting division that focuses on Web strategy application development using Allaire and Macromedia tools; a training division focused on products that our developers actually use; and a multimedia group that's really doing some wonderful work. Most of our developers double as instructors, so we're training the entire Allaire and Macromedia curriculums, as well as helping to write them. We actually wrote the official curriculum for Macromedia Flash 5 and for Macromedia Generator, as well as contributed to Spectra's five-day course. As for our multimedia group, we're about to relaunch the Fig Leaf Web site with some really hot Flash objects.

CFDJ: You've recently won three Readers' Choice Awards from the ColdFusion industry. How does it feel to be embraced by the industry?

SYS-CON Radio interviews Dave Gallerizzo (center) and Steve Drucker (r.) from Fig Leaf.

SD: It's great. One of the founding principles of Fig Leaf was to embrace a community and give out as much information as we took in. We started the very first ColdFusion users group, and we now run one meeting each month in DC, Maryland, Virginia, and Atlanta. We take all the neat things we've done that other companies might consider to be trade secrets, and we give them out to developers as freeware. We say, "Here guys, learn from the stuff we've done; let's learn from each other's mistakes, learn new techniques, and together we'll help Allaire and Macromedia build their respective markets." It benefits everybody.

CFDJ: That's definitely a different strategy from those that others are taking. Is there anything else you want our listeners to know?

DG: The really cool thing about the awards we won – Best Consultancy, Best Design, and Best Training – is that's what Fig Leaf does. All the things we really center on, we got awarded for. So we really push it down that route and we do it very well, and it's because of the technology and the support we get from companies like Allaire and Macromedia that we're able to be as successful as we are. We couldn't do it without them and we definitely want everybody out there to know it.

CFDJ: What plans do you have for making sure you win those three awards again next year?

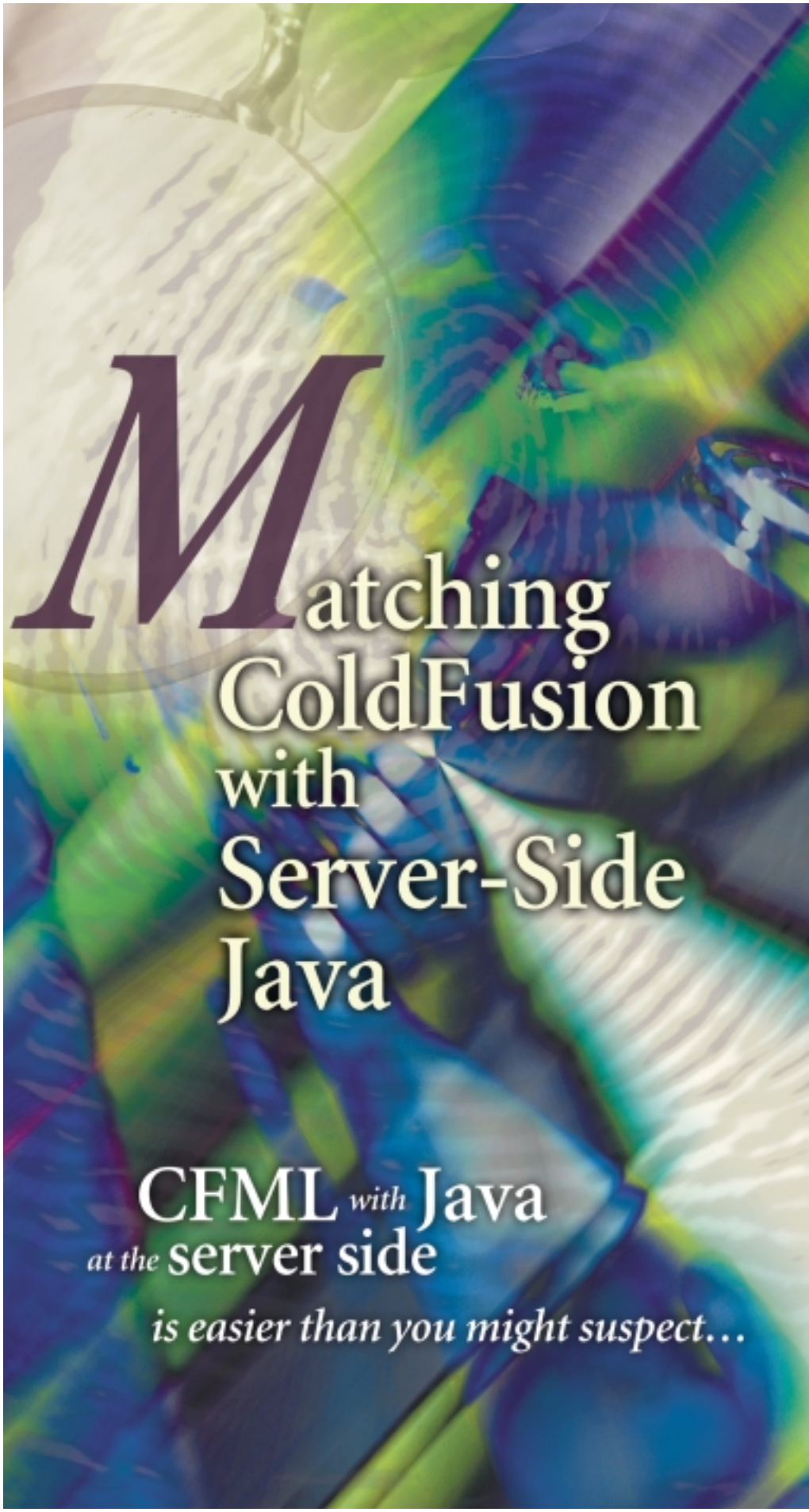
SD: I think we've got a pretty good track record. Basically, the company is focused on empowering our developers and growing the staff. So we're continuing to actively seek opportunities for our instructors to contribute to newsgroups, to talk at various trade conferences, and to author books. Next year we'll have at least two books: a Flash 5 book and a new Wrox book. Fig Leaf acts as an engine to help its developers grow and succeed. We run monthly internal users group meetings where we do some very advanced stuff and focus on new technologies. We think that focusing on employee development is going to give us a return on our investment.

CFDJ: How do people get more information?

SD: They can go to www.figleaf.com or come to one of our users groups on the first Wednesday of the month in Washington, the second Saturday of the month in Atlanta, the third Wednesday of the month in northern Virginia, or the fourth Wednesday of the month in Baltimore.

robert@sys-con.com

BY
CHRISTIAN SCHNEIDER



This article focuses on some interesting features of ColdFusion 4.5 and server-side Java. Introduced in ColdFusion 4 (and greatly extended in 4.5), the CF Server supports Java by providing three ways to access server-side Java with ColdFusion. (We're not talking about client-side Java, which is built into ColdFusion using applets, e.g., the CF tree applet.)

1. Accessing the usual Java classes/beans
2. Accessing EJBs
3. Wrapping Java code into a CFX custom tag

The first two types are supported in ColdFusion through the <CFOBJECT> tag as well as the CreateObject() function, while the third type enables developers to wrap custom Java code into a custom tag interface.

Since this article presents examples of server-side Java code and how to use it within ColdFusion, you should have a basic understanding of Java and its concepts. The Java Development Kit (JDK) includes the Java compiler as well as the runtime environment (the virtual machine) and can be downloaded from JavaSoft at www.javasoft.com. It's necessary to install the JDK to run the examples in this article. Tutorials and reference documentation about the Java language as well as how to install the JDK can also be located on the JavaSoft Web site.

What's the Point?

The main idea is to connect ColdFusion to components written in other languages so it will also be available to other environments. These are mainly distributed objects such as D/COM, CORBA, and EJBs, whereas the Java side focuses on EJBs and simple Java classes (like normal JavaBeans), as well as CFXs developed in Java. ColdFusion developers can easily reuse existing components/objects that are written in other languages or even write their own components in these languages to achieve performance gains that are dependent on the task the component should solve. CFML isn't suitable for certain components such as image processing, since this involves complex algorithms that will run magnitudes slower in CFML than in Java.

XML DevCon EUROPE 2001

Mark your calendar!
***XML DevCon* is coming to London**

Plan to be part of this next big symposium

Our New York and Silicon Valley events **sold out** in record time.

MAXIMIZING XML FOR THE ENTERPRISE

CONFERENCE: February 21–23, 2001
EXHIBITION: February 22–23, 2001
Novotel – London West
London, England

www.xmldevconeurope2001.com

Presented by: SYS-CON MEDIA XML JOURNAL Cosponsored by: OASIS Produced by: CAMELOT COMMUNICATIONS

Although the types of distributed objects (as mentioned) include D/COM and CORBA, this article focuses on Java components and doesn't take the others into account. Even though using D/COM and CORBA objects within ColdFusion is to a certain extent similar to using Java objects, their architecture is different, so a full description of D/COM and CORBA accesses is beyond the scope of this article.

Before We Start...

To connect server-side Java to ColdFusion we need to meet some requirements. These include the proper installation of the JDK (see above), as well as properly configuring ColdFusion Server (4.5) to enable it to talk to Java. The first part (installing the JDK) varies for different operating systems and is best described in the installation tutorials available at www.javasoft.com.

The other part is easier since it has to be done in the CF Administrator (see Figure 1). In the Administrator you can enable Java by providing ColdFusion with the path to the Java Virtual Machine, which points to the file jvm.dll (if you're using NT; otherwise the file may differ). This file is located in the directory c:/program files/ javasoft/ jre/ 1.2/bin/ classic/jvm.dll if you've installed only the Java Runtime Environment (JRE), or in c:/jdk1.2.2/jre/bin/classic/jvm.dll if you've installed the full JDK. Also, the ClassPath for this VM has to be set inside the CF Admin to ensure that the ColdFusion Server can find the classes you wish to access. For general use of server-side Java this should be at least c:\programfiles\jdk1.2.2\jre\lib\rt.jar, c:\cfusion\java\classes\cfx.jar and c:\cfusion\java\classes. When accessing classes that aren't located in the specified ClassPaths (which results in a ClassNotFound Exception), you have to add the appropriate path to the Java ClassPath. To be sure these settings take effect, restart the ColdFusion service after changing it.

From the other settings one can also decide to start up the JVM within ColdFusion automatically at each start of the CF Server. This increases performance slightly for the first request of server-side Java (and might prevent errors if accessing Java for the first time, and ColdFusion from taking too long to start the VM on the fly). Be aware that this option could also prevent your ColdFusion Server from starting if the Java settings are wrong. Therefore I suggest you leave this option unchecked when testing server-side Java with ColdFusion for the first time, and then (assuming everything works fine) you're free to decide whether to check this option to get a small performance increase for the first Java access after a restart.

FIGURE 1: Settings of Java in the CF Administrator

Let's Take a Look Inside

After all the conceptional and configurational stuff I guess you're curious to see some details and code. Let's look at a few examples. First I'd like to introduce the necessary tag to access and use external objects (including Java objects), such as <CFOBJECT> and its equivalent function, CreateObject(). With this tag/function you can create objects from the usual Java classes as well as EJBs, access their properties, and invoke their methods. Objects are returned inside CFML as simple variables, meaning you can assign them and their methods' results using <cfset> (this includes even complex data types being returned, such as arrays).

The creation of a Java object within CFML can be as simple as one line:

```
<CFOBJECT ACTION="CREATE" TYPE="Java"
CLASS="java.util.GregorianCalendar"
NAME="myCalendar">
```

This statement simply creates an object of the type "java.util.GregorianCalendar", which is part of the JRE that's shipped with your Java installation, and saves it into the ColdFusion variable "myCalendar" for future use. Could it be simpler? When using <CFSCRIPT> you can make it as short as this:

```
<cfscript>
     myCalendar = createObject("JAVA",
"java.util.GregorianCalendar") ;
</cfscript>
```

Now we have a variable called "myCalendar" holding a java.util.GregorianCalendar object. We can do everything to this object that we can do with java.util.GregorianCalendar objects including invoking its method "isLeapYear(year)", which checks if a given year is a leap year and returns true if it is, and false if it's not:

```
<cfset theResult =
myCalendar.isLeapYear(2000)>.
```

Here the ColdFusion variable "theResult" will be true since 2000 is indeed a leap year. *Note:* This could also be checked within CFML using ColdFusion's own function, also named "isLeapYear(year)", but doing so through the Java class is for demonstration purposes only.

The class java.util.GregorianCalendar has a default constructor (the method being invoked when the object of a class is created) that assigns the current date to the object. This enables us to display the current date of this Java object to demonstrate its use. It can be done the following way (note that months are being counted in Java starting at zero). See Listing 1 for the full source of this example.

```
<CFSET theYear = myCalendar.get(myCalen-
dar.YEAR)>
<CFSET theMonth = myCalendar.get(myCalen-
dar.MONTH) + 1>
<CFSET theDay =
myCalendar.get(myCalendar.DATE)>
<cfoutput>
Today is <b>#theMonth# / #theDay# /
#theYear#</b>
</cfoutput>
```

That's all for our first marriage of Java with ColdFusion!

The ability to access the entire Java classes' API from within ColdFusion provides a nice extension to what we can do with CF by itself. See the Java API reference at www.javasoft.com for a complete class reference. Assuming you're a curious reader, it's time to extend our simple example so that we access a self-made Java class, right? However, to do so you'll need at least some basic knowledge of Java and its concepts. As the scope of this article is limited and not a Java tutorial, I recommend that Java newbies take a brief look at the Sun Java Tutorial.

Accessing Your Own Classes

To access the self-made class example I coded a simple JavaBean (named "MakeSomewhat") that indeed does something (that's worthless and taken only to demonstrate its basic use). It simply takes a name (as a string) and returns three sentences (as an array of strings) with the name incorporated into them (see Listing 2). That way I can show you how to pass data to that class and get return data from it.

Such simple tasks can often be solved much easier with custom tags in CFML, but as I told you in the intro, Java components are better suited for some special tasks, such as image processing or spell checking. I coded the Java class MakeSomewhat.java as a bean; it has a para-

meterless constructor and so-called setter- and getter-methods to input data and extract data from it. This is the basic idea behind JavaBeans – to keep all internal properties private (encapsulated in a kind of black box) and communicate with them only via the setter/getter-methods. As you may have already guessed, the methods of this class are similar to these:

```
setName(name)
getName() : name
getSentences() : sentences[]
```

It's important to put a custom class into a package (optional when only using in Java, but a must when using with ColdFusion). So note the line "package test;" at the top of MakeSomewhat.java that indicates that it resides in a package named "test" (represented by a directory named "test" where you compile it). This is, of course, only for testing as Sun recommends you use more meaningful package names like "com.myDomain.myProject" where each token (separated by the dot) is a subdirectory. After compiling the class "MakeSomewhat.java" (using javac.exe or any Java IDE), the resulting file "MakeSomewhat.class" must be accessible by the ColdFusion Server. Therefore it has to be placed into one of the directories listed above as the ClassPath under the CF Admin, or the ClassPath setting has to be extended pointing to the directory where you've saved "MakeSomewhat.class".

After placing the file where it's accessible by the CF Server you can – with only a few lines of CFML code – use its whole functionality (see Listing 3). After instantiating an object from the MakeSomewhat class, feed it with the name "John Doe":

```
<CFOBJECT ACTION="CREATE" TYPE="Java"
CLASS="test.MakeSomewhat" NAME="mySomewhat">
<CFSET void = mySomewhat.setName("John
Doe")>
```

It returns an array of all three sentences with the name "John Doe" incorporated into them:

```
<CFSET SentenceArray = mySomewhat.getSen-
tences()>
```

Overloaded Methods

In object-oriented programming it's a common practice to overload methods (including constructors). This results in multiple methods/constructors with different parameter signatures (for example, setting color values in separate RGB values or in one hex string). There's even support for overloaded methods in ColdFusion, as long as the number of arguments are different. According to Allaire, future versions of ColdFusion will have fine enhancements that allow the user to use cast functions that allow method signatures to be built more accurately. To call another constructor instead of the default one of a class (that's being called automatically when accessing a public method of a class for the first time), you have to call it explicitly using the "init()" method with the appropriate arguments:

> "The ability to access the entire Java classes' API from within ColdFusion provides a nice extension to what we can do with CF by itself

```
<CFSET return = myObj.init(arg1, arg2)>
```

Java Exceptions

Like the exception-handling model we all know from ColdFusion, Java has an analogous type of exception model. The nice thing about this is that all exceptions thrown by Java classes invoked within CFML can be caught by the <CFTRY> <CFCATCH> tags. The message field of the <CFCATCH> variable contains the exception text thrown by Java including the class name.

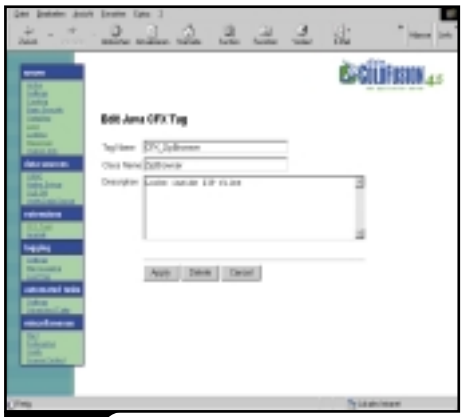


FIGURE 2: Registering CFXs in the CF Administrator

CFX Custom Tags in Java

In the introduction I mentioned three types of Java accesses within ColdFusion, including developing CFXs in Java. Using CFX is quite convenient and even easier than using Java classes within CFML. As all of you know, you can simply invoke any CFX custom tag (developed in Java or C++) as follows:

```
<CFX_MakeSomethingUseful times=42 text="The
quick brown fox jumps over the lazy dog">
```

This invokes a CFX with two attributes. Note that Java CFX custom tags have to be registered inside the CF Administrator with their class name and a description (see Figure 2). Be sure to use only the name of the class without the ".class" ending to avoid troubleshooting when CF can't find the CFX. Of course, the class file should be accessible to the CF Server's ClassPath specified in the CF Admin.

Even though it's easy to use CFX custom tags, it's more interesting to develop them. Let's take a look behind the scenes at CFX custom tags written in Java. As with all CFXs they can handle numerous tasks including the parsing and returning of attributes/variables, the parsing and returning of query recordsets (handy to output using CFML), the dynamic generation of HTML code for the client, and even throwing custom exceptions.

How are CFXs coded in Java? They share a method invoked from the CF Server process called "processRequest(Request, Response)" that takes a request- and response-object (much like the concept of classical servlets). The proper existence of this method is guaranteed as all CFX custom tags must implement the interface "custom tag" from Allaire in the package com.allaire.cfx located in c:\cfusion\java\classes\cfx.jar. The request object handed into the processRequest() method as a parameter holds all attributes of the <CFX_> invocation accessible through its attributeExists(attributename) and getAttribute- (attributename) methods. As expected, the response object has several methods for returning data into the calling template by setting variables back in the caller page (as the CALLER-scope in CFML <CF_>-CustomTags does). These methods of the response object include addQuery (name, columnarray), which adds a query result set into the caller template, which can be filled with data through the method setData() of Allaire's Query object available in the cfx.jar package.

Listing 4 provides a CFML source for invoking a nice CFX custom tag called *ZipBrowser* that can open and parse the zip files shown (see Figure 3). The Java

source code for that CFX (extended from Allaire's sample code inside the CF Studio help) is provided in Listing 5.

Basically, a CFX (written in C++, Delphi, or Java) can do whatever the targeted language supports. It can communicate with the CFML template using the predefined interfaces with the CFML page regarding the passing of attributes and the returning of data. Under the hood this CFX can use other classes to solve many different tasks. Go to Allaire's Tag Gallery (www.allaire.com/developer/gallery.cfm) to search for available CFX custom tags to solve certain tasks. Also note that many software vendors have wrapped their own Java and C++ APIs into ColdFusion CFXs to support easy connection of CFML with their products, such as credit card processing software, image and charting software, and much more.

What About EJBs?

As the JavaBeans' component model wasn't fitting well for transaction and database intensive tasks, a new component model was built. Named "Enterprise JavaBean," it takes a predefined set of interfaces and database transaction management features into account. The result was a complete server architecture built around it with products from IBM, BEA, and Sun, among others.

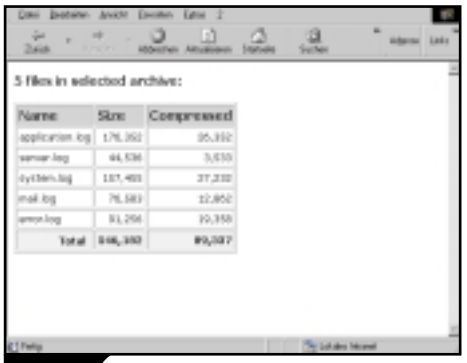


FIGURE 3: Results from our sample ZipBrowser CFX

Looking inside EJBs from the Java side would extend this article's scope into that of a book, so I'll only look briefly at EJBs from the ColdFusion side. To put it simply, you can treat an EJB as if it were just another Java class. (Please, all Java gurus out there, don't flame me on this; it's just from the usage of EJBs within CFML, not the development of EJBs!) There is, of course – even when only accessing EJBs from within CFML – some overhead compared to "just another Java class," such as fetching an initial context or looking up an EJB from its name. However, once you've instantiated an EJB object, you can treat that object within CFML like all other Java objects by accessing its methods and properties.

Look inside the CF Studio help for some examples of using an EJB within CFML. Be aware that to understand fully the use of EJBs, you must understand the concepts behind the EJB architecture (again, out of this article's scope). For the curious among you, look at Listing 6, which contains the sample code, taken from the CF Studio help, for accessing an EJB. If some of you are interested in reading an article about using EJBs with ColdFusion, e-mail me (see address at the end of this article).

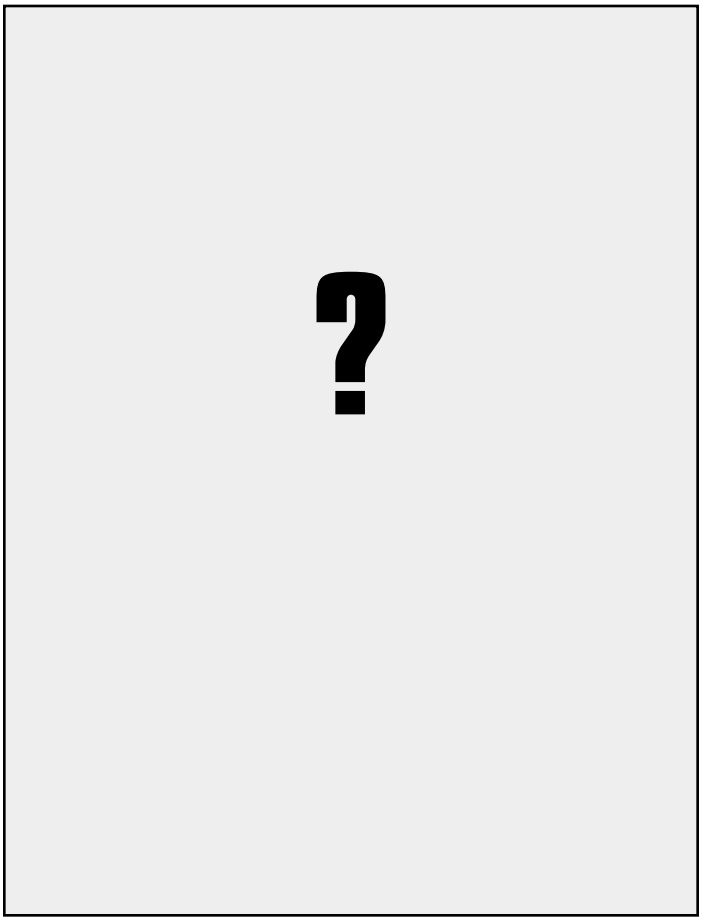
Summary

I hope this technical overview of what's possible with ColdFusion and Java may whet your appetite for more. What I personally like about ColdFusion is that the guys from Allaire constantly manage to connect CF to all kinds of standards out there, such as supporting all major native database drivers, Web protocols, directory services, data standards, distributed object technologies, and now, in CF 4.5, even Java through EJBs and normal Java classes as well as their CFX API. The best thing is that everything is there at your fingertips to use in the form of simple and efficient <tags>.

About the Author

Christian Schneider is a ColdFusion developer with over three years of intensive experience in developing CF-based intranet applications for banks and logistic corporations.

mail@christian-schneider.de



Listing 1: Our first fusion of CF with Java

```
<!--- // Create a java.util.GregorianCalendar Object // --->
<CFOBJECT ACTION="CREATE" TYPE="Java" CLASS="java.util.Grego-
rianCalendar" NAME="myCalendar">

<!--- Default Constructor is being called implicitly --->

<!--- Check if 2000 is a leap-year --->
<CFSET is2000LeapYear = myCalendar.isLeapYear(2000)>

<!--- Retrieve current date --->
<CFSET theYear = myCalendar.get(myCalendar.YEAR)>

<CFSET theMonth = myCalendar.get(myCalendar.MONTH) + 1>
<CFSET theDay = myCalendar.get(myCalendar.DATE)>

<cfoutput>
 <html>
 <body>
  Is 2000 a leap-year? <b>#is2000LeapYear#</b>
  <p>
  Today is the <b>#theMonth# / #theDay# / #theYear#</b>
 </body>
 </html>
</cfoutput>
```

Listing 2: Our sample bean MakeSomewhat.java

```
// MakeSomewhat.java
package test;

public class MakeSomewhat {
  // Properties
  private String name = "";
  private String[] sentences = new String[3];
  // Constructor
  public MakeSomewhat() {
       this.sentences[0] = "undefined";
  }

  // public setter
  public void setName(String aName) {
       this.name = aName;
       this.sentences[0] = "This is the first example to
integrate the name "+this.name+" in a sentence...";
       this.sentences[1] = "This is the second example to
integrate the name "+this.name+" in a sentence...";
       this.sentences[2] = "This is the third example to
integrate the name "+this.name+" in a sentence...";
  }


  // public getter
  public String getName() {
       return this.name;
  }

  // public getter
  public String[] getSentences() {
       return this.sentences;
  }

}
```

Listing 3: Using our sample bean within CF

```
<!--- // Using our MakeSomewhat bean // --->

<!--- Create a MakeSomewhat Object (note the package name
here) --->
<CFOBJECT ACTION="CREATE" TYPE="Java" CLASS="test.MakeSome-
what" NAME="mySomewhat">
<!--- Set the name --->
<CFSET void = mySomewhat.setName("John Doe")>

<!--- Retrieve the result array --->
<CFSET SentenceArray = mySomewhat.getSentences()>
```

```
<cfoutput>
 <html>
 <body>
 <!--- Loop over Array --->
 <cfloop index="i" from="1" to=#ArrayLen(SentenceArray)#>
  #SentenceArray[i]# <br>
 </cfloop>
 </body>
 </html>
</cfoutput>
```

Listing 4: Using our sample CFX

```
<html>
<body>

<!--- Retrieve the content listing of a real zip-file (here
c:\logfiles.zip) --->
<CFX_ZipBrowser ARCHIVE="c:\logfiles.zip" NAME="ZipContent">

<cfoutput>
 #ZipContent.RecordCount# files in selected archive:
</cfoutput>

<table border="1">
 <tr bgcolor="#cccccc">
  <td><b>Name</b></td>
  <td><b>Size</b></td>
  <td><b>Compressed</b></td>
 </tr>

 <cfset sumSize = 0>
 <cfset sumCompressed = 0>

 <!--- Display the zip-file's content --->
 <CFOUTPUT QUERY="ZipContent">
  <tr>
   <td>#ZipContent.Name#</td>
   <cfset sumSize = sumSize + ZipContent.Size>
   <td align="right">#NumberFormat(ZipContent.Size)#</td>
   <cfset sumCompressed = sumCompressed + ZipContent.Com-
pressed>
   <td
align="right">#NumberFormat(ZipContent.Compressed)#</td>
  </tr>
 </CFOUTPUT>

 <!--- Total --->
 <cfoutput>
  <tr align="right" bgcolor="##dddddd">
   <td><b>Total</b></td>
   <td><b>#NumberFormat(sumSize)#</b></td>
   <td><b>#NumberFormat(sumCompressed)#</b></td>
  </tr>
 </cfoutput>

</table>

</body>
</html>
```

Listing 5: The sample <CFX_ZipBrowser> extended from Allaire's CF Studio help

```
// ZipBrowser.java

import com.allaire.cfx.*;
import java.util.Hashtable;
import java.io.FileInputStream;
import java.util.zip.*;
public class ZipBrowser implements CustomTag
{
   public void processRequest( Request request, Response
response )
      throws Exception
   {
    // validate that required attributes were passed
      if (  !request.attributeExists( "ARCHIVE" ) ||
            !request.attributeExists( "NAME" ) )
```

```
      {
         throw new Exception(
            "Missing attribute (ARCHIVE and NAME are both "
+
            "required attributes for this tag)" ) ;
      }

    // get attribute values
      String strArchive = request.getAttribute( "ARCHIVE" )
;
      String strName = request.getAttribute( "NAME" ) ;

    // create a query to use for returning the list of files
      String[] columns = { "Name", "Size", "Compressed" } ;
      int iName = 1, iSize = 2, iCompressed = 3 ;
      Query files = response.addQuery( strName, columns ) ;

    // read the zip file and build a query from its contents
      ZipInputStream zin = new ZipInputStream( new FileIn-
putStream(strArchive) ) ;
      ZipEntry entry;
      while ( ( entry = zin.getNextEntry()) != null )
      {
         // add a row to the results
         int iRow = files.addRow() ;

         // populate the row with data
         files.setData( iRow, iName, entry.getName() ) ;
         files.setData( iRow, iSize,
String.valueOf(entry.getSize()) ) ;
         files.setData( iRow, iCompressed,
String.valueOf(entry.getCompressedSize()) ) ;

         // finish up with entry
         zin.closeEntry() ;
      }
      // close the archive
      zin.close() ;
   }
}
```

Listing 6: A sample taken from the CF Studio help of accessing an EJB within CF

```
<!---
You would use CFOBJECT to create and call all the appropri-
ate objects. The sequence below assumes that the WeblLogic
JNDI is used to register and find EJBHome instances :
--->

<cfobject action=create  type=JAVA
class="weblogic/jndi/Environment"  name=wlEnv>
<CFSET ctx = wlEnv.getInitialContext()>
<CFSET ejbHome = ctx.lookup("statelessSession.TraderHome")>

<CFSET trader = ejbHome.Create()>
<CFSET value = trader.shareValue(20, 55.45)>
<cfoutput>
  Share value = #value#
</cfoutput>
<CFSET value = trader.remove()>

<!---
The CFOBJECT tag creates the WebLogic Environment object,
which is then used to get the InitialContext. The context
object is used to look up the EJBHome interface. The call to
create() results in getting an instance of stateless session
EJB.
--->
```

CODE LISTING

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

Making the Case for ColdFusion

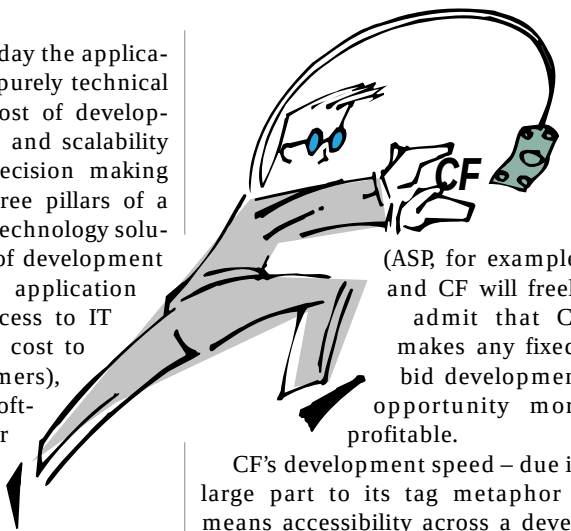
A financially sensible technology solution

BY
RUSTIN
ROSS

The application server market is red-hot with vendors, open-source software developers, and application-service providers competing to be the platform for business Internet solutions. Businesses, better educated about the relationship between technology and their bottom line, are again looking at technology investments as they would any other...money matters.

At the end of the day the application must work – a purely technical decision. But the cost of development, maintenance, and scalability should dominate decision making thereafter. These three pillars of a financially sensible technology solution are composed of development time (how long an application takes to create), access to IT talent (what does it cost to get good programmers), and the cost of the software/hardware over time (total cost of ownership).

On this score Allaire's ColdFusion reigns supreme. However, as the CIO for a Chicago IT recruiting firm and a ColdFusion developer, I believe Allaire has done a poor job of communicating this critical message to the marketplace.



Cost of Development

Rapid application development (RAD) technologies (like CF) get a bad rap in some hard-core technology circles, but from a business perspective it just makes sense. The old saw, "Time is money," still cuts. CF's RAD capabilities are unmatched by anything short of an Application Service Provider solution for time-to-market, translating into lower fees for contract or permanent programmers. Tangentially, consulting firms that code in another language (ASP, for example) and CF will freely admit that CF makes any fixed-bid development opportunity more profitable.

CF's development speed – due in large part to its tag metaphor – means accessibility across a development team, where senior developers and novices can work the same application in concert. From a business perspective this means less expensive HTML coders with a bit of training can make meaningful contributions to development, saving serious cash. By comparison, there's simply no way to plug an HTML-savvy developer into a .jsp or .asp solution, no matter how simple.

Beyond the tag metaphor, Allaire's "Developer's Exchange" Tag Gallery provides developers with a rich library, prebuilt functionality, and access to highly specialized solutions that shorten development time. Savvy CF coders have also found the Exchange to be a source of secondary revenue for their code, and end users might benefit from packaging the objects their e-commerce projects generate for resale.

Finally, as CF is an architecture that easily supports the inclusion of components from other popular languages, developers can incorporate prebuilt, complex objects into their applications with only a few lines of code. A commitment to CF doesn't have to be a monogamous relationship.

Cost of Maintenance

Debugging and expanding an application is a financial threat to businesses. Competitive advantages through technology are short-lived and expensive to maintain.

A solid application architecture anticipates the changing nature of e-strategies and the migration toward emerging technologies and standards that provide a competitive advantage. Application servers that lack support for a wide array of diverse technologies cripple a business's ability to compete and cost that organization more when it comes to duplicating like functionalities from other server packages.

ColdFusion, which is accessible to numerous document object models, programming languages, object brokers, and nearly anything that can be custom-coded in C++ or Java, can be the only exhibit in the museum of technological diversity. Again, CF's tag-based metaphor

reduces those very complex functional concepts like XML, for example, into comparatively easy-to-understand tags like CF_WDDX and encapsulates the package in a RAD environment.

By example, the cost and time needed to implement a CF_WDDX solution over a comparable data transfer channel using Perl or Java provide ample evidence of CF's ability to provide time-based savings.

Scalability

The most frequent criticisms still levied against ColdFusion are concerns about the server's scalability. Despite Allaire's addition of load balancing and improved session management, the message hasn't reached the masses.

A properly architected CF solution can scale across multiple servers geographically dispersed and can support as many users as the underlying hardware and OS can sustain. That said, it seems Allaire's offerings on NT are actually more expensive to deploy across multiple installations than a comparable .asp solution (based on Microsoft's competitive upgrade pricing). Fortunately, with ColdFusion now shipping for Linux, a combination of CF, MySQL, and Linux can scale at a very low cost.

> "The old saw, 'Time is money,' still cuts

Conclusion

Sex sells...even in Newton, Massachusetts. Java and the J2EE specification are sexy and expensive technologies. Moreover, C#, XML, and application service provider–based solutions may pave new directions in the processing and delivery of data. But Allaire's still unparalleled strength is, I believe, based on the simplification of complex and powerful capabilities and on enabling the soft-core and low-cost IT professional to deliver professional results.

Allaire delivered cost-effective solutions at a time when skepticism about the Internet prevented bonfires with VC dollars as kindling. As e-business is again under the gun to cut costs and produce profits, the CF community and Allaire have a second chance to sing the praises of ColdFusion that would be music to any CFO or CEO's financial ears.

ABOUT THE AUTHOR

Rustin Ross, a ColdFusion developer for over three years, is CIO at Roy Talman & Associates, an IT recruiting firm. Prior to his current position, Ross was the COO for a national equipment liquidation firm and the architect of a number of proprietary applications for that industry.

RUSTIN@ROYTALMAN.COM

CFDJ news

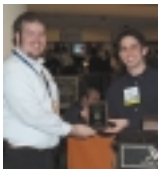
Macromedia Announces Next-Generation Dreamweaver

(*Monterey, CA*) – Macromedia has introduced the next generation of their Dreamweaver Platform featuring all new versions of Macromedia Dreamweaver 4, Dreamweaver UltraDev 4, and Fireworks 4. This major upgrade unifies the design, production, and development workflows, and enables efficient authoring at every step of the development process.

www.macromedia.com

Macromedia, Allaire in Joint Development Effort

(*Washington, DC*) – Macromedia, Inc., and Allaire Corp. have announced plans to jointly develop a toolkit that will enable Allaire ColdFusion and JRun developers to use the user interface and media capabilities of the Macromedia Flash Player. The toolkit, code named *Harpoon*, will provide developers with a set of common user interface controls for Web applications built with ColdFusion and JRun servers. Harpoon was previewed at the 2nd Annual Allaire Developer Conference and is available now through a free public beta. It will be free to developers in early 2001 on www.macromedia.com and www.allaire.com.



Macromedia is presented with awards at the Allaire Conference in Washington, DC, by *CFDJ* editor-in-chief Robert Diamond.

SYS-CON Names Silverman VP, Marketing and Sales, *Wireless Business & Technology*

(Montvale, NJ) – **SYS-CON Media, Inc.** (www.sys-con.com), headquartered in Montvale, New Jersey, has named Miles Silverman as vice president of marketing and sales for its newest title, ***Wireless Business & Technology*** (www.wireless-magazine.com).

"We're delighted to have Miles Silverman on board," said Fuat Kircaali, founder and CEO of **SYS-CON**. "Miles will have a critical role in the launch of our most recent and largest new title, ***Wireless Business & Technology***."

"I'm very excited to join the **SYS-CON** team and look forward to being part of one of the biggest magazine launches of recent years," said Silverman. "***Wireless Business & Technology*** aims to be the world's leading publication serving the wireless industry and Internet technology developers utilizing new and emerging wireless technologies around the globe."

Silverman spent years managing media buying/planning for such New York ad agencies as Dentsu/Young & Rubicam and Ted Bates before embarking on his career in print publishing. His publishing track record includes a 10-year run with 101 Communications LLC (formerly SIGS Publications, Inc.), where as ad director he played a key role in the founding and publishing of two successful titles: *The X Journal* and *Java Report*.

ADVERTISERS INDEX

ADVERTISER	URL	PH	PG
ABLECOMMERCE	WWW.ABLECOMMERCE.COM	360.253.4142	2-3
ACTIVEPDF			4
AFFINITY	WWW.AFFINITY.COM	800.646.0662	17
ALLAIRE CORPORATION	WWW.VUE.COM/ALLAIRE	877.460.8679	41
ALLAIRE CORPORATION	WWW.ALLAIRE.COM	888.939.2545	35, 55
ANDREWS TECHNOLOGY	WWW.ANDREWSTECHNOLOGY.COM	888.393.0159	30
BIZNIZWEB,INC		281.367.4016	57
CAREER OPPORTUNITIES	WWW.SYS-CON.COM	800.582.3089	57
CF HOSTING	WWW.CFHOSTING.COM	770.982.8382	25
CFXHOSTING	WWW.CFXHOSTING.COM	866.239.4678	13
COLDFUSIONDB.COM	WWW.COLDFUSIONDB.COM		57
CONCEPTWARE AG	WWW.CONCEPTWARE.COM	49(0)6196-4732-0	15
CORDA TECHNOLOGIES	WWW.POPCHART.COM	801.802.0800	23
CYBERSMARTS	WWW.CYBERSMARTS.NET		62
CYSCAPE		800.932.6369	58
EVOLUTION B	WWW.EVOLUTIONB.COM/SYNERGYFORFREE		59
INTELIANT	WWW.INTELIANT.COM	800-815.5541	33
INTERLAND	WWW.INTERLAND.COM	877.887.3864	7
INTERMEDIA	WWW.INTERMEDIA.NET	800.379.7729	60
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	49
MACROMEDIA	WWW.MACROMEDIA.COM	415.252.2000	18-19
MISTRAL DESIGN GROUP	WWW.PURPLEHOSTING.COM		36
NETMORF	WWW.NETMORF.COM	877.533.MORF	9
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	37
SITEHOSTING.NET	WWW.WEARECFHOSTING.COM	877.NTHOSTING	43
THESHORTLIST.COM	WWW.SHORTLIST.COM		24, 54
UUNET	WWW.INFO.UU.NET/WEBHOSTING.COM	877.465.7107	11
VIRTUALSCAPE	WWW.VIRTUALSCAPE.COM	877.VSCAPE4	27
WIRELESS BUSINESS & TECH	WWW.WIRELESS-MAGAZINE.COM.		53
WOODBOURNE SOLUTIONS	WWW.WOODBOURNESOLUTIONS.COM	877.668.8575	51
XML DEVCON EUROPE	WWW.XMLDEVCONEUROPE2001.COM		47