

ColdFusion Developer's Journal

ColdFusionJournal.com

July 2003 Volume: 5 Issue: 7

Announcing...

2003 CFDJ
READERS'
CHOICE
AWARDS



VOTE NOW!

GO TO WWW.SYS-CON.COM

Editorial

Red Skies Smiling at Me...

Robert Diamond page 5

CF Community

Tales from the List

Simon Horwith page 7

Design Patterns

Singleton Pattern

Brendan O'Hara page 20

CFUGs

A Forum of Support and Technology for Web Professionals

page 48

ColdFusion News

page 50

RETAILERS PLEASE DISPLAY
UNTIL SEPTEMBER 30, 2003

\$8.99US \$9.99CAN

08>



0 71486 02689 1

**SYS-CON
MEDIA**

DRK:

Christian Cantrell 8

Macromedia's DevNet Resource Kit: Just the Beginning

Dreamweaver: One Product for All:
Dreamweaver MX

The possibilities are endless

John Wilker

12

Foundations: The Best Recipe
Cooking is a lot like programming

Hal Helms

16

Security: Defeating DoS Attacks

Protect your apps from denial of service attacks

Joe Danziger

26

**<BF> on <CF>: When One ColdFusion
Is Not Enough**

One server, lots of ColdFusions

Ben Forta

28

CF INCLUDEs: The Secret Powers of Includes

CFINCLUDE is much more than just a way to 'pull in code'

Charlie Arehart

34

JavaScript: JavaScript Without the Headaches

Spare yourself hours of frustrating debugging

Tom Peer

38

CF and Java: Extending ColdFusion with Java

Full text searching using Jakarta Lucene

Aaron Johnson

44

NETQUEST

www.nqcontent.com

EDGE WEB HOSTING

www.edgewebhosting.net

ACTIVEPDF
www.activepdf.com

Jeremy Allaire, *CTO, macromedia, inc.*
Charlie Arehart, *CTO, systemanage*
Michael Dinowitz, *house of fusion, fusion authority*
Steve Drucker, *CEO, fig leaf software*
Ben Forta, *products, macromedia*
Hal Helms, *training, team macromedia*
Kevin Lynch, *chief software architect, macromedia*
Karl Moss, *principal software developer, macromedia*
Michael Smith, *president, teratech*
Bruce Van Horn, *president, netsite dynamics, LLC*

editorial

editor-in-chief

Robert Diamond robert@sys-con.com

technical editors

Charlie Arehart charlie@sys-con.com
Raymond Camden raymond@sys-con.com

editorial director

Jeremy Geelan jeremy@sys-con.com

executive editor

Jamie Matusow jamie@sys-con.com

editor

Nancy Valentine nancy@sys-con.com

associate editors

Gail Schultz gail@sys-con.com
Jean Cassidy jean@sys-con.com

assistant editor

Jennifer Van Winckel jennifer@sys-con.com

production

production consultant

Jim Morgan jim@sys-con.com

art director

Alex Botero alex@sys-con.com

associate art directors

Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Tami Beatty tami@sys-con.com

contributors to this issue

Charlie Arehart, Christian Cantrell, Joe Danziger,
Ben Forta, Brendan O'Hara, Hal Helms, Simon Horwith,
Aaron Johnson, Tom Peer, John Wilker

editorial offices

SYS-CON MEDIA

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

©copyright

Copyright © 2003 by SYS-CON Publications, 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.

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ

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.

All brand and product names used on these pages
are trade names, service marks, or trademarks
of their respective companies.

Red Skies Smiling at Me...

No, that's not a misprint, and I haven't lost my mind quite yet. (Some friends might tell you differently, as might the leprechaun in my closet, but we'll save that for a future editorial.) Red Sky is the code name for the next ColdFusion MX release – a

FREE (one of the greatest words in the English language) updater being released this summer

that helps take CFMX up to new levels of greatness.

We've been playing with it here at **CFDJ** for the past few weeks as part of the beta program, and I'm proud to report that SYS-CON.com (putting our servers where our mouths are) will also be one of the early adopters when it's released this summer. To join the beta program, visit

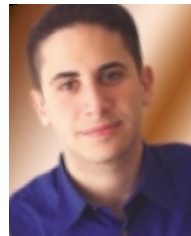
www.macromedia.com/go/cfmxbeta.

We're lining up lots of great Red Sky coverage starting in next month's issue, including everything you need to know to get up and running and to start taking advantage of the new features designed to make your lives easier and your servers happier – not to mention faster and more stable as well.

Red Sky has addressed over 400 different customer issues with ColdFusion MX. Macromedia has been listening to the suggestions (and the complaints) that you've been making and this new update delivers on many different fronts, many of which you can start taking advantage of right upon install. They've done their best to address the most significant issues and requests for ColdFusion MX.

The first thing you'll notice is an extremely simplified install and migration program that will make upgrading a simple task whether you're coming from CFMX or an earlier release. If you're still running CF 4/5 because you're reluctant to upgrade, and you've been scared off by some of the stories about CFMX's original installer, then this is the release you've been waiting for. One of the first things you'll notice after installation is that it's faster...and in some cases...it's a lot faster. Performance has been streamlined overall, with a variety of contributing factors that will help to speed up your server and your CF applications.

First, the CFML compiler has been optimized, which, off the bat, results in increased runtime performance and stability. Adding to that are updated JDBC/ODBC drivers for database access, and they're now distributing it



By Robert Diamond

with the newest Java Virtual Machine from Sun. For those using Microsoft Access for smaller scale sites – or if you're gutsy, larger scale sites – there's a new Unicode driver that should relieve some of the headaches that you may currently be facing with international or other characters.

There's new support for Windows 2003 Server – the new platform from Microsoft that we'll be covering shortly as well. Stay tuned for full details of all this and more in upcoming issues of **CFDJ** as we guide you through everything you need to know!

Exciting times are ahead in the world of ColdFusion, which is making preparation even more exciting for **CFDJ**'s panel discussion that will take place June 21 at CFUN-03 in Maryland. The discussion will have passed by the time this issue is in your hands, but we'll be providing coverage of that as well.

It's a good time to be a CF developer, and an even better time to get involved with the CF community, the events, and the user groups if you're not already taking part. Just as we strive to make **CFDJ** each month – they are a fantastic resource.



About the Author

Robert Diamond is vice president of information systems for SYS-CON Media, and editor-in-chief of **ColdFusion Developer's Journal**. Named one of the "Top thirty magazine industry executives under the age of 30" in *Folio* magazine's November 2000 issue, Robert holds a BS degree in information management and technology from the School of Information Studies at Syracuse University. Visit his blog at www.robertdiamond.com.

robert@sys-con.com

NEW ATLANTA COMMUNICATIONS

www.newatlanta.com

Tales from the List

Cheers Mates!

Those of you who frequent the CFDJList already know that in early May of this year I relocated from Washington, DC to London, England to work as a private consultant. Not only does this mean that I can now enjoy a pint of fantastic ale on a much more regular basis, but also that I was able to attend and present at this year's CF-Europe conference. This is my first article written from "across the pond." For the benefit of those of you stateside and elsewhere, the CF-Europe conference and the impressions it left me with, are the focus of this month's **Tales from the List** column.

CF-Europe 2003, billed as the "European Macromedia Developer Conference" was held in London at the Olympia Conference Centre, May 29-30. The conference was organized and run by Niklas Richardson (the UK ColdFusion User Group Manager) and by staff members from Prismix Ltd., with much additional help from Lucas Sherwood - Macromedia's product evangelist in the European Union.

Stephen Morretti, Northern UK CFUG manager and frequent CFDJList contributor was on hand as conference photographer as well as to help with other general conference administrative tasks. Over 20 CFUGs from the European Union participated in the event, which was attended by roughly 250 developers. The show gave attendees the opportunity to attend any of 27 presentations in addition to the two keynote speeches and a panel discussion at the end of Day 2. While I was not able to attend every presentation, I was able to see part or all of many of them.

This conference, more than any I've attended in the past, focused intensely on rich Internet application (RIA) development, development of applications that blend a Flash presentation layer with a ColdFusion server-side back end. Ben Forta's keynote on Day 1 centered on Macromedia's new Data Connection Components for rapid development of rich Internet apps. Aral Balkan gave a presentation on using the Data Connection Components with XML (as well as some insight as to how to properly use the compo-



By Simon Horwith

nents so as not to jeopardize security). Simon Frank talked about rich Internet application ROI (return on investment). Michael Quack gave a presentation on managing rich Internet application projects. Philip Cielen talked about building next-generation applications with Flash and ColdFusion. Even the panel discussion was a rich Internet application development Q&A session.

Without a doubt, RIA development was a major focus.

Not everyone was there to talk about rich Internet application development. Coincidentally, it was the American presenters who led most of the server-side topic discussions. Charlie Arehart spoke about JSP as well as about some of the hidden gems in ColdFusion MX. Michael Smith spoke about Flip (Fusebox Lifecycle Process). I spoke about

—continued on page 10

About the Author

Simon Horwith has been using ColdFusion since version 1.5. He is a Macromedia Certified Advanced ColdFusion and Flash developer and is a Macromedia Certified instructor. In addition to administering the CFDJList List Serve and presenting at CFUGs and conferences around the world, he has also been a contributing author of several books and technical papers. Originally from the Washington, DC area, Simon is currently working as a private consultant in London, England.

simon@horwith.com

A
COMPREHENSIVE
AND
DIVERSE SET
OF
COLDFUSION
CONTENT

Macromedia's DevNet Resource Kit: Just the Beginning

What is the Macromedia DevNet Resource Kit, and how can I get it? As both a Macromedia DevNet Resource Kit (DRK) product manager, and the Macromedia Server Community Manager, I hear these questions often from both the online and offline ColdFusion communities. Starting with volume 3, the DRK has become extremely relevant to ColdFusion developers, and future volumes will be no exception.

The DRK is a developer resource that Macromedia releases quarterly containing components, extensions, utilities, sample applications, and reference materials for several Macromedia products. DRK volumes 1 and 2 heavily emphasized Macromedia

Flash and Dreamweaver; however, the goal of DRK 3 was to provide the ColdFusion community with a comprehensive and diverse set of ColdFusion content, including ColdFusion components, user-defined functions, custom tags, sample applications, book chapters, and articles. And although I can't go into detail about future DRKs, I can say that DRK 3 is just the beginning.

ColdFusion content on DRK 3 is divided into three parts: custom tags and components, utilities, and sample applications. In this article, I discuss my favorite projects in each category.



By Christian Cantrell

Custom Tags and Components

DRK 3 contains several custom tags and components that are designed to be integrated into other ColdFusion projects or utilities. Rather than being standalone applications, they can be considered application building blocks.

The Calendar Component

The calendar component is a model of a single page in a calendar. All you have to do is set the month and the year, then call `getPage` to get back a two-dimensional array that models the columns and rows of a calendar page. The two-dimensional array will always have seven columns (for every day of the week) and the right number of rows for the specified month and year. The values of the elements in the two-dimensional array will either be the appropriate day of the month, or -1 when they fall outside of the month's range.

The calendar component is really designed to encapsulate the messy logic you need to render calendar pages in HTML. It has all the functions you need to quickly and easily add a calendar to your site.

The Init Custom Tag

The init tag is a sophisticated encapsulation technique many developers already use to initialize and configure their applications. Any code in the body of the init tag is only executed the first time the tag is encountered during the life span of the application, so it's a great place to put your applications' initialization routines, like defining application variables for data sources, parsing XML files, etc. It is thread-safe, but not entirely synchronized, so you can use it in your `Application.cfm` file without worrying about single-threading your application and dramatically slowing it down. The init tag also allows you to dynamically reevaluate any code in its body simply by passing a special query string parameter in to the page that contains the tag, so you can update your configuration without having to restart your server.

The Recordpager Custom Tag

The recordpager custom tag automatically generates linked page numbers to allow users to browse through multiple pages of records returned from a database query. It is used in circumstances where you want to limit the number of records to display on a single page, but still allow users to access the rest of the records on subsequent pages. The generated page numbers are linked to a page that you specify, or will automatically default to the current page since paging through record sets usually means making multiple requests to the same page. The recordpager tag will also automatically pass the specified page the record number that it should start with when displaying the next set of records, as well as optionally allow you to pass in a struct with name/value pairs that also need to be passed to other pages of query results. Finally, you can pass style class definitions in to the recordpager in order to adapt its look and feel to any application.

Utilities

DRK 3 contains a set of ColdFusion projects that are considered utilities, which means they are more extensive than just single tags and components, but are still designed to be integrated into larger projects. Here are two of my favorites:

Data File Access (DFA)

DFA allows ColdFusion developers to interact with XML and comma-delimited files (CSVs) as though they were database tables. The easiest and most flexible way to store and retrieve data in an application is to use a database and SQL; however for smaller projects, creating an entire database can be more hassle than it's worth. With DFA, you get all the advantages of using SQL to both save and retrieve data, but the underlying data structure is either an XML or CSV file.

Let's say you decide to put together a quick survey to allow your co-workers to vote on their favorite activity for a company picnic. You want to be able to save and retrieve the results quickly using SQL, but you might not want to go through the hassle of setting up a database. Using DFA, you can use a simple XML or CSV file to store your data while still being able to leverage the convenience of SQL. DFA even comes with the `dfaQuery` tag, which works almost exactly like the `cfquery` tag you are already accustomed to using.

Lindex (Lucene Indexing Engine)

One of the most powerful aspects of ColdFusion MX is that it is implemented almost entirely in Java. Writing utilities that leverage Java is less a matter of integrating ColdFusion MX and Java and more a matter of naturally using Java to extend ColdFusion. The fact that Java can be used so easily to extend ColdFusion means that developers can leverage all of the open-source Java projects out there, such as those being developed under the Apache Jakarta project.

DRK 3 contains a project called Lindex, which stands for Lucene Index. Lucene is an open-source Apache Jakarta project for creating and searching document collections. ColdFusion comes bundled with the Verity text search engine; however, the advantage of using Lindex to integrate Lucene into your applications is that since it is pure Java, it will run on any platform that ColdFusion MX runs on, including Macintosh OS X. That means you can develop and deploy your ColdFusion MX application on two separate platforms without having to worry about compatibility issues.

Lindex works almost identically to Verity in that you create collections which you then populate with text, HTML, PDF, and ColdFusion files. Lindex includes both a ColdFusion and a Java API, as well as an administrative interface very similar to the ColdFusion Verity interface.

Sample Applications

DRK 3 contains two sample ColdFusion applications. You can use the ColdFusion sample applications on the DRK to learn best practices and see examples of development techniques and principles, or you can simply install them for your own use.

Macromedia XML News Aggregator

The Macromedia XML News Aggregator, or MXNA, is an extremely useful application for gathering, parsing, storing, and displaying RSS news feeds. RSS, or Really Simple Syndication, is an XML format that many news sites and weblogs publish in order to allow RSS aggregators to syndicate their content. For example, my weblog (www.macromedia.com/go/cantrell) does not just contain an HTML version of my posts, but it also produces an XML version, as well (www.macromedia.com/go/ccantrell_rss). Anyone is welcome to use my RSS feed to either aggregate the content of my weblog locally on their computer, or to republish the content of my XML feed anywhere they want.

MXNA is based on the source code for the Web site fullasagoog.com, which is a very popular aggregator of Macromedia MX – related weblogs. The MXNA source code on DRK 3 has a very different look than fullasagoog, however, and has a few extra features. You can see a production version of MXNA focusing on Macromedia MX technologies, technology news, and Web development in general, running at www.macromedia.com/go/weblogs.


If you go to Macromedia's version of MXNA, you will find that we are aggregating somewhere in the neighborhood of 150 feeds. Imagine trying to keep up with 150 news sites and weblogs which make anywhere between one and probably a dozen posts a day. MXNA allows you to visit a single site, browse by category, and quickly find posts that are relevant to you.

Running your own version of MXNA will allow you to customize both your feeds and your categories. At Macromedia, we run one external version of MXNA (at the URL above) and one internal version which we use to aggregate things like internal project status reports. With MXNA, you can easily set up your own personal RSS aggregator, or launch your own RSS portal.

RSS 1.0 XML Feed Creation and Management Application (RSSify)

RSSify is a Rich Internet Application, or RIA, which means the user interface is implemented in Macromedia Flash MX while the back end is implemented as a ColdFusion component. RSSify is an excellent example of integrating Macromedia Flash and ColdFusion using Macromedia Flash Remoting in order to achieve a richer, more efficient user experience.

RSSify provides users with a simple and intuitive interface for creating and managing RSS XML feeds. The interface allows users to import, save, load, delete, and author RSS feeds, as well as publish them to a server where they can be made accessible to the rest of the world. The Macromedia Flash interface handles the creation and formatting of the XML document while the ColdFusion back end handles the persistence and publishing of the feeds. RSSify is perfect when you need to generate an RSS news feed (to be aggregated by MXNA, for instance), but you don't already have an automated process for doing so.

To find out more about these and other ColdFusion projects on DRK 3, visit the DRK 3 home page at www.macromedia.com/go/drk3. To find out more about how you can get DRK 3 and future versions of the DRK, visit the Macromedia DevNet subscription page at www.macromedia.com/devnet/subscriptions. 

Note: Portions of this article were originally published on the Macromedia DevNet Resource Kit, Volume 3.

About the Author

Christian Cantrell is the Macromedia Server Community Manager. He has been developing large-scale, Web-based applications in ColdFusion, Java, JSP, and Macromedia Flash for the last five years. He is the author of numerous tutorials and white papers, and is coauthor of Flash Enabled: Flash Design & Development for Devices. Keep up with Christian by reading his blog at www.macromedia.com/go/cantrell.

cantrell@macromedia.com

cf community —continued from page 7

the Data File Access API included on DRK3. Shlomy Gantz gave one of the most impressive presentations I've ever seen. His topic was the least technical (Project Management) but was probably the most relevant for all in attendance.

The keynote on Day 2 was given by Tim Buntel – Macromedia's ColdFusion product manager. He spoke about the misconceptions some people have about ColdFusion and its future and what we developers can do to help dispel these myths when the opportunity arises.

I think that, more than anything, I walked away from the conference with an understanding of where Macromedia developers are heading and where they're being led by Macromedia, not just here in the UK, but in general. The developers here do tend to be a bit more "no nonsense" about things (which I personally think is great). They love Macromedia products and they will continue to use them as long as

Macromedia provides the best tools for the job.

I found that the developers at the conference were extremely enthusiastic about what they saw in the presentations, and are eager to see ColdFusion gain in popularity in the European market. Many of the developers I spoke with seemed more intent on adding Flash MX to their development arsenal now that they've had some firsthand exposure to what the "rich Internet experience" is all about (from both a user and a developer perspective).

As you can imagine, CF-Europe was not just an educational summit; it also served as an opportunity for developers to socialize with one another. For me personally, it was both a chance to finally meet so many of the folks in the European community whom I've dealt with online for years as well as an opportunity to feel out what the overall development community in Europe is like compared to that in the U.S. I am happy to report – and was not at all surprised to find – that developers in Europe

are just as relaxed and friendly as they are in the States (maybe more). I was also very impressed with the level of professionalism shown to the conference presenters. Audiences generally sat very quietly and attentively during the presentations up until the very end of each... at which point they very enthusiastically and energetically showed the speaker their appreciation.

In all, CF-Europe was a great conference for everyone – speakers, organizers, and attendees. I have to say that above all else, I was most impressed with Macromedia's strong showing of continuing support to help ColdFusion developers take their applications and their skills to the next level. Not only do I think this is a good indicator of the theme we can expect at the MAX conference this November, but I imagine that by then the message will be even stronger – especially with the additions (I assume) by that time of Macromedia Central, the next version of the ColdFusion Application Server, and who knows what else, to the Macromedia line of products. 

COOLFUSION

www.coolfusion.com

One Product for All: Dreamweaver MX

The possibilities are endless

Dreamweaver has been around for years. We've all seen it in some version or other, but it has finally come into its own in the most recent version, MX.

Dreamweaver has always been a tool for creating code. Early versions had some understanding of ColdFusion, and subsequent versions knew some ASP. However, it was always regarded as a designer's tool, having only (at first) a WYSIWYG interface and very poor interaction with non-HTML code (ASP, PHP, ColdFusion, etc...).

Times have changed, and so has Dreamweaver, all for the better. Dreamweaver has evolved from its humble (though widely popular) designer tool beginnings, going through a phase where it was split into two products – Dreamweaver and Dreamweaver UltraDev – to where it is now, one product for all – Dreamweaver MX.

Shortly after the merger of Allaire and Macromedia, ColdFusion Studio and HomeSite became HomeSite+. The push was for all of us to use Dreamweaver MX. HomeSite+ was added to Studio MX as an “extra.” I say it's an extra because it's not billed as part of Studio MX and must be installed separately (HomeSite+ can be found in



By John Wilker

the HomeSite+ folder on the Studio MX CD. Dreamweaver was enhanced in its evolution to MX, bringing more of a developer interface with the available Code view for editing of raw code, as well as its traditional WYSIWYG development. Few programs can so easily move between two such

different environments. Dreamweaver does so with relative ease.

HomeSite+ was initially a sour grape for most ColdFusion Studio users. It was not customizable and lacked the features we liked. A few months ago, however, an update was released that makes HomeSite+ a major option for those who do not want the overhead that comes with Dreamweaver. You can customize the menus, relocate panes, pretty much everything we missed in Studio (see Figure 1).

(The Updated HomeSite+ installer can be found at: www.macromedia.com/cfusion/tidrc/index.cfm?product=homesite) For more information on HomeSite+ and its features, you can see the June 2003 **CFDJ** article, “Getting into HomeSite+,” Volume 5, issue 6).

Neat New Things

Like all new versions of a product, Dreamweaver has better, cooler, more powerful functionality than its predecessors, and some of that cool functionality will appeal to those of us who still hand code; remember, Dreamweaver MX is more than just the next version of Dreamweaver; it is also the replacement of a powerful tool with an enormous user base.

Dreamweaver plays very well with others now. XML and Style Sheet documents can be seamlessly integrated into your code like never before. By linking a DTD (that Dreamweaver can access) to your page, the Attributes and values of the XML file become available. Context menus allow you to work with XML with ease. Gone are the days of having to remember all of the styles you defined in your CSS file. When you link it to your page, Styles become context for class and ID tag attributes. No more having to have the CSS file open all the time (see Figure 2).

The only drawback, or rather shortcoming, in this handy functionality is that modular code seems incompatible. Pages that have the basic HTML file tags in different modules or includes don't get the benefit of the context feature. Only pages with the complete HTML tags do. Hopefully this is something that will be addressed sooner or later.

Site vs Project

If you remember the Project pane in Studio, you'll understand the basics of the

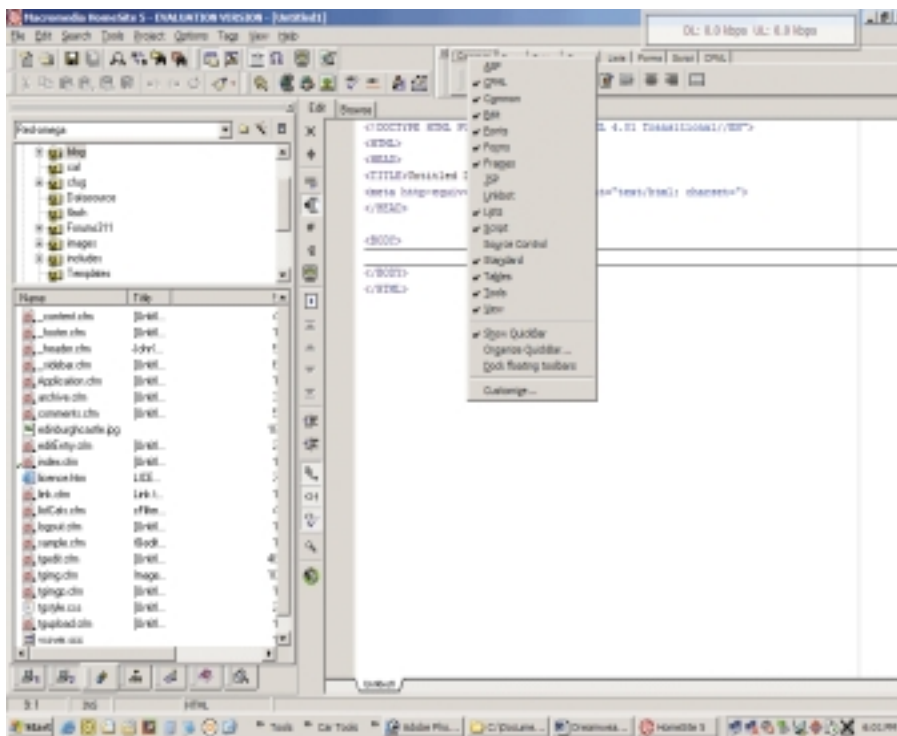


Figure 1: The finalized application being initialized and then displaying the DesDev feed.

Site pane in Dreamweaver (see Figure 3). The Site pane is a much more powerful tool than the Project pane was. Setting up a site is a lot more involved as far as your options and configuration choices than the old Project wizard.

Sites allow you to manage your code locally, on the testing server, as well as from a remote location, all from the same pane. You can quickly switch between views and even promote code easily. The local files can be integrated with a source control tool like Microsoft's VSS.

A few neat side benefits to the new Site pane include the ability to automatically generate a site map – talk about handy! You can also have Dreamweaver manage design notes. These notes are kept locally to make notes on pages and the whole site. Dreamweaver manages them for you. You can also take stock of a site's assets with a single tab. The Assets tab allows you to group and view the parts of a site for easy digestion. View all the Flash movies, all the images. See all the links, even all the colors used in the site.

Templates Are Fine, Until Someone Gets Hurt

Templates aren't exactly new to Dreamweaver. They're much more pow-

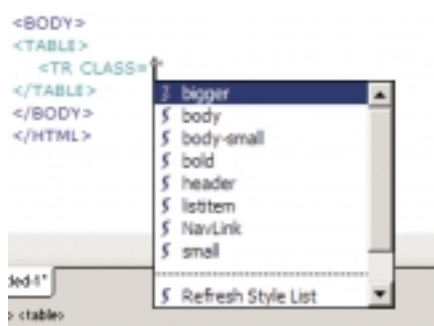


Figure 2: Style sheet elements in context dramatically simplifies development of sites with a large number of styles.

erful though. Dreamweaver MX brings the use of templates to a whole new level.

It would not be an exaggeration to say that templates now have their own language within Dreamweaver and that a developer could create some truly powerful code. Template script allows the creator of a template to make much more than a simple page with a big blank spot in the middle where text is copied into. Templates can be incredibly dynamic – so that Sally can actually put JavaScript in the top of her pages and has much more control over the page than

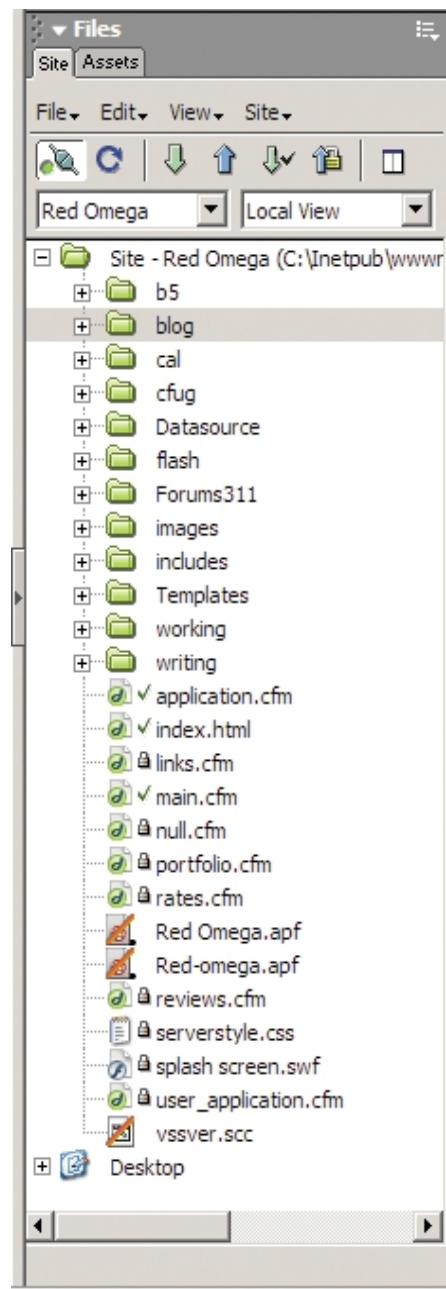


Figure 3: The Site pane shows not just the files that Dreamweaver can open but the files that Studio MX applications can open. You can also cloak certain extensions to keep them out of the way. Those with the red slash are cloaked.

Mark, who has very little control and is limited to putting text in specific areas (see Figure 4).

The use of template parameters allows developers the power to not have to hard-code image paths and links in the master template file. By defining these parameters as constants at the top of the page, subsequent changes won't need to be

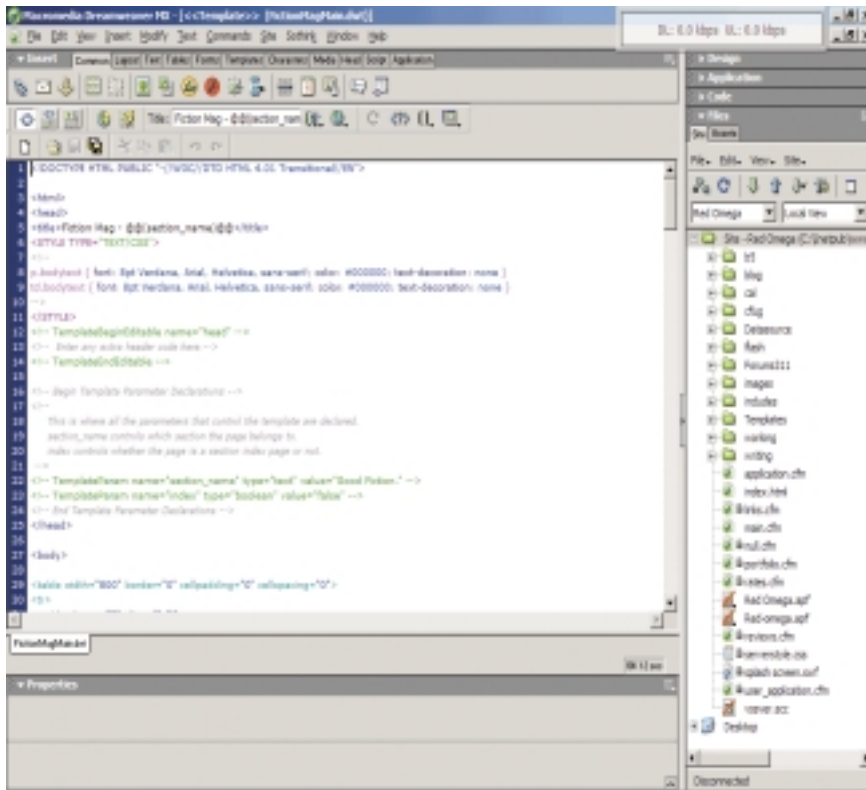


Figure 4: An example of a template written for a small Web site. You can see that there is quite a bit you can write in template script.

hunted down throughout the page, not to mention the rest of the site. All 14 (for example) links to the Clear shim graphic file can be updated by changing one line of code. This change can then be effected site wide by saving the changes to the master template.

I have come across only two drawbacks to using templates. One is in more complex sites where, say your <HTML> and <HEAD> tags are in a module and your closing </HTML> tag is in yet another module. Trying to save a template in this condition will result in an error. The code a template comes from must be well formed and intact within the single file.

The other drawback is that making changes to a template can be problematic in an enterprise environment. If a template itself needs to be changed – let's say I'm adding a new menu item to the navigation – every page based on the template has to be checked out of the code repository in order for the changes to propagate throughout the site.

The assumption here is that you're using some form of code repository; like Visual Source Safe, etc. For small operations and projects this is a less important issue. For the sites I maintain I'm the only developer, so the code is usually all checked out by me anyway and there's never anyone else working on it. For environments where a team is working on a site, one person making changes like that can be problematic if the others are not ready to check their changes back in, in order to let the template changer do his thing.

Probably the most natural use of templates is in corporate intranet environments, where you likely have many content contributors of various skill levels, and many departments with varying rights. Marketing may need rights to more pieces of an intranet, things like footers, headers, content that appears in more than one place, etc., while other departments like HR, may only need to update the static content within the benefits page.


Templates aren't a replacement or even a very good alternative to a true content management solution, but in smaller projects and teams they may be just what you need for a fraction of the cost.

More Than a Single Product, a Suite

Dreamweaver on its own is already an outstanding application. Studio MX is a suite for a reason. Dreamweaver, Flash, and Fireworks work hand in hand to help you create complete Web applications. Dreamweaver doesn't author Flash movies, Flash MX does. Dreamweaver isn't a graphics tool, Fireworks is. Prior to Studio MX, when you wanted to add a graphic to your code you had to separately open Fireworks (or another tool, but we'll use Fireworks) and create or edit the image, save it, and then put it into your code.

Dreamweaver allows you to place an image placeholder in your code that will let you pop right into Fireworks, do what you have to, and pop back into Dreamweaver, a much more fluid interaction. The same type of action is available with Flash. This functionality goes both ways. From Fireworks you can save a sliced graphic right into Dreamweaver.

The Dreamweaver Era

As we move forward and Macromedia continually improves upon Dreamweaver MX, the potential it brings for bridging the designer and developer worlds dramatically increases. If an entire creative team is able to use the same tool in very different ways – coding the back-end language, designing the site's look and feel, integrating Flash components, tying it all together – the possibilities become endless for the things that single team can accomplish. 

About the Author

John Wilker works for Sequent Technologies, a small startup company in Walnut, CA, as the lead Web developer/tech writer. John has been working with ColdFusion since 1997.

john@red-omega.com

IVIS TECHNOLOGIES

www.ivis.com

The Best Recipe

Cooking is a lot like programming

About two years ago, I decided that it would be “fun” to learn to cook. I figured I would be a quick learner; after all, I liked to eat (passion for the subject), I had been a skilled cabinetmaker (possessed manual skills), and I enjoyed watching “Iron Chef” on the Food Network (had an available learning resource). Compared to cabinetmaking and software development, how hard could it be?

When I told my wife, she adopted a Mona Lisa-type smile and told me what a good idea she thought it was. She was remarkably encouraging. So, I started on my journey to becoming a chef (the simple title of “cook” not seeming to properly express my culinary ambitions).

Having seen that the TV chefs all had really great cookware, knives, and gadgets, I went and did likewise – and during the time that many of the dot-coms were failing for lack of revenue, I single-handedly kept cooking.com in business. After watching this process for some time, my wife asked me when I was going to actually start cooking. I tried explaining: “Mastery of any new field of endeavor must have foundations and I am building those foundations now.”

Susie countered: “Really? Well, all I know is that you can’t move around in the kitchen without falling over Calphalon pots and pans, Sabatier knives, Cuisinart food processors, KitchenAid mixers – and you still haven’t made a meal!”

Well, that was easily enough remedied. When my wife announced that her mother would be coming over for dinner on Saturday night, I announced that our menu for the evening would include Beef Tenderloin with Red-Wine and Marrow Sauce, Whole Stuffed Artichokes Braised



By Hal Helms

in White Wine, Potato Nests with Sautéed Shitake Mushrooms, and Carrots Glazed with Balsamic Vinegar and Butter. (Having in the past learned something about lowering expectations, I didn’t let on that I had plans for whipping up a homemade Lemon Sorbet with Blackberry Sauce for dessert.)

It all started well enough – we have a great farmer’s market in Atlanta, and shopping there was actually fun. I came home with rather more than I intended, but no matter – my life as a chef had begun!

I knew that the TV chefs didn’t use cookbooks, but thought that for my first couple of soirees, I might be forgiven this indiscretion. I now began perusing cookbooks and resources in earnest. Ah! Here was something interesting: apparently, the secret to great tenderloin was cooking it at 200 degrees – a temperature much lower than most of the recipes had suggested. And here was something else: the really good balsamic vinegar was aged for as much as 100 years – and had the prices to prove it: a 4.5 oz bottle cost over \$200! I hoped that my more frugal selection (only \$22 for the same-sized bottle) wouldn’t doom my efforts.

Saturday finally came – of that I’m sure. What exactly happened on

Saturday I’m far less certain of – a phenomenon psychologists call blocking, in which a person deletes memories of events too painful to recall. I do remember certain images, like mental snapshots: the family seated at the table resplendent with fine china, crystal – and a completely raw roast (my indispensable instant-on digital thermometer barely registering 90 degrees).

I recall my mother-in-law’s look when she discovered that I hadn’t removed the inedible parts of the artichoke (who knew you couldn’t eat the whole thing?) I remember my wife gamely trying to eat the sodden mass of indistinguishable ingredients that were on her plate. And I can still see my son staring in horror and fascination at his purple carrots. We never made it to the sorbet. We abandoned the “meal” and made a quick trip for hamburgers.

Well, since then, I’ve learned some lessons along my torturous path to being able to produce decent meals and over those two years, I’ve continually seen the parallels between lessons in cooking and lessons in software development.

What I’ve Learned About Cooking – and Programming

- **Cooking is hard – really hard.** TV chefs make it look easy; don’t believe it. If you’re going to master something as complex as cooking, it won’t happen quickly or painlessly. You’re going to have some spectacular failures, a good deal more mediocre results, and maybe, occasionally, a real success.
 - *Parallel:* Programming is hard, too. While ColdFusion makes it easier to get started in coding, if you’re going to excel at developing applications, you’re going to have to go beyond the basics – and for that, you’re going to have to learn and apply good software development practices, things

MACROMEDIA

www.macromedia.com/go/certification/

that transcend the specific language.

- **When you do your job well, you've simply met people's expectations. When you fail, you eat out.**
– *Parallel:* When you've written great code, your boss probably won't call a general company meeting to praise you in front of your peers. That sort of thing only happens in movies. A better strategy is to excel for the simple joy that comes from excellence. Of course, as Louis Pasteur once observed, chance favors the prepared mind. We can't do much about chance; we can do a great deal about preparing our minds.
- **Success is all about preparation.** This is what the French call *mise-en-place*, a term referring to having all the ingredients necessary and ready to combine at the moment of cooking. If you start on a dish without proper preparation, you'll quickly find that your carrots will burn while you're struggling to clarify the butter. *Note:* TV chefs have a small army of assistants who do their preparation for them.
– *Parallel:* Don't be so quick to start writing code. Take the time to thoroughly think through the job you're going to do. Do you really understand what the user is doing? Have you thought about the fact that the requirements that the user has assured you are set in stone are, in fact, absolutely bound to occur? Have you investigated buying third-party software for parts of your application? What will integration be like? Are you using a framework? Are you rigorous about creating tests before you code? Are you making use of design patterns? All this requires discipline up front, but it pays great dividends when the unexpected begins to occur.
- **Good tools make a big difference.** They're not magic and they don't guarantee success, but when you're ready to use them, good tools will support you. The last thing a novice needs is tools that betray them.
– *Parallel:* If a chef is concerned about things like pans, pots, and knives, we need to be concerned about XML, SQL, Java, etc. Mastery comes from knowing which tool to use in which case. There's usually more than one right answer for this and part of learning about your tools is learning the

ones that you prefer – those that “fit your hand” best, so to speak.

- **There's simply no substitute for learning from another, in person.** While books are great, they can't give you the actual experience of doing it, and of seeing how an experienced professional thinks and plans. Have you ever wondered how to tell when a piece of meat is cooked, and to what degree: rare, medium, or well? There's a great technique involving your fingers and comparing the firmness of the muscle between your thumb and forefinger with the meat you're cooking. Unfortunately, I can't really tell you, I need to show you!
– *Parallel:* While books can help you, there are some extremely important aspects of software development that simply don't translate to the written word, so take every opportunity to work with others from whom you can learn. Also, don't overlook the value of good instructor-based training; you can jumpstart your skills in the right class.
- **Cooking is about coordination as much as tools or technique.** You're not cooking a single dish, but putting together a meal. If the potatoes are done while the roast is still bleeding, you have a failure, no matter how good the potatoes are.
– *Parallel:* Ultimately, our ability to succeed doesn't rely on being able to masterfully manipulate a three-dimensional array or to handle matrix math, but to produce a complete software application. If you do a great job writing XPath queries, but the software is late, buggy, or doesn't integrate with other systems, you have a failure on your hands no matter how good the XPath queries are.
- **Simplicity rules.** In many areas of craft and art, complexity is a killer. Now this doesn't mean that the end result should be “simple” (e.g., hamburger with fries), but that when creating dishes that are involved, the entire operation should be broken down into a series of simple steps that can be handled easily.
– *Parallel:* Much of good software craftsmanship practices are related to keeping things simple. This is the key to encapsulation – being able to concentrate on only one thing at a time

and, when done, forget about the implementation and concern yourself only with the interface.

- **Knowing how something is done is not the same as being able to do it.** A logician might phrase it thus: knowledge of the process is necessary but not sufficient. What could be easier to describe – or harder to master – than the making of a truly great pie crust or an omelet? There is simply no substitute for practice. And practice usually entails a high risk of failure until the process is mastered. The purpose of practice is to transfer the intellectual knowledge into your fingertips, your nose, your taste so that you are so intimately connected with your materials that you do the right things naturally. Practice makes what is at first unnatural, natural.
– *Parallel:* We might call this the Spectator Syndrome, in which people convince themselves that reading about something, or hearing a talk on it, is the same as having actual application experience. One large problem with this syndrome is that it makes us prey to the hype from vendors, who can slant the talk or presentation to present their product or technology in the best possible light. But we all know that no product or technology is perfect, and if our first real exposure to it is on a job where success is critical, we're likely to lead lives of high excitement in areas that we wish to be highly predictable.
- **Dull knives are dangerous.** Dull knives require a much greater degree of force, and with that force comes the possibility of the knife slipping and doing damage to either the cook, the food, or both. It's not glamorous work, but learning to properly sharpen knives is very important for success.
– *Parallel:* Have you customized your editor so that you're using shortcuts, code templates, snippets, etc? If not, you're using a dull knife. One feature of ColdFusion Studio that I rely on is the ability to create a code template, to provide a keyboard shortcut that will automate the typing of common code. This has a huge impact on productivity and is very simple to set up. The principle is that for the tools we use on a daily basis, we should learn enough to sharpen them so that they serve our purpose more exactly.

- **Don't try new stuff on company, but do try it on family.** What can be more discouraging, as well as embarrassing, as trying out a new dish that flops in front of company? My mother-in-law could not have been more gracious about the beef tenderloin fiasco, but I still wish I had tried it out before she came over. On the other hand, if you don't experiment, you'll become boring, so try new stuff out on your family. But keep the number of a good pizza delivery place nearby for those times that your experiments are less than successful.
- *Parallel:* Don't volunteer to base a mission-critical application on untried technology ("Gee, how hard can CORBA be, after all?") or you'll set yourself up for failure. But, too, make use of your projects to go to school on. Pick some aspect of the project that is not mission critical, particularly a small part of the product, and try out new technologies on that. If you fail, you can switch back to a proven technology and you'll have gained valuable experience. If you succeed,


you've just added another tool you can use on mission-critical products.

I could regale you with tales from two years of trial and error. During the first year, it was mainly error – such as the time that I decided to cook Thanksgiving dinner. You know, when they tell you, "Put the turkey in the oven," but they don't tell you which side goes up; you're just expected to know that. Well, now I know: it's the opposite side I put up on my first Thanksgiving attempt.

Nor can I forget the first time I tried to make choux pastry. The results were so bad that we had to air out the house. Then, there was the leek soup I made without first completely peeling the leeks. And the three pounds of Chilean sea bass (at \$15.00/lb) that went into the trash. Or the Christmas dinner of Sole Florentine that went the same way as the sea bass. But you get the point.

I'd like to assure you that after some initial "false starts," things went smoothly and today, I'm a first-rate chef. I'd even be willing to settle for saying that after some very unsuccessful attempts, I

mastered the art and craft of cooking and that I'm a darned good cook. But I can't honestly say either of those things.

I can say that after two years of practice – sometimes failing, sometimes succeeding – that I still love cooking and that I'm getting better at it every day. I'm getting to enjoy the successes while not being too tough on myself when things fail. And I think that's not such a bad place to be – in both cooking and programming. Because ultimately, it's not about the ability to impress others or get paid lots of money (not that I object!); it's about taking pride in your work and having the humility to acknowledge the difficulties of attaining true excellence while still persevering. That, it seems to me, is the best recipe for real success. 

About the Author

Hal Helms (www.halhelms.com) is a Team Macromedia member who provides both on-site and remote training in ColdFusion, Java, and Fusebox.

hal@fusebox.org

CFDYNAMICS

www.cfdynamics.com

Design Patterns in ColdFusion: Singleton Pattern

Ensure that a CFC has only one globally accessible instance

PART 5 OF A SERIES

With the introduction of ColdFusion Components, or CFCs, in ColdFusion MX, we as ColdFusion developers have a way to leverage object-oriented programming, or OOP, within CFML.

Sometimes it is necessary in object-oriented programming to ensure that there is always one, but never more than one, of a certain object. There is also the need to access that single object from within any application. This idea of a single, globally accessible object has numerous programmatic advantages for us as developers.

It also makes the underlying J2EE memory management and garbage collection on which ColdFusion MX is built more efficient. And while CFCs are not the pure objects we may be familiar with in Java, they are an extremely useful object-like construct that we can utilize in several ways.

We can use CFCs as a static class library of related methods. We can use CFCs as a way to leverage Web services within ColdFusion, or we can even create a CFC object, or more precisely an "instance" in object-oriented terminology. We can then treat our CFC instance almost identically to an object in a language like Java. When working with CFCs we can create a single instance fairly easily. We can ensure that this one instance is accessible wherever necessary. In our example we will even make our CFC instance persist across multiple hits to the server. This is going to be fun!



By Brendan O'Hara

Creational, Structural, and Behavioral

Last month we talked in some detail about the three types of design patterns: creational, structural, and behavioral. We also introduced our first structural pattern, the Composite pattern. The previous three articles in this series, Template Method, Iterator, and

Strategy, were all behavioral patterns. The third type of patterns, creational patterns, deals with object creation.

Object-oriented programming concepts are relatively new to ColdFusion developers, so object creation as a concept, is totally foreign to us. To that end, this month we introduce our first creational pattern, the Singleton, with what is hopefully an example that will make sense of object creation as well as of the pattern itself.

There Can Be Only One

As with each previous article in this series I like to refer to the original "intent" of the pattern as laid out in the book *Design Patterns: Elements of Reusable Object-Oriented Software*. Clearly the pre-eminent work on the subject of design patterns, it is a must-read if you have an interest in mastering object-oriented pro-

gramming. Its authors, the so-called "Gang of Four," established the general "intent" of the Singleton pattern to be the following:

"Ensure a class only has one instance, and provide a global point of access to it."

Well, that's actually pretty self-explanatory. We want one and only one instance of our CFC object and we want a way to access that CFC object from wherever we need to. In fact the Singleton pattern as a whole is at first glance somewhat easier to comprehend than many other design patterns. However, it is no less useful or powerful.

In a stateless Web environment, object creation is rarely an issue we need to deal with because every object that is "instantiated" is done behind the scenes and disappears again without us being any the wiser. We are not only going to create a Singleton CFC instance, we are also going to persist that instance across many pages and across many users.

Managing Applications in ColdFusion

ColdFusion developers have always controlled applications and application-specific data from within either a central Application.cfm file under the webroot (such as /inetpub/wwwroot/ on IIS) or within app-specific Application.cfm files within each application's directory. Inside these files we put everything from login and permission checks to setting application or request variables with static data that our application will need to run.

We also include a <CFAPPLICATION> tag, which allows us to define the scope of an application, allow session or client variables, and set application or session timeouts. We may go even further by having the app-specific Application.cfm files <CFINCLUDE> the webroot's Application.cfm file, which may set application or request variables for global data or data used by two or more applications. This is quite useful as it allows us to separate out common data, as opposed to duplicating the code, so we don't have a maintenance nightmare if any of that data happens to change. Another useful tool in working with application-specific data is XML configuration files.

XML Configuration Files

XML is everywhere these days and is being touted as a cure for all of your programmatic ills. Recently, as ColdFusion MX has taken hold with its new XML parsing capabilities, I have seen many discussions and recommendations with regard to using XML configuration files to store application-specific and/or global data. The XML file is read and parsed for every page request that is loaded on the server. In fact XML is becoming the standard repository for storing static attributes in Web development with both Java and .NET using XML configuration files extensively. I am not a proponent of using XML to store this kind of data, but it is an interesting idea that does seem to be gaining ground in the ColdFusion community.

The application management example we will be going over implements the Singleton pattern to ensure that there is one and only one CFC "object" in memory. It could use one or more XML configuration files to import static data into the CFC when it is instantiated. This data will need to be referenced in the application, but in our example we are hard coding the data as "properties" of the CFC.

Application Manager Base Class

The CFC that will make this all work for us is ApplicationMgr.cfc, shown in Listing 1. The CFC has two private structures, one called "App" and one called "My". The properties set into the App structure are the default settings for the <CFAPPLICATION> tag. These properties can be overridden in any application-specific CFC that extends ApplicationMgr.cfc. The execute() method outputs the <CFAPPLICATION> tag and any additional processing that you would normally include in Application.cfm. Alternatively, this additional processing, such as login or permission checks, should probably be placed in other functions within the CFC. This allows granular control over each method that can then be overridden in any future "derived" CFCs.

The properties set into the My scope effectively take the place of request or application scope variables. They are not put into the "this" scope so they are actually private variables that cannot be written over but can be accessed using the getValue() method passing in a single argument, which is the string name of the particular variable being accessed. An example is shown below:

```
<CFOUTPUT>#myApp.getValue("DSN")#</CFOUTPUT>
```

Last but not least we need the Singleton pattern to ensure that a maximum of one instance is created and that all calls to the ApplicationMgr.cfc return that same instance. To ensure you have the necessary control over the instantiation process it is normal to make the objects constructor private in Java or similar

languages. Since CFCs do not have a classic constructor in the same way Java does we simply need to create an accessor method. An accessor method is a static method, in this case getInstance(), which will check to see if the global Singleton instance already exists and will instantiate it if it doesn't. Regardless, it then returns the reference to the CFC object to the calling page.

In this example we are storing our Singleton ApplicationMgr.cfc instance in the seldom-used Server scope, which allows the CFC instance to persist in memory on the server until ColdFusion Application Server is stopped or the actual server is rebooted. This allows us access to a persistent object from within our Web pages, which in and of itself is something of a rarity.

Now one thing to notice is that we first check to see if the server variable in which we are storing our Singleton object already exists and that it passes the IsStruct() test, which all CFC instances will. Then we put an exclusive lock on the variable using the <CFLOCK> tag. This ensures that once the first person instantiates the CFC, everyone else will get that same instance as well. We then check for existence again because someone may have been creating it as we were reading it initially.

Now that we have an exclusive lock we can know definitively whether or not it exists or whether it needs to be created. This type of locking is something common to multithreaded applications. In fact, the alternate name for multithreaded Singleton implementations is the Double Checked Locking Pattern.

Although the accessor method can create the Singleton, most of the time it will already exist in the Server scope when the acces-

SOUTHWEST SUPPORT SOLUTIONS

sales@swsupportsolutions.com

sor method is called. If some form of runtime information is needed, said processing is normally done inside the accessor method.

BillingAppMgr.cfc and QuoteAppMgr.cfc

As we examine the code for BillingAppMgr.cfc we notice that it doesn't actually do that much. We do override the applications name variable, which we store in the "App" scope. In addition we have added a DSN and ReportServer variable to the My scope so we can access them from within our application. Additionally BillingAppMgr.cfc overrides the getInstance() method because it needs to look for its application-specific server scope variable that will store its CFC instance. Code for BillingAppMgr.cfc is shown below and in Listing 2.

```
<cfcomponent extends="com.mycompany.ApplicationMgr"
displayname="BillingAppMgr">
<!-- App.* Stores Private Application Data
    Accessible only within CFC -->
<cfparam name="App" type="struct" default="#structnew()#">
<cfset App.name = "Billing"><!-- "application_name" --->
<!-- My.* Stores Public Application Data
    Accessible via GetValue("VarName") --->
<cfparam name="My" type="struct" default="#structnew()#">
<cfset My.DSN = "Billing"><!-- datasource name --->
<cfset My.ReportServer = "Crystal"><!-- Report Server --->
<cffunction name="getInstance" access="public"
    returntype="struct" output="no">
    <cfif NOT IsDefined("Server.ao_BillingAppMgr_AppObj") OR
        NOT IsStruct(Server.ao_BillingAppMgr_AppObj)>
        <cflock timeout="10" throwontimeout="No"
            type="EXCLUSIVE" scope="SERVER">
            <cfif NOT IsDefined("Server.ao_BillingAppMgr_AppObj") OR
                NOT IsStruct(Server.ao_BillingAppMgr_AppObj)>

                <cfset Server.ao_BillingAppMgr_AppObj = this>
            </cfif>
        </cflock>
    </cfif>
    <cfreturn Server.ao_BillingAppMgr_AppObj>
</cffunction>
</cfcomponent>
```

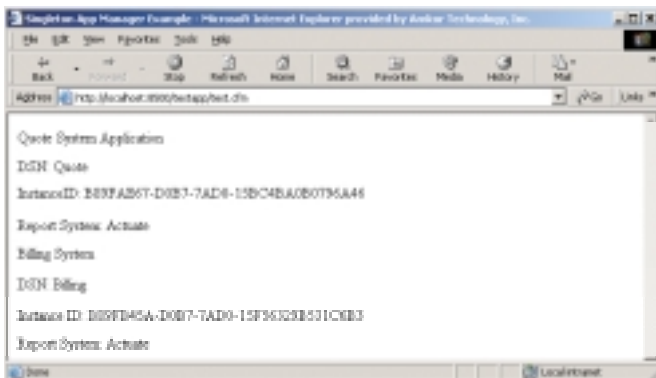


Figure 1: Our Singleton Application Manager test page

Now the Quote Application Manager works identically to the Billing Application Manager except with different App.Name, My.DSN, and My.ReportServer variables. Of course the InstanceID will also be different since they are different instances. The QuoteAppMgr.cfc is shown in Listing 3.

Application.cfm

Now with our application manager CFCs in hand what goes in Application.cfm? Well let's look at the billing system's Application.cfm because the billing system needs to access both billing and quote application-specific data. Application.cfm is shown below and in Listing 4:

```
<cfset Billing =
CreateObject("component", "com.mycompany.Billing.BillingAppMgr")>
<cfset Quote =
CreateObject("component", "com.mycompany.Quote.QuoteAppMgr")>
<cfset Billing = Billing.getInstance()>
<cfset Quote = Quote.getInstance()>
<cfset Billing.Execute()>
```

Well at five lines that's a pretty short Application.cfm. The first two lines create references to the BillingAppMgr and QuoteAppMgr CFC objects respectively. Lines 3 and 4 call the getInstance() methods for each of these returning the current Singleton objects regardless of whether they are newly created or have been sitting in server memory for weeks. The last line calls the execute method for Billing Application Manager because that's the actual application we are currently in.

The execute() method outputs the <CFAPPLICATION> tag, and any additional processing you may have normally added to Application.cfm. Now we can get application-specific data anywhere within our billing application, and even more exciting, to me at least, is the idea that we could add application-specific methods to our application manager CFCs. These methods need not be accessed only within Application.cfm; they can in fact be accessed anywhere within the application.

Testing the Singleton

The Singleton Application Manager example will output the InstanceID, DSN, and ReportSystem for each application. The code is shown below and in Listing 5:

```
<html>
<head>
    <title>Singleton App Manager Example</title>
</head>
<body><br>Quote Application<br><br><br>
<cfoutput>DSN:&nbsp;#Quote.getValue("dsn")#</cfoutput>
<br><br><br><cfoutput>InstanceID:&nbsp;#Quote.getValue("InstanceID")#</
cfoutput>
<br><br><br><cfoutput>Report
System:&nbsp;#Quote.getValue("ReportSystem")#</cfoutput>
<br><br><br>Billing Application
<br><br><br><cfoutput>DSN:&nbsp;#Billing.getValue("dsn")#</cfoutput>
<br><br><br>
<cfoutput>Instance ID:&nbsp;#Billing.getValue("InstanceID")#</cfoutput>
<br><br><br>
```

```
<cfoutput>Report System:&nbsp;&#Quote.getValue("ReportSystem")#</cfout-  
put>  
</body>  
</html>
```

The InstanceID is useful for testing purposes to ensure that no matter how many page hits or sessions you open, you will always be looking at the same CFC instance. The InstanceID is simply a UUID created on instantiation. The output of the example is shown in Figure 1.

Runtime Information


If you need any runtime information, say the number of current users accessing the application, it can be stored as a variable and that variable can be accessed at any time or it can be determined programmatically on the fly when the getInstance() method is called.

A company I have consulted for uses a product called TeamStudio Screensurfer, which they use to access so-called "green screen" applications from their AS400 "mainframe." In May 2003, *CFDJ* ran a detailed review of the product ("Teamstudio Screensurfer: Bring screen-based applications to the browser in HTML format quickly and painlessly," Vol. 5, issue 5). This product can be used programmatically (explained in the review) or as a Web emulator.

When used as an emulator you need to maintain a constant connection from your client to the server. So if you have a five concurrent connection license it will only allow, you guessed it, five concurrent connections. You may want your application

object to keep track of who has a session open and how long they have been working in it. For as long as our application object persists on the server, it can collect information, which is then accessible to us programmatically. We may want to give a "friendly" message to a prospective TeamStudio Screensurfer user saying that all five connections are being used and maybe list the current users and how long their current sessions have been active.

Conclusion

Whether you are maintaining a simple Web site or a complex n-tier application, you may come to the conclusion that a single "instance," no more and no less, of a certain CFC object is required. As the much-repeated quote from one of my favorite movies, the original "Highlander," (not those awful sequels mind you) puts so succinctly: "There can be only one." When dealing with the Singleton pattern there can be only one and that's exactly the way we want it. 

About the Author

Brendan O'Hara is one of the coauthors of Advanced Macromedia ColdFusion MX Application Development, published by Macromedia Press. Brendan has a Macromedia ColdFusion MX Developer Certification along with Java and Linux certifications from Penn State University. Brendan has just been named to Team Macromedia for ColdFusion. He is a ColdFusion, Java, and .NET software architect in the Philadelphia suburbs.
brendantohara@yahoo.com

CRYSTALTECH

www.crystaltech.com

Listing 1: ApplicationMgr.cfc

```
<cfcomponent displayname="ApplicationMgr">
<!---
App.* Stores Private Application Data
Accesible only within CFC
--->
<cfparam name="App" type="struct" default="#structnew()#">
<cfset App.name = "AbstractApp">
<cfset App.clientManagement = "Yes">
<cfset App.clientStorage = "registry">
<cfset App.setClientCookies = "Yes">
<cfset App.sessionManagement = "No">
<cfset App.sessionTimeout = CreateTimeSpan(0, 2, 0, 0)>
<cfset App.ApplicationTimeout = CreateTimeSpan(0, 2, 0, 0)>
<cfset App.setDomainCookies = "Yes">
<!---
My.* Stores Public Application Data
Accesible via GetValue("VarName")
--->
<cfparam name="My" type="struct" default="#structnew()#">
<cfset My.InstanceID = createUUID(>

<cffunction name="getInstance" access="public" returntype="struct" out-
put="no">
    <cfif NOT IsDefined("Server.ao__ApplicationMgr_AppObj") OR
        NOT IsStruct(Server.ao__ApplicationMgr_AppObj)>
        <cflock timeout="10" throwontimeout="No"
            type="EXCLUSIVE" scope="SERVER">
            <cfif NOT IsDefined("Server.ao__ApplicationMgr_AppObj") OR
                NOT
IsStruct(Server.ao__ApplicationMgr_AppObj)>
                <cfset Server.ao__ApplicationMgr_AppObj =
this>

                </cfif>
            </cflock>
        </cfif>
        <cfreturn Server.ao__ApplicationMgr_AppObj>
    </cffunction>

<cffunction name="Execute" access="public" output="Yes">
    <cfapplication
        name="#App.name#"
        clientManagement="#App.clientManagement#"
        clientStorage="#App.clientStorage#"
        setClientCookies="#App.setClientCookies#"
        sessionManagement="#App.sessionManagement#"
        sessionTimeout="#App.sessionTimeout#"
        applicationTimeout="#App.applicationTimeout#"
        setDomainCookies="#App.setDomainCookies#"
    </cfapplication>

<!--- GET THE VALUE OF A PROPERTY --->
<cffunction name="getValue" access="public" output="false">
    <cfargument name="name" required="Yes" type="string">
```

```
        <cfif IsDefined("My.#arguments.name#")>
            <cfreturn My[arguments.name]>
        </cfif>
    </cffunction>

</cfcomponent>
```

Listing 2: BillingAppMgr.cfc

```
<cfcomponent extends="com.amkor.ApplicationMgr"
displayname="BillingAppMgr">
<!---
App.* Stores Private Application Data
Accesible only within CFC
--->
<cfparam name="App" type="struct" default="#structnew()#">
<cfset App.name = "Billing"><!--- "application_name" --->
<!---
My.* Stores Public Application Data
Accesible via GetValue("VarName")
--->
<cfparam name="My" type="struct" default="#structnew()#">
<cfset My.DSN = "Billing"><!--- datasource name --->
<cfset My.ReportSystem = "Crystal"><!--- Report Server --->
<cffunction name="getInstance" access="public" returntype="struct" out-
put="no">
    <cfif NOT IsDefined("Server.ao__BillingAppMgr_AppObj") OR
        NOT IsStruct(Server.ao__BillingAppMgr_AppObj)>
        <cflock timeout="10" throwontimeout="No"
            type="EXCLUSIVE" scope="SERVER">
            <cfif NOT IsDefined("Server.ao__BillingAppMgr_AppObj") OR
                NOT
IsStruct(Server.ao__BillingAppMgr_AppObj)>
                <cfset Server.ao__BillingAppMgr_AppObj =
this>

                </cfif>
            </cflock>
        </cfif>
        <cfreturn Server.ao__BillingAppMgr_AppObj>
    </cffunction>

</cfcomponent>
```

Listing 3: QuoteAppMgr.cfc

```
<cfcomponent extends="com.amkor.ApplicationMgr" displayname="QuoteAppMgr">
<!---
App.* Stores Private Application Data
Accesible only within CFC
--->
<cfset App.name = "Quote"><!--- "application_name" --->
<!---
My.* Stores Private Application Data
Accesible via GetValue("VarName")
--->
<cfset My.DSN = "Quote"><!--- datasource name --->
```

```

<cfset My.ReportSystem = "Actuate"><!-- Report Server -->
<cffunction name="getInstance" access="public" returntype="struct" output="no">
    <cfif NOT IsDefined("Server.ao__QuoteAppMgr_AppObj") OR
        NOT IsStruct(Server.ao__QuoteAppMgr_AppObj)>
        <cflock timeout="10" throwontimeout="No"
            type="EXCLUSIVE" scope="SERVER">
            <cfif NOT IsDefined("Server.ao__QuoteAppMgr_AppObj") OR
                NOT
IsStruct(Server.ao__QuoteAppMgr_AppObj)>
                <cfset Server.ao__QuoteAppMgr_AppObj =
this>

                </cfif>
            </cflock>
        </cfif>
        <cfreturn Server.ao__QuoteAppMgr_AppObj>
    </cffunction>
</cfcomponent>

```

Listing 4: Application.cfm

```

<cfset Quote =
CreateObject("component", "com.mycompany.Quote.QuoteAppMgr")>
<cfset Quote = Quote.getInstance()>
<cfset Billing =
CreateObject("component", "com.mycompany.Billing.BillingAppMgr")>

```

```

<cfset Billing = Billing.getInstance()>
<cfset Billing.Execute()>

```

Listing 5: test.cfm

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>Singleton App Manager Example</title>
</head>
<body><br>Quote System Application<br><br><br>
<cfoutput>DSN:&nbsp;#Quote.getValue("dsn")#</cfoutput>
<br><br><br>
<cfoutput>InstanceID:&nbsp;#Quote.getValue("InstanceID")#</cfoutput>
<br><br><br>
<cfoutput>Report System:&nbsp;#Quote.getValue("ReportSystem")#</cfoutput>
<br><br><br>Billing
System<br><br><br><cfoutput>DSN:&nbsp;#Billing.getValue("dsn")#</cfoutput>
<br><br><br>
<cfoutput>Instance ID:&nbsp;#Billing.getValue("InstanceID")#</cfoutput>
<br><br><br>
<cfoutput>Report System:&nbsp;#Quote.getValue("ReportSystem")#</cfoutput>
</body>
</html>

```

Download the Code...
Go to www.coldfusionjournal.com

PAPERTHIN

www.paperthin.com

Defeating DoS Attacks

Protect your apps from denial of service attacks

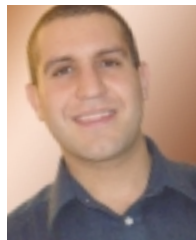
A denial-of-service (DoS) attack is an attempt by a single person or a group of people to disrupt an online service. It is designed to bring the server and network to its knees by flooding it with useless traffic. A DoS attack is the most common type of Internet attack and can be launched against your site at any time with relative ease. This can affect you by eating up your bandwidth and server resources to service these bogus requests while leaving no resources to fill legitimate requests.

So how do you know if you're being attacked? There are many variations of DoS attacks, and some are more difficult to catch than others. Some programs, such as the Apache Web server, have rearchitected their software to protect against this sort of attack. The method described here will allow you to catch and automatically defeat DoS attacks against your applications, regardless of the Web server used.

All you'll need to add to your database is a single table to keep track of active sessions (see Listing 1). You can decide whether to implement this on an application level or across the entire server, and decide which database makes sense for doing this tracking. Although there is a slight performance hit to enable this feature, you'll gain other relevant information on current site usage such as how many users are online, what page they're currently viewing, and how many total pages they've retrieved during the session.

Some may feel that we're creating a set of data for tracking sessions in a database that could just as well be created using ColdFusion's client variables. We'll discuss that alternative briefly in the conclusion.

We only need to make a couple of changes to our application to enable our DoS checking. First, in your



By Joe Danziger

application.cfm, we'll add code to check if a session already exists in our database. If it does, we'll update the lasthit timestamp as well as the last page and total pages viewed; otherwise we'll add a new record for this session (see Listing 2). We use a combination of CFID and CFTOKEN instead of URLTOKEN to keep things compatible with earlier versions of ColdFusion.

Now we have all of the current sessions for our site within a database. We'll take advantage of ColdFusion's built-in session management to help us detect attacks. Normally, ColdFusion will generate a new

CFID and CFTOKEN for each individual browser that it encounters. Since many DoS attacks use an automated program and not a browser, ColdFusion ends up generating a new CFID and CFTOKEN for each request. Now we need a way to check for this.

We set up a scheduled script to run every 5 minutes and monitor our session table (you can modify the interval if necessary). Our check script (see Listing 3) first deletes any sessions from our database that are older than our specified session length (as defined in a CFAPPLICATION tag, which though not shown here, is 60 minutes in this case). We then count how many distinct URLTOKENs we have for each IP address. You should note that it is perfectly valid to have multiple sessions for a single IP if your users are sharing an Internet connection or going through a proxy server. Therefore, we'll set our threshold high enough (50) so that regular usage of the site will not be affected. You can modify this threshold as you see fit to make the check more or less sensitive. Finally, we create a list of any offending IP addresses and save this to an application variable.

You may wonder about locking the last line, assigning the application.DoS variable. While in CF5 it's always best to lock any reference to persisted variables, and in CFMX it's wise to do so to avoid "race conditions" when multiple users or sessions run code at the same time. In this case, however, this is a task that will be run only by the CF scheduler and then only once at a time.

The last thing you'll need to do in order to protect your site is to add some code at the top of your application.cfm to stop offending requests dead in their tracks (see Listing 4). Although we can't ignore these requests completely (as they've already made it through to our application), we can halt them right away so that we do not waste processing power or bandwidth.

Listing 1: Create database table

```
CREATE TABLE sessions (
  id          int IDENTITY (1,1) NOT NULL,
  urltoken    varchar(12),
  ipaddress   varchar(15),
  template    varchar(150),
  firsthit    datetime,
  lasthit     datetime,
  pages       int
)

PRIMARY: id
INDEXES: urltoken, lasthit
```


Listing 2: Tracking our sessions

```
<!-- check for current cfid and cftoken in query -->
<cfquery name="exists" datasource="dos">
    SELECT id FROM sessions WHERE urltoken = '#cfid#cftoken#'
</cfquery>
<!-- if exists, update, otherwise add new -->
<cfif exists.recordCount gt 0>
    <cfquery name="updateHit" datasource="dos">
        UPDATE sessions SET lasthit = #Now()#,
            template = '#cgi.script_name#',
            pages = pages + 1
        WHERE id = #exists.id#
    </cfquery>
<cfelse>
    <cfquery name="insertHit" datasource="dos">
        INSERT INTO sessions (urltoken,ipaddress,template,firsthit,lasthit,pages)
        VALUES ('#cfid#cftoken#','#cgi.remote_addr#','#cgi.script_name#',
            #Now()#, #Now()#,1)
    </cfquery>
</cfif>
```

Listing 3: checkDoS.cfm scheduled script

```
<!-- delete old records -->
<cfquery name="getOld" datasource="dos">
    DELETE FROM sessions
    WHERE lasthit < #CreateODBCDateTime(DateAdd("n",-60,Now()))#
</cfquery>

<!-- check for attacking IPs -->
<cfquery name="check" datasource="dos">
    SELECT ipaddress, count(urltoken) as hitCount FROM sessions
    GROUP BY ipaddress
    HAVING count(urltoken) > 50
    ORDER BY count(urltoken) DESC
</cfquery>

<!-- set bad ip list -->
<cfset application.DoS = ValueList(check.ipaddress)>
```

Listing 4: Halting invalid sessions

```
<cfif ListFind(application.DoS,cgi.remote_addr)>
    <h2>DoS Attack Suspected!!!</h2>
</cfif>
```

We mentioned earlier that this process of tracking and analyzing sessions, while it has value beyond just this task, is partly already done for us if we enable CLIENTMANAGEMENT="yes" in our CFAPPLICATION tag. ColdFusion will then automatically write to a repository a set of fixed client variables similar to those we've set up (CFID, CFTOKEN, HitCount, LastVisit, TimeCreated, and URLToken). It will then update these values for each client that visits your application.


There are a couple of problems with using this data. First, CF doesn't automatically write the IP address for the visitor. You could easily add that one value by adding to your application.cfm the single line:

```
<CFSET client.ipaddress=cgi.remote_addr>
```

But even then, in order to process the data, you need to manage it in the location and format in which CF creates it. It writes these

data either to a database or the registry, depending on the setting of the associated CLIENTSTORAGE attribute. If CLIENTSTORAGE="registry", or if it's not specified at all, CF will write these variables to the registry. CLIENTSTORAGE can also point to a database.

The problem is that the database approach creates tables (Cdata and CGlobal) and columns that are not easily parsed for our purposes. And while we could write CFREGISTRY tags to extract the data from the registry, it is challenging to then convert that to query-like data to be analyzed as we have. (If you want to explore it, the client data is written to the registry in HKEY_LOCAL_MACHINE\SOFTWARE\Macromedia\ColdFusion\CurrentVersion\Clients).

DoS attacks are on the rise and growing exponentially each year. You need to protect yourself, or at least be aware of the damage that a DoS attack can do. If you have a Web page with 200K of graphics and data, calling that six or seven times a second will consume a full T1 of bandwidth. Hopefully most of you will only need this tag to keep track of users currently on your site, but for those having attack problems, this method will provide a self-correcting way to insulate your site from trouble. 

About the Author

Joe Danziger is the founder and president of DJCentral.com, an online promotional tool for disc jockeys and other members of the electronic dance music industry. He has been developing professional ColdFusion solutions since version 1.5.

joe@djcentral.com

**PACIFIC
ONLINE**
www.pacificonline.com

When One ColdFusion Is Not Enough

One server, lots of ColdFusions

By now, every ColdFusion developer knows, or should know, that ColdFusion MX sits on top of underlying Java architecture. Some ColdFusion developers have even been brave enough to attempt ColdFusion/Java integration, creating applications that leverage Java code where appropriate. But most developers are failing to take advantage of what is undoubtedly the most significant benefit of ColdFusion on top of J2EE, a benefit so important that for some organizations it has become the only reason needed to upgrade.

Understanding the Problem

Have you ever run into any of these situations?

- Your ColdFusion application is running perfectly. That is, until someone else (no, not you) writes really poor code that kills performance or worse, hurts all the applications including yours in the process.
- You have several applications on your server, and want to apply a product update to some of them (the ones you have tested against) but not others.
- You host applications for multiple departments or clients on the same server, but can't give developers access to the ColdFusion Administrator for fear that they'll tamper with each other's settings.
- Or for that matter, you want to have different ColdFusion Administrator



By Ben Forta

settings for different applications (trusted cache, custom tag paths, site-wide error handlers, and more).

- You don't have multiple physical servers, but still want to provide failover in case your application goes down.

The common problem in all of these situations is that there is one shared copy of ColdFusion on any given server. All settings are shared, all paths are shared, all data sources are shared, all updates are shared, and even all bad code (or the results thereof) is shared.

There is really only one solution to this problem. Each application needs to have its own ColdFusion. That way applications won't interfere with each other. It's as simple as that. Of course, that requires buying lots of servers and lots of copies of ColdFusion, right? Wrong. Read on.

Is There a Single Server Solution?

At DevCon 2001 in Orlando, I publicly demonstrated the following:

- I wrote CFML code that would throw ColdFusion into an endless loop (no easy task mind you).
- I ran that code and showed that ColdFusion was no longer responding to requests properly.
- I then opened another browser window and requested a ColdFusion page from the same server, and it loaded perfectly (even though the first page was still stuck and not responding).

The trick? There wasn't one. What I demonstrated was all real, no smoke and mirrors. The reason it worked is that I had more than one copy of ColdFusion installed on my computer, it's as simple as that.

Well, maybe not so simple. It did take us longer to deliver this functionality than we had originally planned, but with ColdFusion MX for J2EE we have indeed delivered the ability to run multiple instances of ColdFusion on a single server.

Understanding Multiple Instances

J2EE servers are application servers, they serve applications, and usually

Note:

This entire discussion pertains to ColdFusion MX for J2EE. Standalone ColdFusion does not support the running of multiple instances on a single server.

lots of them. In a J2EE world, applications are deployed onto a J2EE server, and each Java application lives in a safe and isolated environment. Each Java application is treated as if it were running on its own server – applications can be stopped, started, upgraded, tweaked, and managed entirely independent of any other Java application. In fact, the same Java application may be deployed multiple times on the same J2EE server, and each instance of that same application can also be stopped, started, upgraded, tweaked, and managed entirely independent of any other instances of that same Java application.

What has this to do with ColdFusion? ColdFusion MX is actually not a server; technically it's an application, a Java application that runs on top of a J2EE server. And like any Java application running on a J2EE server, you may deploy multiple instances of ColdFusion MX on the same physical hardware, multiple instances that are entirely independent of each other.

The Benefits

What are the benefits of running multiple instances of ColdFusion MX on J2EE? There are lots, some obvious and some less so:

- A ColdFusion instance cannot step on the feet of any other ColdFusion instance. That bad SQL statement that someone else wrote need no longer hurt the performance of your application.
- Each ColdFusion instance has its own ColdFusion Administrator and its own ColdFusion settings. The owner of each ColdFusion instance can safely make ColdFusion Administrator changes that will not affect other instances.
- ColdFusion instances can be upgraded independent of each other.
- Multiple instances are more secure than a single shared instance.
- Most J2EE servers (including Macromedia JRun) allow you to cluster instances on the same box. In other words, you can have multiple copies of the same ColdFusion application (each running in a separate

ColdFusion instance) on the same machine, and cluster them so that you have application failover on a single box.

- ColdFusion even allows the SESSION scope to be clustered. In the past (ColdFusion 4 and 5) it was close to impossible to use SESSION and clustering at the same time. You either had to use CLIENT variables and write them to a shared database, or use some sort of sticky IP to essentially prevent clustering while a SESSION was active. ColdFusion MX on top of J2EE leverages Java sessions that can be clustered (on some J2EE servers). As such, ColdFusion SESSION variables (with the notable exception of CFCs) can be shared across ColdFusion instances (on the same box or even on different boxes).

In other words, you get all the functional benefits that you'd get by running ColdFusion on multiple servers (without needing multiple servers), you get the ability to cluster your applications even on a single box, you get dramatically

HOSTMYSITE.COM

www.hostmysite.com

improved session state support in clusters, and the best part is... you don't need to buy a copy of ColdFusion for each instance. The ColdFusion license allows for as many instances as you'd like on the same server. So, you get all of the listed benefits, and cost savings too.

So, How Many Instances?

Now that you know that you can create as many instances as you'd like, the next question is, should you? You can use ColdFusion MX for J2EE as you do standalone ColdFusion, lots of applications in a shared instance. You can also create multiple instances. Or you can do both.

I've already listed all the pros to creating multiple instances of ColdFusion, but there is an important con – each instance is a full ColdFusion install running a full version of ColdFusion. As such, each additional instance requires additional system resources, about 100MB RAM (or so) per instance. RAM is pretty cheap nowadays, but even so, having hundreds of instances is probably very impractical. (An ISP, for example, hosting several hundred ColdFusion clients on a single box, will probably not create an instance for each and every client.)

The ideal number of instances is really dependent on your application and needs. You may have applications that can share an instance. You may have some applications that need their own instances. You may even have applications that each

“While installing and configuring ColdFusion on top of J2EE is not as simple as installing standalone ColdFusion, it's actually not that painful either”

require multiple instances. The key is that you have the flexibility to do whatever works best for you. ColdFusion supports it, and makes it remarkably cost effective too.

Getting Started

By now you probably do not need any more convincing of the value of ColdFusion on top of J2EE. If you are itching to get started, here are some things to keep in mind:

- Obviously, you'll need a J2EE server. Popular choices include IBM WebSphere, BEA WebLogic, Sun ONE, and our own Macromedia JRun. If you are not yet running a J2EE server then look at JRun – not only is it the most economical commercial J2EE implementation, you'll also find it to be the easiest to install and configure.
- Once you're running a J2EE server you'll need to install ColdFusion on top of it. Installing ColdFusion for J2EE is very different from installing standalone ColdFusion. J2EE applications are installed using special files (EAR or WAR files) that contain entire applications. These files are expanded and processed by management software provided with the J2EE server (and each J2EE is different of course). The process of installing ColdFusion involves running the ColdFusion installer program which creates an EAR or WAR file, and then using the J2EE server management tools to deploy the created EAR or WAR file.
- For what it's worth, servers like JRun will allow you to deploy ColdFusion as an EAR file or as a WAR file. EAR files can contain more resources than a WAR file can, but that's not really relevant to ColdFusion installs. From a ColdFusion perspective, the advantage to using WAR files is that using an EAR file you'd need to run the ColdFusion installer to generate a unique EAR file for each instance, and using WAR files you can reuse the same WAR file repeatedly.
- When you install a J2EE application (an EAR or a WAR) it's installed into a virtual path called a *context root*. So, path `http://myserver/app1/` would point to the app1 context root, and `http://myserver/app2` would point to the app2 context root. It is the context root in a URL that tells the J2EE server which application to use. Each application (and thus each instance of ColdFusion) goes into its own context root. If you are deploying a WAR file the management interface will prompt for the context root to use at

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
ACTIVEPDF	WWW.ACTIVEPDF.COM	866.GOToPDF	4
CFDYNAMICS	WWW.CFDYNAMICS.COM	866.233.9626	19
COOLFUSION	WWW.COOLFUSION.COM	631.737.4668	11
CRYSTALTECH	WWW.CRYSTALTECH.COM	1-877-323-HOST	23
EDGE WEB HOSTING	WWW.EDGEWEBHOSTING.NET	1.866.EDGEWEB	3
FUSETALK	WWW.FUSETALK.COM	866.477.7542	31
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.HOST	29
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800.379.7729	Cover IV
IVIS TECHNOLOGIES	WWW.IVIS.COM	602.346.5045	15
LINUXWORLD CONFERENCE & EXPO	WWW.LINUXWORLDDEXPO.COM		32-33
MACROMEDIA	WWW.MACROMEDIA.COM/GO/CERTIFICATION1/		17
MACROMEDIA	WWW.MACROMEDIA.COM/GO/DRK3/		Cover III
NETQUEST	WWW.NETQUEST.COM		Cover II
NEW ATLANTA COMMUNICATIONS	WWW.NEWATLANTA.COM		6
PACIFIC ONLINE	WWW.PACIFICONLINE.COM	877.503.9870	27
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	25
SOUTHWEST SUPPORT SOLUTIONS		866.654.5443	21
SYS-CON SUBSCRIPTIONS	WWW.SYS-CON.COM/2001/SUB.CFM	888.303.5282	37
WEB SERVICES EDGE WEST 2003	WWW.SYS-CON.COM	201.802.3069	42-43


General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in ColdFusion Developer's Journal. Advertisements are to be printed at the discretion of the Publisher. This disclaimer includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

deployment time. If you are deploying an EAR file then the ColdFusion installer will prompt for the context root, which will be embedded in the EAR file itself (this is why I said that using WAR files to deploy ColdFusion is advantageous in that the same WAR file could be used for each instance).

- It is possible to have ColdFusion MX run in the *default context* (/ without a specified context root in every URL). The exact steps to configure this vary based on the J2EE server being used, so consult your J2EE server documentation.
- Most J2EE servers (including JRun) allow application instances to be bound to different virtual hosts so that you do not need to always use the context root. Unfortunately, this tends to be very J2EE implementation specific, so consult your J2EE server documentation. JRun, for example, supports the ability to deploy multiple applications at the default context path so long as each is deployed into a different virtual host.
- Each instance of ColdFusion gets its own copy of the ColdFusion Administrator, Flash Remoting, and other ColdFusion features and technologies. To invoke the correct ColdFusion Administrator, for example, you just need to be sure to use the right context root in the URL.
- Clustering J2EE applications is also very J2EE implementation specific. JRun, for example, comes with an interface called the JMC (JRun Management Console), which has a series of screens that walk you through cluster creation. Again, consult your J2EE server documentation.

If all this sounds foreign to you, don't panic. While installing and configuring ColdFusion on top of J2EE is not (and cannot be) as simple as installing standalone ColdFusion, it's actually not that painful either (especially if you are running JRun). And Macromedia is committed to making the process even easier in the future.

Summary

ColdFusion now comes in two distinct flavors, standalone ColdFusion and ColdFusion for J2EE. Although all versions of ColdFusion run on top of J2EE servers (standalone ColdFusion has a scaled down version of JRun embedded within it), the ability to run multiple instances requires ColdFusion on top of a J2EE server. For many organizations, the security, scalability, stability, manageability, and cost savings of running multiple instances of ColdFusion are the compelling reasons to upgrade not just to ColdFusion MX, but to ColdFusion MX for J2EE. 

About the Author

Ben Forta is Macromedia's senior product evangelist and the author of numerous books, including ColdFusion MX Web Application Construction Kit and its sequel, Advanced ColdFusion MX Application Development, and is the series editor for the new "Reality ColdFusion" series. For more information visit www.forta.com.

ben@forta.com

FUSETALK

www.fusetalk.com



Where Open Minds Meet

- **Learn** how companies have achieved higher profits and increased their productivity by utilizing Linux
- **Participate** in LinuxWorld's **world-class** education program and benefit from interactive training in the **all-new Hands-on Labs!**
- **Discover** the latest innovations and technologies from the hottest companies around
- **Hear** the latest developments and updates on the state of open source at our analyst roundtable discussion

CORNERSTONE SPONSOR



PLATINUM SPONSORS



GOLD SPONSOR



SILVER SPONSOR





Conference: August 4-7, 2003

Expo: August 5-7, 2003

The Moscone Center
San Francisco, CA

Join us this August and see why Linux is thriving! Government agencies and companies in the telecommunication, financial services, retail and manufacturing industries are turning to Linux to save money. Isn't it time you did? **LinuxWorld. Where Open Minds Meet.**

Attend Keynotes and learn from inspiring visionaries who are devoted to Linux and open source.



Tuesday, August 5th
10:30am-11:30am

Linux: The Next Step
Peter Blackmore
Executive Vice President
Enterprise Systems Group
Hewlett-Packard



Tuesday, August 5th
1:30pm-2:30pm

Jonathan Schwartz
Executive Vice President
Software Group
Sun Microsystems, Inc.



Tuesday, August 5th
4:30pm-5:30pm

Matthew Szulik
Chairman, Chief Executive
Officer and President
Red Hat



Wednesday, August 6th
10:30am-11:30am

**Linux and the Evolution of
the Internet**
Irving Wladawsky-Berger
General Manager
IBM Corporation



Wednesday, August 6th
3:30pm-4:30pm

Charles Rozwat
Executive Vice President
Server Technologies Division
Oracle Corporation

Special Presentation

Open to All Registered Attendees

Tuesday, August 5th
12:00pm-1:00pm

The Golden Penguin Bowl

Host: Chris DiBona, Vice President – Marketing;
Co-Founder, Damage Studios, Inc.



Analyst Roundtable

Open to All Registered Attendees

Wednesday, August 6th
1:30pm-2:30pm

State of Open Source Roundtable

Moderator: Larry Augustin, Partner, Azure Capital Partners

Panelists: Pierre Fricke, Executive Vice President of Web Application Infrastructure and Product Lifecycle Management (PLM) Infrastructure, D.H. Brown Associates, Inc.; Daniel Kusnetzky, Vice President, System Software Research, IDC; Ted Schadler, Principal Analyst in Software, Forrester; George Weiss, Vice President and Research Director, Gartner

www.linuxworldexpo.com

The Secret Powers of Includes

CFINCLUDE is much more than just a way to 'pull in code'

You've probably been told for years that CFINCLUDE is like a compile-time directive that "pulls code" from another file into your template for reuse. That's wrong. I'll prove it to you. Indeed, have you ever tried to include something other than a CF template? You can. Wonder what including a text or XML file might do?

If you're reusing CFML code, have you ever wished you could reuse a template *and* have its associated Application.cfm be executed? You can. Not with CFINCLUDE, though. In fact, did you know there are at least 13 ways to reuse code? It's not just CFINCLUDE, custom tags, Application.cfm, and CFLOCATION anymore. (Why do I consider CFLOCATION a means of reuse? I'll discuss this later.)

In this article, we'll see these magical secret powers of includes, some of which apply to CFMX only, some that apply to CF4 and 5, and some that are in BlueDragon only. But the first two assertions above, about whether CFINCLUDE is a compile-time directive and includes other than CF templates, are really not new at all. You may have been laboring under false presumptions for years.

CFINCLUDE As Code Include

I'm willing to bet that you've always been told (and simply accepted) that CFINCLUDE's job was to pull code in from another template for reuse. A classic example may be reuse of headers and footers. You define the needed HTML tags for the header in something like header.cfm, and use it in your template with CFINCLUDE TEMPLATE="header.cfm". The result is that the header appears at the top of the page.



By Charlie Arehart

Simple, right? Of course, the beauty is that if you ever need to change the included file, you change it once and all files that include it will be effected immediately.

You probably also know that if you wanted to use variables in both the including and included templates, you need to be careful because they shared variable scopes. Let's assume we have main.cfm that uses CFINCLUDE TEMPLATE="included.cfm". If you set a variable in main.cfm, its value is seen in included.cfm and vice versa. It's important to understand that because if you aren't careful, you might change the value of a variable in included.cfm and not realize it will "flow back" into main.cfm, where the output will be "Sally" (see Figure 1).

Most programmers know these things and just accept them as gospel. And it

gives all the more credence to the notion that CFINCLUDE "pulls the code" of included.cfm into main.cfm, and then interprets/compiles main.cfm. Indeed, the very notion of "include" files is common in other languages (such as C's #include) and they work just that way, so it's natural to assume they would do so in CFML.

But CFML's include is different. Need proof? For one thing, if it was a "compile-time" directive, you could open a tag in the calling template and close it in the included template. But that's not possible.

Still further proof that it's not a compile-time directive is that if you change the included file, the including file (main.cfm, above) would have to be recompiled to see the changes. That's not what happens.

Indeed, now that CFMX creates compiled Java classes, you can see that for yourself. Look at the list of Java classes created by CFMX (in cfusionmx\www-root\WEB-INF\cfclasses). Using Windows Explorer or My Computer (on a Windows machine), sort the list by Modified Date so that the most recent dates are at the top of the list. If you just created the two templates above, and ran them, their corresponding class files would be listed first. (The mapping of CF template names to Java class names in CFMX is beyond the scope of this article.)

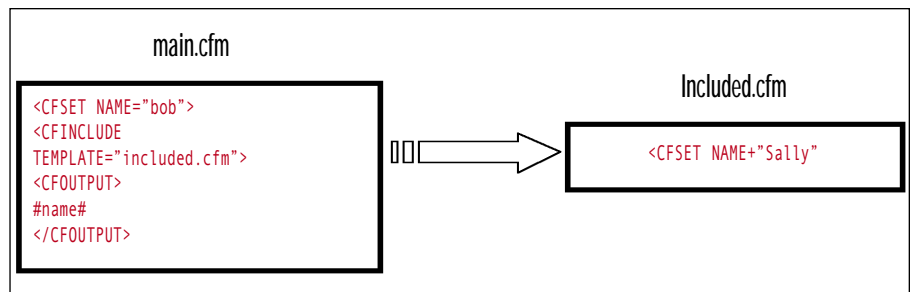


Figure 1

The important point to note, however, is that if you change the included file (included.cfm in our example) and run the calling program (main.cfm), you'll see that while the underlying class file for included.cfm is updated, that for the calling file is not. This shows that CFINCLUDE is a runtime – not a compile-time – directive. And the same was true in CF 4 and 5 (and BlueDragon).

Still further proof lies in the fact that you can set the included template name dynamically. "What?" you may exclaim. The TEMPLATE attribute can be a variable? Yes. Try it.

CFINCLUDE As Output Grabber

So what's really happening? It may shock some to learn that CFINCLUDE really works much like a custom tag. Not quite, but perhaps very differently than you've been led to believe. What you get when you CFINCLUDE a template is shared access to the processing of the template (creation of variables, etc.) and the *output* of that template. It may feel like "inclusion of source code" but it's not, really. That's quite a surprise for many, I'm sure.

But here's something that may surprise many. If you carry this observation (that CFINCLUDE grabs output) to its logical conclusion, then it would seem reasonable that you could use it to pull in more than just CFML. Indeed, even that very phrase prolongs the incorrect assertion. You're not "pulling in" CFML. You're executing it and inserting the output of the named file at the given point in the program. But what if it's not CFML that you're including?

Could you CFINCLUDE an XML file? Sure! Again, it's not about inserting the "code" of that included file, just its output. The only question is whether it makes sense to "include" such a file. But clearly, the notion of CFINCLUDE being just a means to "reuse CFML code" is a limited one.

Some files you include won't display properly unless the MIME type of the page has been set correctly for the given file. And you may want to turn off debugging and any other whitespace generating aspects of your CFML page. Here's an example:

```
<cfsetting showdebugoutput="No">
<cfcontent type="text/plain">
<cfinclude template="test.xml">
```

If you have trouble seeing the output in Internet Explorer, try sending it to Netscape instead. IE tries to render the XML using a built-in stylesheet. If the output isn't pure XML (as if there are CFML errors being rendered as HTML on the page), IE won't show the page at all.

CFINCLUDE As Content Sender

If you could include an XML document this way, then clearly CFINCLUDE is more than just for code. How about a CFINCLUDE of a PDF file? Will that work? What will happen? Sure it will work, if you preface it with <CFCONTENT TYPE="application/pdf">, in order to set the MIME type. And if the browser is set up for PDF, the document will open in the reader.

That may blow away some who have struggled with (or been prevented from using) CFCONTENT as the means to send the file as well as set the MIME type. (Indeed, I've long felt that while the use of CFCONTENT's FILE attribute should be restricted for security reasons, the TYPE attribute when used alone shouldn't at all.) This does beg the question of when you would choose one over the other.

There is at least one difference between the two: CFCONTENT's FILE attribute accepts a full path (drive

letter/directory/file) whereas CFINCLUDE's TEMPLATE attribute permits only a relative path (or can leverage an Administrator mapping). Also, CFCONTENT allows specification of the MIME type in the tag itself, and it stops processing of the page. Clearly CFCONTENT has its purpose. But if you don't need either of those benefits and are precluded from using CFCONTENT, CFINCLUDE is an interesting alternative.

Does the ability to use CFINCLUDE in a similar way open the door to a security issue? If you presume that locking down CFCONTENT locked down any way to "send" files to the browser, I suppose so. Just remember that with both, it's only an exploit for someone able to create CFML code on the server.

CFINCLUDE As File Reader

Here's another interesting twist: What if you used it to read in a text file, surrounding it with CFSAVECONTENT? Would that be the same as CFFILE ACTION="read". Yep. Here's an example, assuming you want to read in a file called test.txt:

```
<CFSAVECONTENT VARIABLE="input">
  <CFINCLUDE TEMPLATE="test.txt">
</CFSAVECONTENT>

<CFOUTPUT>#input#</CFOUTPUT>
```

Again, this may trouble those who've locked down CFFILE as the only means to read in a file. Of course, there are still other ways to read in files, using CFHTTP and Java classes, so this may

Don't Miss CFDJ's Next Issue!



Lucene and ColdFusion...

Writing a Java-based CFX Tag: How to write a Java-based CFX tag using the ColdFusion Extension Application Programming Interface to create, populate, and search a Lucene index.

ColdFusion MX/J2EE Hybrid Application...

10 Best Practices: Surely there are infinite possibilities for integrating CFMX with J2EE, but which offer the best that both frameworks have to offer?

Design Patterns...

Creating Object Instances in ColdFusion: Creating and managing CFC instances.

ColdFusion and SQL Server Permission Integration...

A how-to guide for setting up a ColdFusion 5 server and a Microsoft SQL Server 7.0 that will execute a DTS package through the ColdFusion Application Server.

Plus...

EXCLUSIVE COVERAGE OF RED SKY, the next ColdFusion MX release.

not be as big a deal from a security standpoint. But it may still surprise many. Now do you see why I titled the article as I did?

Processing Application.cfm on Include

There's one aspect of CFINCLUDE, and indeed custom tags, that has long bothered some. Well, some may like how it works, while others likely haven't even noticed.

When you CFINCLUDE code or call a CF custom tag, have you ever wished that the included/called template would execute just like a full-fledged CF template, at least with respect to executing any associated Application.cfm (and optional OnRequestEnd.cfm)?

Again, to some this would seem lunacy to want it. But just know that there are cases when it's been desirable. But CFINCLUDE doesn't work that way, nor do custom tags. The closest you could come was using CFHTTP (or CFLOCATION, if you didn't mind changing pages) to cause the user to see the full-fledged output of one template from another.

Am I going to tell you that you can make CFINCLUDE work differently? No. (Though I will point out in a moment that BlueDragon does indeed offer that option.)

What I'll point out first is that there's a little-known fact about one of the new features in CFMX, which some may still not yet know exists, called `getpagecontext().include()`.

If you read the CFMX docs (particularly Chapter 32 of the "Developing CFMX Applications with CFML" manual), you'll learn that this "function" is used to include the output of JSP pages within CFML. I say "function" because the format of the syntax is unlike anything we've ever had in CFML. The closing parens at the end of `getpagecontext().include()` that follows takes as its argument the JSP page to be included, as in `getpagecontext().include("mypage.jsp")`.

But this isn't just a curious alternative format, nor is it just for calling JSP pages. It can call CFML templates as well. More important, this form has the unique characteristic of causing the included file to really be processed just like a full-fledged page – and yes, with any Application.cfm (and optional OnRequestEnd.cfm) that would be executed if you browsed the template directly.

This is a staggering discovery for those who need that solution. If you ever catch yourself using CFHTTP against a local file just to get the output of a page because you want it executed "fully" but have its output included inside another page, this is the solution for you. And who knows what other problems may exist for which this may be the solution. Please drop me a line to let me know.

BlueDragon, an alternative platform for running CFML applications, does add a couple of things to make your life easier with respect to this discussion. The new include functionality for processing Application.cfm, discussed above, comes as a benefit of the underlying J2EE server on which CFMX runs. And like CFMX, BlueDragon is also built on the J2EE platform. Indeed, while it doesn't yet support most CFMX tags, BlueDragon does offer Java-based features like this as well as J2EE sessions, the ability for JSP pages to exist alongside your CFML templates, and the ability to integrate with Java in many of the same ways that CFMX can.

And with respect to this discussion of includes, it also adds a couple of things to make your life easier (and they're different from CFMX because they were added to BlueDragon before CFMX came out).

Similarly, while there is an available `getpagecontext().forward()` in CFMX for server-side redirection (as I discussed in the June 2002 *CFDJ* article, "New Possibility in CFMX: Server-Side Redirects"), BlueDragon offers the simpler CFFORWARD. This is quite different from CFLOCATION and an important new tool for many applications. But it's the same concept as CFMX's approach. Indeed, to use Java/OO terminology, it's the same "implementation," but a different "interface". Again, New Atlanta had theirs first and it seems MM just didn't think of it. There are just a few such differences where New Atlanta has decided to innovate rather than wait.

While the forward approaches and CFLOCATION may seem more about redirecting control in a program, the fact is that they can be key to reuse of code when designing to the Model-View-Controller paradigm.

13 Ways to Reuse

That leads nicely to the last point I had raised at the beginning: that there are now 13 ways to reuse code in CFML. The tradi-

tional ones from CF 4 and 5 and on into MX (and of course BlueDragon) are CFINCLUDE, custom tags, CFMODULE (admittedly, just another way to call custom tags versus the CF_approach), Application.cfm, OnRequestEnd.cfm, CFLOCATION, and CFSCRIPT-based UDFs.


CFMX adds `getpagecontext.include()` and `getpagecontext.forward()` as well as CFCs and CFFUNCTION, while BlueDragon adds CFINCLUDE PAGE and CFFORWARD. New Atlanta has committed to supporting CFMX tags and functions in its Release 4 version, due later this year. And who knows, perhaps we'll see CFFORWARD and CFINCLUDE PAGE showing up in CFMX some day. That's one of the great things about competition.

You might even add CFX custom tags to the list of code reuse options, but those don't allow reuse of CFML code or other Web content, which is clearly the focus of this article.

Summary

In any case, whether you use CFMX, CF 4 or 5, or have made the move to BlueDragon, I hope you've learned a lot from this article.

We learned that CFINCLUDE is much more than just a way to "pull in code". It opens doors to interesting similarities to CFCONTENT and CFFILE. And we learned that while CFINCLUDE can't call upon an included page to execute its associated Application.cfm, the new J2EE-based variants in both CFMX and BlueDragon can.

There are certainly a lot of surprises in the secret powers of includes. I hope this may open the doors to some very interesting new application development possibilities. Again, please let me know if it does. 

About the Author

Charlie Arehart is co-technical editor of ColdFusion Developer's Journal and a Macromedia Certified Advanced ColdFusion developer and trainer. He has recently become CTO of New Atlanta Communications, makers of BlueDragon. In his new role, he will continue to support the CFML community, contributing to several CF resources, and speaking frequently at user groups throughout the country.
charlie@newatlanta.com

A LIMITED TIME SAVINGS OFFER FROM SYS-CON Media

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

AND SAVE UP TO \$400 AND RECEIVE UP TO 3 FREE CDs!

RECEIVE
YOUR DIGITAL
EDITION
ACCESS CODE
INSTANTLY
WITH YOUR PAID
SUBSCRIPTIONS



TO ORDER

- Choose the Multi-Pack you want to order by checking next to it below. • Check the number of years you want to order. • Indicate your location by checking either U.S., Canada/Mexico or International. • Then choose which magazines you want to include with your Multi-Pack order.

Pick a 3-Pack, a 6-Pack or a 9-Pack

<input type="checkbox"/> 3-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 6-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 9-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.

☐ **Linux World Magazine**

U.S. - Two Years (24) Cover: \$143	You Pay: \$79.99 /	Save: \$63 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 /	Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ **Java Developer's Journal**

U.S. - Two Years (24) Cover: \$144	You Pay: \$89 /	Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 /	Save: \$22
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ **Web Services Journal**

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ **.NET Developer's Journal**

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ **XML-Journal**

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ **WebLogic Developer's Journal**

U.S. - Two Years (24) Cover: \$160	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$80	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

☐ **ColdFusion Developer's Journal**

U.S. - Two Years (24) Cover: \$216	You Pay: \$129 /	Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 /	Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 /	Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 /	Save: \$20
Intl - Two Years (24) \$264	You Pay: \$189 /	Save: \$75 + FREE \$198 CD
Intl - One Year (12) \$132	You Pay: \$129.99 /	Save: \$2

☐ **Wireless Business & Technology**

U.S. - Two Years (24) Cover: \$144	You Pay: \$89 /	Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 /	Save: \$22
Can/Mex - Two Years (24) \$192	You Pay: \$139 /	Save: \$53 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$79.99 /	Save: \$16
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ **WebSphere Developer's Journal**

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

☐ **PowerBuilder Developer's Journal**

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

3-Pack
Pick any 3 of our
magazines and save
up to **\$275⁰⁰**
Pay only \$175 for a
1 year subscription
plus a **FREE CD**

- 2 Year - \$299.00
- Canada/Mexico - \$245.00
- International - \$315.00

6-Pack
Pick any 6 of our
magazines and save
up to **\$350⁰⁰**
Pay only \$395 for a
1 year subscription
plus 2 **FREE CDs**

- 2 Year - \$669.00
- Canada/Mexico - \$555.00
- International - \$710.00

9-Pack
Pick 9 of our
magazines and save
up to **\$400⁰⁰**
Pay only \$495 for a
1 year subscription
plus 3 **FREE CDs**

- 2 Year - \$839.00
- Canada/Mexico - \$695.00
- International - \$890.00

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Subscribe Online Today www.sys-con.com/2001/sub.cfm

**SYS-CON
MEDIA**

JavaScript Without the Headaches

Spare yourself hours of frustrating debugging

Coding ColdFusion and coding JavaScript are about as far apart on the productivity spectrum as it's possible to be. CF tags are neat, easy to read, tolerant, and fun to write. JavaScript is none of that. It has all the drawbacks of traditional languages, with the added disadvantage of sitting somewhat awkwardly within the HTML document model.

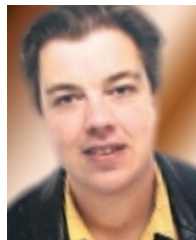
Anything but the most trivial JavaScript is difficult to write, difficult to debug, and difficult for either the author or another developer to revisit, but, as much as I'd like to, it isn't possible to do without it for interactive Web site development. For all its faults, JavaScript remains the principal means of delivering client-side functionality.

For better or worse, we have to learn to live with it, and fortunately ColdFusion makes it easy to integrate JavaScript into your apps without having to code and debug chunks of JavaScript every time you want to extend the basic HTML form controls.

Making the Most of CFFORM and CFINPUT

The primary and most underrated way of integrating JavaScript with your CF apps is through simple CFFORM and CFINPUT tags. It's wrong to assume that just because you want to do advanced validation you need to start from scratch coding all your own routines. These tags form a great platform to start building more advanced validations.

Obviously, there are the REQUIRED and VALIDATE attributes that will do the most common validations without having to write or debug JavaScript, but the



By Tom Peer

ONVALIDATE and ONERROR attributes can be used to do more advanced validations without necessarily having to get too involved either.

About OnValidate

ONVALIDATE is the more useful and more powerful. The value supplied to the

attribute is simply the name of the JavaScript function to call to validate the field. The parameters passed to the routine are always the same: the form object, the field object, and the value of the field. For example, in a basic validation where you want a password field to be 6 letters or more, you'd use the following code:

```
<script language="JavaScript">
    function validatePassword(form,field,value)
    {
        if (value.length < 6) {
            return 0;
        }
        else {
            return 1;
        }
    }
</script>
```

This would then be called by a CFINPUT tag like this:

```
<cfinput type="Password" name="pw"
    message="Please enter a password of
    at least than 6 letters"
    onvalidate="validatePassword">
```

(See Listing 1 for complete listing. This also shows a neat trick for using a hidden cfinput to validate a textarea tag.)

On balance, the fact that the parameters are fixed like this is a help, although the drawbacks are obvious. What if you want to vary the required length? You can't pass the length as a parameter, so you end up altering your JavaScript – which is specifically what we want to avoid by using tags like CFINPUT.

This problem (which we'll look at later) is outweighed by the convenience of not having to code references to the form, field, and value each time. It doesn't, of course, limit you to only referencing that field – you can reference other fields quite simply, as in this check for a confirmation password field:

```
// Check that the confirm password field matches the first
function
checkPasswordsMatch(form,field,value) {
    if (value != form.pw.value) {
        return 0;
    }
    else {
        return 1;
    }
}
```

Which you'd use in conjunction with a CFINPUT:

```
Confirm: <cfinput name="pw_check" message="The
passwords do not match"
    onvalidate="checkPasswordsMatch">
```

(See Listing 2 for complete example.)

This again suffers from the problem that the validation script isn't parameterized – the name of the first password field is hardwired into the code, something that'd give the object purists palpitations.

About OnError

ONERROR can be used to override the default behavior of simply displaying an error message if the validation fails. You can use it to display different error messages for different validation failures (e.g., blank or not enough characters).

For instance, the following script could be used to validate the pw_check field shown previously in Listing 2.

```
function pw_check_error(form,field,value) {
    if (value.length == 0) {
        alert('Please confirm your password');
    }
    else if (value != form.pw.value) {
        alert('The passwords do not match');
    }
}
```

(See Listing 3 for complete example.)

NB the onerror script is called if required=yes and the field is blank, as well as when the onvalidate function returns false.

The Challenges of Generated Code

This now has made the problem of having parameters hardwired into the JavaScript even worse. The error messages should be parameterized or even dynamic if we're using a multilanguage setup.

Experienced developers looking at this will be thinking it might be best to start from scratch with their own validation routines, and when you look at code like Listing 3, you can see why.

The solution most developers would adopt is to write JavaScript routines that take things like the minimum length and the error messages as parameters and apply them either in onchange handlers for the fields or in onsubmit handler for the form. Something like this:

```
<input onchange="CheckPW(this.form,
this,this.value, 6, 'Please confirm your password', 'The passwords do not match')">
```

(Note: Don't try this! It's just hypothetical.)

This is the first step on the familiar path of teeth-gnashing frustration that is coding your own JavaScript, and before

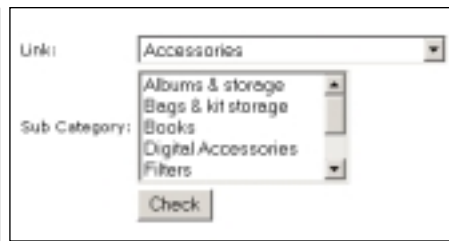


Figure 1: cf_twoselectsrelated is one of the most popular custom tags



Figure 2: cf_swapbox is one of many variants on the multiselect theme for when checkboxes won't work.

we go down that road, let's look at another option for tidying up our code.

If you look at the code generated by CFFORM, you'll notice one thing very quickly – there's lots of it. Also you'll see that it isn't especially generic. The names of fields are hardwired into the code in much the same way as the examples here.

The same is true if you look at the JavaScript generated by several other automated systems. The developers of these systems have made a conscious decision to do it this way instead of trying to create completely reusable functions and objects. JavaScript just doesn't warrant effort spent on trying to make it neat and generic. You can waste hours trying to get it right and then find it doesn't work on one particular build of Mozilla.

Solving the Problem with Custom Tags

Instead, there's a different philosophy you can adopt – that it doesn't necessarily matter how neat the JavaScript is or how much of it there is, as long as it's generated automatically and the code that generates it is properly modular and reusable.

Say for instance we wanted to tidy up our password field. According to correct design principles, the only things we should be including in the page are the necessary details of the field – its name, its minimum length, the fact that it's a password, and that we want a "check" field, all of which can be neatly done with a ColdFusion custom tag:

```
<cf_password name="pw" minlength="6"
```

```
check="yes">
```

(See Listing 4; also see the listing for cf_password.)

All the JavaScript and the cf input tags can then be generated by the custom tag, with the JavaScript placed correctly in the <HEAD> element with a CFHTMLHEAD. The fieldname "pw" may be hardwired into the JavaScript, but as long as it isn't hardwired into the CFML that generates the JavaScript, that doesn't matter. The same goes for the minimum length. This system is all correctly parameterized and reusable without having to waste hours coding JavaScript. It's also readable and easily understandable in the same way as CFML form tags.

Whenever you include client-side functionality you should always adopt this approach, either writing your own custom tags – or even better – using some of the many excellent tags in the developer's exchange. One of the most common requirements is for "related selects" – where selecting a value in one select list changes another list to show only related values.

If you aren't already, you should be using one of the many custom tags available to do this job. Cf_twoselectsrelated (see Figure 1) is a great, simple tag that can be customized quite easily to fit in with cfform rather than using its own validations. The hidden cfinput technique allows for easy validation. In Listing 5, the name of the second select is SUB_CATEGORY, and we can require this or validate it using:


```
<!-- <CFINPUT NAME="SUB_CATEGORY" required="Yes"
message="You must select a sub category"> -->
```

Rather than modifying the tag itself to include this, I recommend creating a new tag that then calls Cf_twoselectsrelated; otherwise you can end up with versioning problems. (Of course, another option for providing these kinds of interfaces is Flash and Flash Remoting. For more information, see Ben Forta's November 2002 **CFDJ** article, "Data Entry Reformed," Vol. 4, issue 11. But there are clearly times when JavaScript alone may make more sense.)

Other useful custom tags include cf_swapbox (Figure 2, Listing 6), cf_cross_select, tags for picking dates or times, and tags for dividing forms up into "tabbed" controls.

If you look at the JavaScript generated by any of these tags, you'll find the same story – lots of JavaScript with hardwired field names and parameter values. It doesn't matter, as long as the ColdFusion generating it is properly modular. If you're going to start coding your own custom tags, you can spare yourself many hours of frustrating debugging by following the same philosophy.

And while this article presumes you're already skilled with JavaScript, if you're just getting started and could use some hand-holding, see Charlie Arehart's June 2000 **CFDJ** article, "Getting Focus(ed) – and a Quick JavaScript Overview," Vol. 2, issue 6.

Spend time on what's most productive: keep your ColdFusion pages neat, and don't lose any sleep over your JavaScript. 

About the Author

Tom Peer is a Web developer specializing in online journals and magazines (www.tompeerconsulting.com). He was formerly manager of the New Scientist online, one of the top science Web sites, and now runs an agency providing design, development, and hosting services to a number of other UK publishers. tom@tompeerconsulting.com

Listing 1

```
<script language="JavaScript">
function validatePassword(form,field,value) {
    if (value.length < 6) {
        return 0;
    }
    else {
        return 1;
    }
}
</script>

<cfform name="form1">
<table>
<tr>
<td>Password:</td>
<td><cfinput type="Password" name="pw"
message="Please enter a password of at least 6 letters"
onvalidate="validatePassword"></td>
</tr>
<tr>
<td>Text:</td>
<td><TEXTAREA name="address" rows="3" cols="26"></TEXTAREA>
<!-- <CFINPUT name="address" required="yes" message="Please enter some text" -->
</td>
</tr>
<tr>
<td>&nbsp;</td>
<td><input type="submit" value="Check"></td>
</tr>
</table>
</cfform>
```

Listing 2

```
<script language="JavaScript">

// Require minimum 6 characters for password field
function validatePassword(form,field,value) {
    if (value.length < 6) {
        return 0;
    }
    else {
        return 1;
    }
}

// Check that the confirm password field matches the first
function checkPasswordsMatch(form,field,value) {
    if (value != form.pw.value) {
        return 0;
    }
    else {
        return 1;
    }
}
</script>

<cfform name="form1">

Password: <cfinput name="pw" required="Yes" message="Please enter a
password of more than 6 letters"
onvalidate="validatePassword">

Confirm: <cfinput name="pw_check" message="The passwords do not match"
```

```
onvalidate="checkPasswordsMatch">
<input type="submit" value="Check">
</cfform>
```

Listing 3

```
<script language="JavaScript">

// Require minimum 6 characters for password field
function validatePassword(form,field,value) {
    if (value.length < 6) {
        return 0;
    }
    else {
        return 1;
    }
}

// Check that the confirm password field matches the first
function checkPasswordsMatch(form,field,value) {
    if (form.pw.value != value) {
        return 0;
    }
    else {
        return 1;
    }
}

function pw_check_error(form,field,value) {
    if (value.length == 0) {
        alert('Please confirm your password');
    }
    else if (value != form.pw.value) {
        alert('The passwords do not match');
    }
}
</script>

<cfform name="form1">
<table>
<tr>
<td>Password:</td>
<td><cfinput type="password" name="pw" required="Yes"
onvalidate="validatePassword"
message="Please enter a password of at least 6 letters">
</td>
</tr>
<tr>
<td>Confirm:</td>
<td><cfinput type="password" onvalidate="checkPasswordsMatch"
onerror="pw_check_error" required="Yes" name="pw_check">
</td>
</tr>
<tr>
<td></td>
<td><input type="submit" value="Check"></td>
</tr>
</table>
</cfform>
```

Listing 4

```
<cfform name="form1">

<table>
  <tr>
    <td>Password:</td>
    <td><cf_Password field="pw" minlength="6" check="yes"
      htmlbetween="</td>
    </tr>
    <tr>
    <td>Confirm:</td>
    <td>">
    <tr>
    <td>&nbsp;</td>
    <td><input type="submit" value="Check"></td>
    </tr>
  </table>
</cfform>
```

Listing 5

```
<!-- Create your own query here as required -->
<cf_example_query return="links">

<cfform name="form1">

<table>
  <tr>
    <td>Link:</td>
    <td> <CF_TwoSelectsRelated
      QUERY="Links"
      NAME1="CATEGORY"
      NAME2="SUB_CATEGORY"
      DISPLAY1="LINKS_CATEGORY"
      DISPLAY2="SUB_CATEGORY"
      VALUE1="LINKS_CATEGORIES_ID"
      VALUE2="LINKS_SUB_CATEGORIES_ID"
      SIZE2="5"
      AUTOSELECTFIRST="No"
      FORMNAME="form1"
      HTMLBetween="</td></tr><tr><td>Sub Category:</td><td>">

    <!-- <CFINPUT NAME="SUB_CATEGORY" required="Yes" message="You must
      select a sub category"> -->
    </td>
  </tr>
  <tr>
    <td>&nbsp;</td>
    <td><input type="submit" value="Check"></td>
  </tr>
</table>
</cfform>
```

Listing 6

```
<!-- Create your own query here as required -->
<cf_example_query return="request.links">

<cfform name="form1">

<table>
  <tr>
    <td valign="top">Sub category:</td>
    <td><cf_swapbox field="SUB_CATEGORY" formname="form1"
      query="request.links" display="sub_category"
      value="LINKS_SUB_CATEGORIES_ID">
    <!-- <CFINPUT NAME="SUB_CATEGORY" required="Yes" message="You must
      select a sub category"> -->
    </td>
  </tr>
  <tr>
    <td></td>
    <td><input type="submit" value="Check"></td>
  </tr>
</table>
</cfform>
```

cf_password

```
<!--
Display a password field
```

```
--->

<!-- Minimum length of password --->
<cfparam name="attributes.minlength" default="6">

<!-- Name of field --->
<cfparam name="attributes.field" default="pw">

<!-- Use a confirm password field --->
<cfparam name="attributes.check" default="1">
<!-- HTML to put between the main field and the confirm field --->
<cfparam name="attributes.htmlbetween" default="<br>">

<!-- Messages are parameterized --->
<cfparam name="attributes.message1" default="Please enter your password">
<cfparam name="attributes.message2" default="Your password must be at
least #attributes.minlength# letters">
<cfparam name="attributes.message3" default="Please confirm your password">
<cfparam name="attributes.message4" default="The passwords do not match">

<cfsavecontent variable="script">
<cfoutput>
<script language="JavaScript">
  // Require minimum characters for password field
  function validate#attributes.field#(form,field,value) {

    if (form.#attributes.field#.value.length < #attributes.minlength#) {
      return 0;
    }
    else {
      return 1;
    }
  }

  // Display error messages
  function #attributes.field#_error(form,field,value) {
    if (form.#attributes.field#.value.length == 0) {
      alert('#attributes.message1#');
    }
    else if (form.#attributes.field#.value.length < #attributes.minlength#) {
      alert('#attributes.message2#');
    }
  }

  <cfif attributes.check>
  // Check that the confirm password field matches the first
  function check#attributes.field#Match(form,field,value) {
    if (form.#attributes.field#_check.value != form.#attributes.field#.value) {
      return 0;
    }
    else {
      return 1;
    }
  }

  // display errors
  function #attributes.field#_check_error(form,field,value) {
    if (value.length == 0) {
      alert('#attributes.message3#');
    }
    else if (value != form.#attributes.field#.value) {
      alert('#attributes.message4#');
    }
  }
  </cfif>

</script>
</cfoutput>
</cfsavecontent>

<cfhtmlhead text="#script#">

<cfinput type="Password" name="#attributes.field#" onvalidate="validate#attrib
utes.field#" onerror="#attributes.field#_error" required="Yes">

<cfif attributes.check>

  <cfoutput>
  #attributes.htmlbetween#
  <cfinput type="Password" name="#attributes.field#_check"
    onerror="#attributes.field#_check_error" onvalidate="check#attrib
utes.field#Match" required="Yes">
  </cfoutput>

</cfif>
```

Download the Code...
Go to www.coldfusionjournal.com

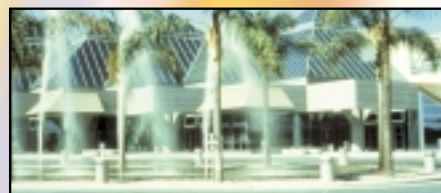
International Web Services Conference & Expo

Web Services Edge ^{WEST} 2003

web services
conference & expo **EDGE**

SEPT. 30 - OCT. 2, 2003

Santa Clara, CA



**EXTENDING THE ENTERPRISE
WITH WEB SERVICES THROUGH JAVA,
.NET, WEBSHERE, MAC OS X
AND XML TECHNOLOGIES**



XML

WebSphere



COMING IN

2004

web services
conference & expo **EDGE**

BOSTON

**FEBRUARY
24-26, 2004**

CONTACTS:

Exhibit & Sponsorship

GRISHA DAVIDA

201 802-3004

grisha@sys-con.com

Conference & Education

MICHAEL LYNCH

201 802-3055

mike@sys-con.com



Over 100 participating companies will display and demonstrate over 300 developer products and solutions.

Over 2,000 Systems Integrators, System Architects, Developers, and Project Managers will attend the conference expo.

Over 60 of the latest sessions on training, certifications, seminars, case studies, and panel discussions will deliver REAL World benefits, the industry pulse and proven strategies.

www.sys-con.com

Java is a registered trademark of Sun Microsystems, .NET is a registered trademark of Microsoft, Mac OS X is a registered trademark of Apple Computer, Inc., WebSphere is a registered trademark of IBM. All other product names herein are the properties of their respective companies.

WEB SERVICES EDGE CONFERENCE & EDUCATION

web
services
conference & expo

EDGE
& expo

SEPT. 30 - OCT. 2, 2003
Santa Clara, CA

WEB SERVICES TECHNOLOGY

Presentations will include discussions of security, interoperability, the role of UDDI, progress of the standards-making bodies, SOAP, and BPM. Case studies cover the design and deployment of Web services in the marketplace.

Sessions will focus on:

- Interoperability
- Enterprise Networks
- Web Services Management
- Web Services Standards
- Web Services Orchestration
- Security (WS-Security, SAML)
- BPEL4WS
- UDDI: Dead or Alive?
- ebXML & Web Services
- EAI & Web Services
- RPC vs. Messaging: Uses and Differences
- User Interfaces for Web Services
- Web Services Best Practices
- Service Oriented Architecture



XML TECHNOLOGY

Presentations will focus on the various facets of XML technologies as they are applied to solving business computing problems. Sessions will include emerging standards in XML Schemas, XML repositories, industry applications of XML, applying XML for building Web services applications, XML/XSLT/XQuery-based programming using Java/.NET, XML databases, XML tools and servers, XML-based messaging, and the issues related to applying XML in B2B/EAI applications. The XML Track is geared for audiences ranging from beginners to system architects and advanced developers.

Sessions will focus on:

- XML Standards & Vocabularies
- Introduction to XForms
- Securing Your XML and Web Services Infrastructure
- XQuery Fundamentals: Key Ingredient to Enterprise Information Integration
- XML and Enterprise Architecture: Technology Trends
- Standards-Based Enterprise Middleware Using XML/Web Services
- XML and Financial Services
- Canonical Documents for Your Business: Design Strategies
- XPath/XSLT 2.0: What's New?
- XML Schema Best Practices
- XML in EAI, Enterprise Portals, Content Management

XML

MAC OS X

OS X represents a new wave of operating systems. It combines the ease of use of a Mac with the power of Unix. Sessions in this track will highlight the use of the Mac OS X platform in applications and Web services development, deployment and management.

Sessions will focus on:

- Introducing OS X (Panther): What's New?
- Quick Applications using AppleScript
- Enterprise Java and OS X
- Developing Web Services Using WebObjects
- Xserve: Ease of OS X and Power of Unix
- Introducing Quartz: 2D Graphics for Apple
- OS X for the Unix Developer
- Securing OS X Applications
- Java and OS X: A Perfect Marriage
- Programming Rich User Interfaces Using Cocoa



JAVA TECHNOLOGY

The Java Track features presentations aimed at the beginner, as well as the seasoned Java developer. Sessions will explore the whole spectrum of Java, focusing on J2EE, application architecture, EJB & J2ME. In addition the Track will cover the latest in SWT, Ant, JUnit, open source frameworks, as well as an in-depth look into the vital role that Java is playing in building and deploying Web services.

Sessions will focus on:

- Enterprise Java 1.4
- Ant Applied in "Real World" Web Services
- Developing Application Frameworks w/SWT
- Empowering Java and RSS for Blogging
- JUnit: Testing your Java w/JUnit
- JDK1.5: The Tiger
- Simplifying J2EE Applications
- Using IBM's Emerging Technologies Toolkit (ETTK)
- Apache Axis
- Meeting the Challenges of J2ME Development
- Integrating Java + .NET
- Squeezing Java



.NET TECHNOLOGY

Presentations will explore the Microsoft .NET platform for Web services. To the average developer, it represents an entirely new approach to creating software for the Microsoft platform. What's more, .NET development products - such as Visual Studio .NET - now bring the power of drag-and-drop, GUI-based programming to such diverse platforms as the Web and mobile devices.

Sessions will focus on:

- ASP.NET
- Security
- VB.NET
- .NET and XML
- Smart Device Extensions for VS.NET
- Best Practices
- Shared Source CLI
- .NET Remoting
- Smart Devices in Health Care Settings
- Mobile Internet Toolkit
- ROTOR
- Portable .NET
- ASP.NET Using Mono
- Using WSE with IBM's WSTK
- GUI applications Using Mono
- Portals - Windows Sharepoint Services/Sharepoint Portal Server
- Windows Server 2003 and IIS 6
- .NET and Java Interoperability
- Distributed .NET for Financial Applications
- Developing C# with Eclipse

Microsoft
.net



For more information visit
www.sys-con.com
or call
201 802-3069

MEDIA SPONSORS:



PRODUCED BY
SYS-CON
EVENTS

Extending ColdFusion with Java

Full text searching using Jakarta Lucene

PART 1

One of the many reasons to use ColdFusion MX is that it comes standard with the majority of the tools you'll need to write full-featured, dynamic Web applications. Tags like `<cfquery>` and `<cfmail>` make it relatively simple to query a relational database and send e-mail. In the same way, you can use `<cfsearch>` and `<cfindex>` to create and search Verity full-text indexes.

There are, however, a couple of situations when you can't use the full-text searching capabilities of Verity. The ability to run ColdFusion MX on the Apple OS X operating system, while a boon to developers who code on the Apple platform, does not include the ability to use Verity. Programmers who work in a hybrid J2EE/ColdFusion MX environment (possibly using ColdFusion MX for J2EE) cannot natively use the Verity search capabilities in the J2EE environment. Finally, programmers who need customized searching and indexing capabilities may find the standard Verity integration limiting.

Enter Lucene, an open source full-text searching framework from the Apache Jakarta project, which, when combined with ColdFusion MX, can be run on Apple OS X, can be programmatically accessed by both J2EE and ColdFusion MX developers, and can be fully customized and extended.

This two-part series will illustrate two different methods you can use when approaching a ColdFusion and Java integration project. In this article, I'll walk you through the creation of three CFML scripts, all using native CFML syntax: one that creates, populates and optimizes



By Aaron Johnson

Lucene indexes; one that searches the same index; and a final script that optimizes the index.

In the next article, I'll show you how to write a Java-based CFX tag using the ColdFusion Extension Application Programming Interface to create, populate, and search a Lucene index.

This article is not intended to be an in-depth introduction to the Lucene API. If you're interested in learning more about the internal workings of Lucene, SYS-CON Media's sister publication, *Java Developer's Journal*, featured an article entitled "Search-Enable Your Application with Lucene," which you can find a link to in the resources section at the end of this article.

Before beginning, you'll need to make sure that your system (be it Unix, Linux, Windows, or Apple) is appropriately configured:

- ColdFusion MX must be installed; you'll see that I'm using the ColdFusion MX integrated Web server running on port 8500 throughout the examples.
- You'll need to download Lucene; binaries and source code are available on the Jakarta Apache site. You'll see links

to those resources at the end of this document.

- After downloading the Lucene JAR file, you'll need to add the location of the JAR file to the classpath in ColdFusion Administrator (<http://localhost:8500/cfide/administrator>; click on "Java and JVM" under "Server Settings," and type the full path to the location of the Lucene JAR file you downloaded (see Figure 1). Make sure that you restart the ColdFusion service after saving your changes.
- Finally, create a folder in your `/cfusionmx/wwwroot/` called "Lucene," into which you'll put the source code written during this article.

Creating a Lucene Index Using ColdFusion

Now that you've downloaded Lucene, modified the ColdFusion classpath, restarted ColdFusion, and created the Lucene folder, you should be ready to write some code! Open up your IDE of choice, create a new file, and save it as "luceneindex.cfm" into the `cfusionmx/wwwroot/lucene/` directory. The first thing you'll need to create a Lucene index is a `StopAnalyzer` object, which is a Java object that eventually will "tokenize" or split up the large blobs of text you feed to the engine into individual words. You'll use `cfscript` syntax to get access to a `StopAnalyzer` object:

```
analyzer = CreateObject("java",
"org.apache.lucene.analysis.StopAnalyzer");
```

The first line of code above uses the `CreateObject()` function of ColdFusion to assign the "analyzer" variable to a variable of the type "org.apache.lucene.analysis.StopAnalyzer". If you're familiar with

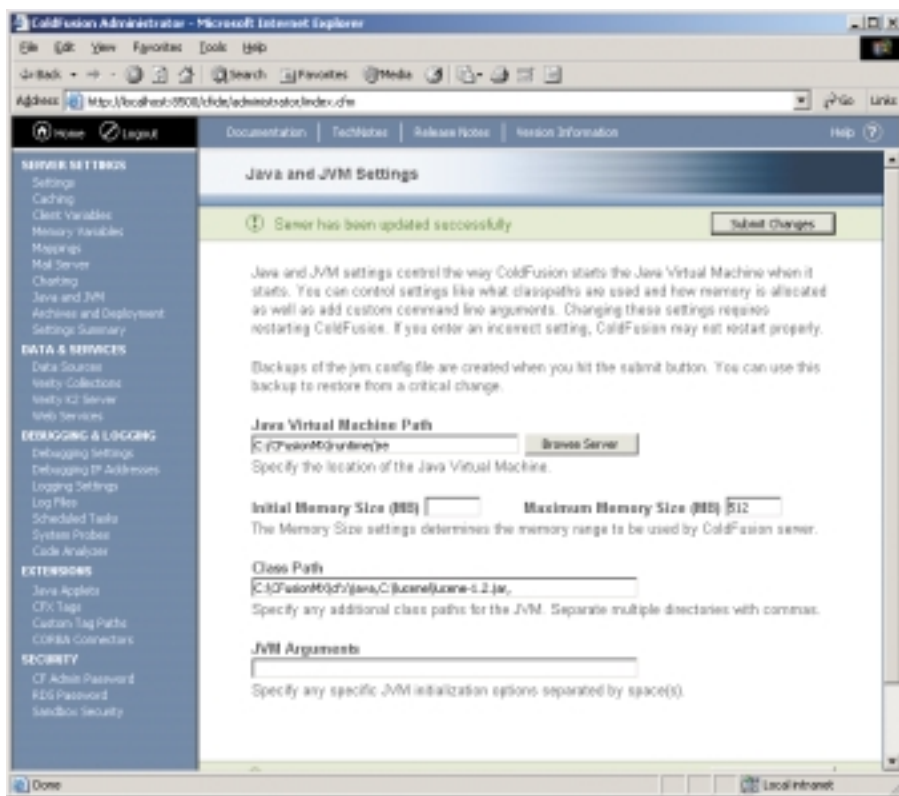


Figure 1: cfadmin.bmp

Java, the cfsript syntax roughly maps to the following Java syntax:

```
StopAnalyzer anazlyer = new StopAnalyzer();
```

You should note that at this point in the code, you do not have a reference to an object; no object has yet been created on the heap in Java. The next line uses a method, "init()" to call the default constructor of the StopAnalyzer object, which returns a reference to a Java object:

```
analyzer.init();
```

Those last two sentences are important so I'll repeat: using the CreateObject() function does not get you access to an instance of an object. In order to access a Java object, you must either a) first call the CreateObject() method and then the init() method, which in the above example, maps to the default constructor in Java, or b) call any nonstatic method on the object, which causes ColdFusion to then instantiate the object for you.

Now that we have a StopAnalyzer object, we'll need to get an object that handles the creation of the Lucene index:

```
writer = CreateObject("java",
"org.apache.lucene.index.IndexWriter");
```

```
writer.init("c:\cfusionmx\wwwroot\lucene\docsindex", analyzer, true);
```

The first line looks very much like the syntax we just used to create the StopAnalyzer object. However, the second line is different. Instead of calling the default Java constructor, we're calling a constructor that takes three arguments: a) the path to the index we want to create (note that Lucene writes out numerous files or "segments" as part of its indexing routine just like Verity does; the "path" mentioned here is the path where you want those files stored); b) a StopAnalyzer object that you created above; and c) a Boolean variable that determines whether to create an index or simply update an index.

To make it easier to understand what's going on, I've hard coded the path to the index above. You'll notice that the source code included at the end of this article uses a variable from ColdFusion's attributes scope, which allows you to determine where you want your index files stored at runtime.

Next, you'll need to instantiate a couple more Java objects:

```
document = CreateObject("java",
"org.apache.lucene.document.Document");
field = CreateObject("java",
"org.apache.lucene.document.Field");
system = CreateObject("java",
"java.lang.System");
```

Next, you'll need to read in the content of the file you want to index. www.cflib.org is a wonderful resource for ColdFusion functions. I found one called FileRead() that does exactly what you'll need. I included a link to the function in the resources section so that you can download it, but for now, just know that I'm using that function to read the entire contents of a file as a string (again I've hard coded the path to the ColdFusion documentation home page on my local system. The source code for the finished product can be downloaded at sys-con.com/coldfusion/sourcecfm).

```
content =
FileRead("c:\cfusionmx\wwwroot\cfdocs\dochome.htm");
```

Now I need to extract the title from the HTML document. I use the FindNoCase() function to get the start and end points of the elements <title> and </title> and then use the Mid() function to get the value of <title></title>:

```
startTitle = FindNoCase("<title>", content);
endTitle = FindNoCase("</title>", content);
if (endTitle GT 0) {
    title = trim(Mid(content, startTitle + 7,
endTitle - startTitle - 7));
}
```

Now you'll use the add() method of the "document" object in combination with the "field" object to add the URL, the title, and body of the file you just read using FileRead() to the document that Lucene will index:

```
document.add(field.Keyword("url",
"http://localhost:8500/cfdocs/dochome.htm"));
document.add(field.Text("title", title));
document.add(field.UnIndexed("summary", content));
document.add(field.UnStored("body", content));
```

Finally, we use the addDocument() method of the writer object to add the

document to the Lucene index

```
writer.addDocument(document);
```

After you're done adding documents to the index, you'll use the `close()` method of the writer object:

```
writer.close();
```

You're done! If you've been following along in your editor, save the file and request the page:

```
http://localhost:8500/lucene/luceneindex.cfm
```

After running the page, you should see a bunch of oddly named files in the `/cfusionmx/wwwroot/lucene/docsindex/` directory. Now you're ready to search the Lucene index.

Searching a Lucene Index Using ColdFusion

To start, create a new file, save it in the `/cfusionmx/wwwroot/lucene/` directory as `"luceneindex.cfm"`. You'll use CFScript syntax again, so add a `<cfscript></cfscript>` block to your script. After completing that, you'll need to instantiate a couple of Java objects, just like you did in the `luceneindex.cfm` script.

```
indexReader = CreateObject("java",
    "org.apache.lucene.index.IndexReader");
searcher = CreateObject("java",
    "org.apache.lucene.search.IndexSearcher");
searcher = searcher.init(indexReader.open
    ("c:\cfusionmx\wwwroot\lucene\docsindex"));
analyzer = CreateObject("java",
    "org.apache.lucene.analysis.StopAnalyzer");
analyzer.init();
luceneQuery = CreateObject("java",
    "org.apache.lucene.search.Query");
queryParser = CreateObject("java",
    "org.apache.lucene.queryParser.QueryParser");
luceneQuery = queryParser.parse("cfx", "body",
    analyzer);
hits = CreateObject("java",
    "org.apache.lucene.search.Hits");
hits = searcher.search(luceneQuery);
```

Starting at the top, I create an `IndexReader` object, an `IndexSearcher` object, and then call the constructor of the `IndexSearcher` object using the

`ColdFusion init()` method. You'll notice that instead of calling the `init` method like this

```
searcher.init();
```

I use a different constructor, on whose argument is an `IndexReader` object. The `IndexReader` object is retrieved using a static method (a static method is a method that doesn't require the instantiation of an object before its use) called `"open"`, whose argument is the path to the index you want to search:

```
searcher.init(indexReader.open("c:\cfusion-
mx\wwwroot\lucene\docsindex"));
```

Again, I hardcoded the path to the index for the sake of simplicity; the source code at the end of this document uses a value from the attributes scope, enabling this script to be used as a CFML custom tag.

After instantiating the `IndexSearcher` object, I created a `StopAnalyzer` object, the same one that I used when creating and populating the index. (It's important to note that whatever Analyzer you use to create and populate your index, you must use the same type of Analyzer object to search your index. If you don't use the same type of object, you'll get inconsistent, if not invalid, results.)

Next, I create a `Query` object and a `QueryParser` object, instantiating the `Query` object by calling a static method of the `QueryParser` object `"parse()"`, which returns an instance of `org.apache.lucene.search.Query`. The `"parser()"` method's arguments are the keywords you want to search for, the property of the index you want to search, and the Analyzer object. Note that the CFML source code for this article substitutes the string `"cfx"` for a variable `"keyword"`, which comes from the attributes scope, again enabling this script to be used as a CFML custom tag. Finally, I create a `Hits` object, which is instantiated using the `IndexSearcher` objects' method `"search()"`.

You're almost done! At this point, the variable `"hits"` has been populated with the results of the Lucene search, so all we need to do is iterate over the variable

`hits`, extract the `Document`, and add the resulting values to a ColdFusion query, which you then return to the calling template. First things first: create a ColdFusion query using an array of columns:

```
localQuery = QueryNew("URL, TITLE, SUMMA-
RY");
```

Next, you'll get a `Document` object (the same kind of object you used when creating the index):

```
doc = CreateObject("java",
    "org.apache.lucene.document.Document");
```

and then iterate over the `hits` collection:

```
for (i=0; i LT hits.length(); i=i+1) {
    doc = hits.doc(javacast("int", i));
    QueryAddRow(localQuery);
    QuerySetCell(localQuery, "url",
        doc.get("url"), i+1);
    QuerySetCell(localQuery, "title",
        doc.get("title"), i+1);
    QuerySetCell(localQuery, "summary",
        doc.get("summary"), i+1);
}
```

There isn't much that's complicated here; the `"hits"` variable contains a method called `"length()"` that returns the number of elements in the collection (much like you can use the property `"recordcount"` when referring to a ColdFusion query). You retrieve the `Document` object from the `hits` collection using the `"doc()"` method. Notice that you must use the `"javacast()"` function to cast the variable `"i"` to the primitive Java type `"int"`. Finally, you use the `"get()"` method of the `Document` object to retrieve the string values of the URL, title, and summary fields, which you originally populated when creating the index. Also, you'll notice that the `hits` collection is a zero-based collection, while the ColdFusion query starts with 1, hence the use of the `"i + 1"`, in the `QuerySetCell()` function.

For the purposes of testing this script, you can use

```
<cfdump var="#localQuery#">
```

outside of the `<cfscript>` block to check

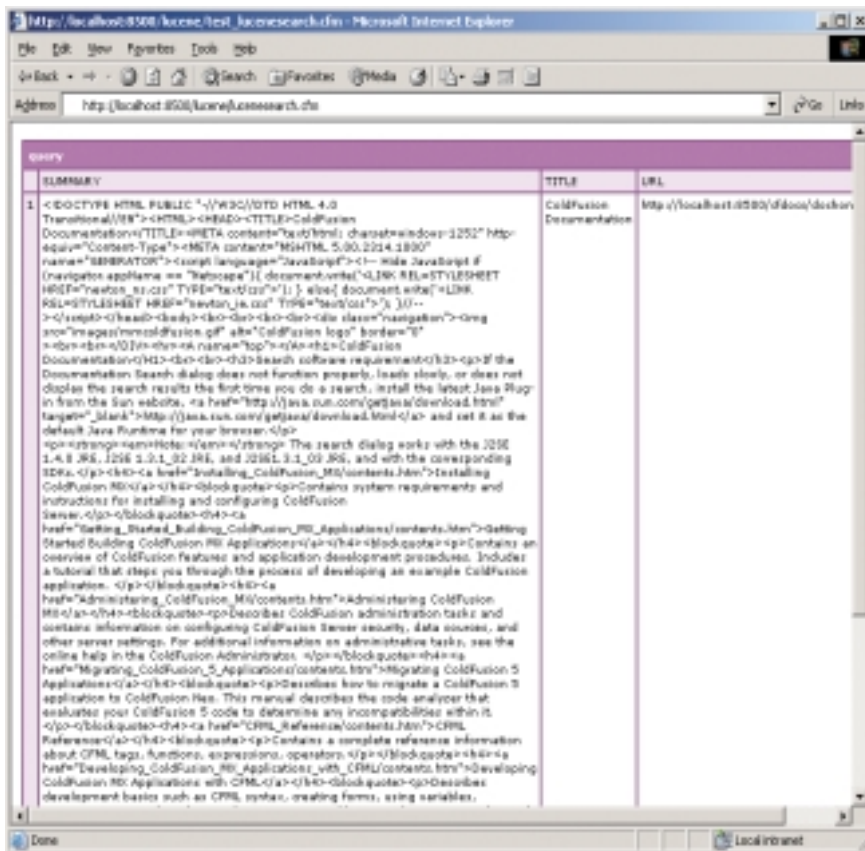


Figure 2: sample_search_result.bmp

the results of your search. If you've been following along and you've correctly created the Lucene index from the first section of this article, you should see something like Figure 2 after dumping the results of the query to the screen.

Optimizing a Lucene Index Using ColdFusion

Adding multiple documents to the Lucene index using the scripts you wrote above leaves the index in a fragmented state. For optimum performance, you'll want to optimize the Lucene index on a regular basis. The Lucene API makes this operation simple:

```
analyzer = CreateObject("java",
"org.apache.lucene.analysis.StopAnalyzer");
analyzer.init();
writer = CreateObject("java",
"org.apache.lucene.index.IndexWriter");
writer.init("c:\cfusionmx\wwwroot\lucene\doc-
sindex", analyzer, false);
writer.optimize();
```

This code should look relatively familiar to you. You start out by instantiating a `StopAnalyzer` object and an `IndexWriter` object and then you call the `IndexWriter` constructor, passing it the path to the Lucene index you want to optimize, along with the `Analyzer` object and a Boolean variable that determines whether or not to "create" an index. Obviously in this situation you don't want to create a new index; you simply want to optimize an existing one. Finally, you call the `optimize()` method of the `IndexWriter` object

```
writer.optimize();
```

which "... merges all segments together into a single segment, optimizing an index for search" according to the Lucene API documentation.

Conclusion

After you've read this article, I encourage you to download and explore the source code. I've exploded the short

scripts into two complete custom tags that you can use to closely mimic the behavior of `<cfsearch>` and `<cfindex>`, which use Verity internally. The `<cf_luceneindex>` tag uses functions downloaded from www.cflib.org to recursively crawl a directory, indexing each file along the way. Additionally, the `<cf_luceneindex>` tag gives you the ability to optimize a Lucene index with only a single line of code:

```
<cf_luceneindex action="optimize"
indexpath="c:\pathtoyourindex\">
```

The `<cf_lucenesearch>` tag gives you the ability to search a Lucene index.

...

Check back next month for the second part of this series on Lucene and ColdFusion: Writing a Java-based CFX Tag.

Resources

- **Jakarta Lucene:**
<http://jakarta.apache.org/lucene>
- **Jakarta Lucene Downloads:**
<http://jakarta.apache.org/builds/jakarta-lucene/release/v1.2>
- **Search-Enable Your Application with Lucene:** www.sys-con.com/java/article.cfm?id=1777
- **ColdFusion MX Documentation:**
<http://livedocs.macromedia.com/cfmxdocs>
- **ColdFusion CFScript Documentation:**
http://livedocs.macromedia.com/cfmxdocs/Developing_ColdFusion_MX_Applications_with_CFML/CFScript.jsp
- **CFLib.org: FileRead.cfm:**
www.cflib.org/udf.cfm?ID=417
- **CFLib.org: DirectoryList.cfm:**
www.cflib.org/udf.cfm?ID=615

About the Author

Aaron Johnson is a senior software architect for Mindseye, Inc., and has been developing large-scale Web sites for companies like FAO Schwarz, FootJoy, and Macromedia using ColdFusion, ASP, C#, and Java since 1996. He is a Certified ColdFusion Developer and a Microsoft Certified Systems Engineer. You can find out more about Aaron via his blog (<http://cephas.net/blog>).

ajohnson@cephas.net

ColdFusion User Groups

For more information go to...

<http://www.macromedia.com/cfusion/usergroups>

New England

New Hampshire
Northern N.E. MMUG
www.mmug.info

Massachusetts
Boston, MA CFUG
www.cfugboston.org

Rhode Island
Providence, RI CFUG
www.ricfug.com

Vermont, Montpelier
Vermont CFUG
www.mtbytes.com/dfug/index.htm

Midatlantic

New Jersey, Raritan
Central New Jersey CFUG
www.freecfm.com/cjcfug/index.cfm

New York
Albany, NY CFUG
www.anycfug.org

New York
Long Island, NY CFUG
www.licfug.org

New York
New York, NY CFUG
www.nycfug.org

New York
Rochester, NY CFUG
www.roch-cfug.org

New York
Syracuse, NY CFUG
www.cfugcny.org

Pennsylvania, Harrisburg
Central Penn CFUG
www.centralpenncfug.org

Pennsylvania
Philadelphia, PA CFUG
www.pacfug.org

Pennsylvania
Pittsburgh, PA CFUG
www.orbwave.com/pgchcfug

Pennsylvania
State College, PA CFUG
www.cfug-sc.org

Southern

Alabama
Birmingham, AL CFUG
www.bcfug.org

Alabama
Huntsville, AL CFUG
www.nacfug.com

Delaware, Dover
Delaware CFUG
www.decfug.org

Delmarva, Dover
Delmarva CFUG
www.delmarva-cfug.org

Florida
Gainesville, FL CFUG
<http://plaza.ufl.edu/aktas>

Florida
Orlando, FL CFUG
www.cforlando.com

Florida
Tallahassee, FL CFUG
www.tcfug.com

Florida
Tampa, FL CFUG
www.tbcfug.org

South Florida
South Florida CFUG
www.cfug-sfl.org

Georgia, Atlanta
Atlanta, GA CFUG
www.acfug.org

Georgia, Atlanta
Georgia CFUG
www.cfugorama.com

Georgia
Columbus, GA CFUG
www.vcfug.org

Kentucky
Louisville, KY CFUG
www.loulexcfug.com

Louisiana
New Orleans, LA CFUG
www.nocfug.org

Maryland
Annapolis, MD CFUG
www.ancfug.com

Maryland
Baltimore, MD CFUG
www.cfugorama.com

Maryland
Broadneck H. S. CFUG
www.cfug.broadneck.org

Maryland, Bethesda
Maryland CFUG
www.cfug-md.org

Maryland
California, MD CFUG
<http://smdcfug.org>

North Carolina
Charlotte, NC CFUG
www.charlotte-cfug.org

North Carolina
Fayetteville, NC CFUG
www.schoolink.net/fcfug/

North Carolina
Raleigh, NC CFUG
www.ccfug.org

Oklahoma
Oklahoma City, OK CFUG
<http://idgweb4.ouhsc.edu/cfug>

Oklahoma
Tulsa, OK MMUG
www.tulsacfug.org

Tennessee
Memphis, TN CFUG
<http://cfug.dotlogix.com>

Tennessee
Nashville, TN CFUG
www.ncfug.org

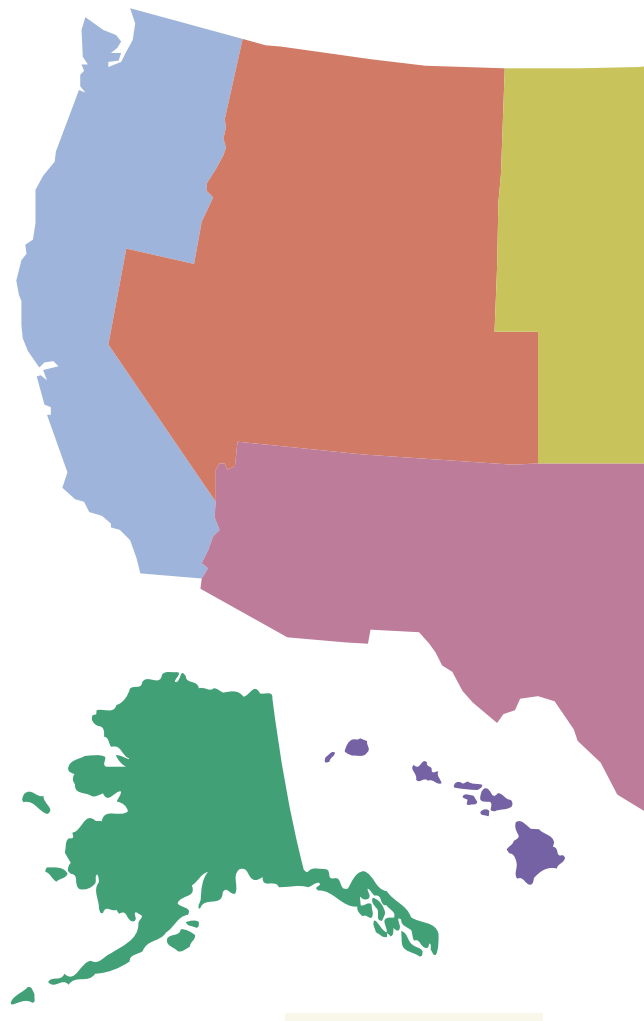
Virginia
Hampton Roads CFUG
www.hrcfug.org

Virginia
Northern Virginia CFUG
www.cfugorama.com

Virginia
Richmond, VA CFUG
<http://richmond-cfug.btgi.net>

Virginia, Roanoke
Blue Ridge MMUG
www.brmug.com

Washington D.C.
Washington, D.C. CFUG
www.cfugorama.com



Midwest

Northern Colorado
Northern Colorado CFUG
www.nccfug.com

Colorado Hamilton M.S. CFUG
<http://hamilton.dpsk12.org/teachers/team-c/intro.html>

Illinois, Champaign
East Central Illinois CFUG
www.ecicfug.org

Illinois, Chicago
Chicago, IL CFUG
www.cfugorama.com

Indiana
Indianapolis, IN CFUG
www.hoosierfusion.com

Indiana, South Bend
Northern Indiana CFUG
www.ninmug.org

Iowa
Des Moines, IA CFUG
www.hungrycow.com/cfug/

Michigan, Dearborn Mmaniacs
CFUG <http://ciwebstudio.com/mmania/mmania.htm>

Michigan, East-Lansing
Mid Michigan CFUG
www.coldfusion.org/pages/index.cfm

Minnesota, Minneapolis
Twin Cities CFUG
www.colderfusion.com

Rockies

Montana, Helena
Montana CFUG
<http://montanacfug.site-o-matic.com>

Nevada
Las Vegas, NV CFUG
www.snfcug.com

Utah
Salt Lake City, UT CFUG
www.slcfug.org

Wyoming, Jackson
Wyoming MMUG
www.wyfcug.org

West Coast

California
Bay Area CFUG
www.bacfcug.net

California, Fresno
Central California CFUG
www.cccfcug.com

California, Inland Empire
Inland Empire CFUG
www.sccfcug.org

California
Los Angeles CFUG
www.sccfcug.org

California
Orange County CFUG
www.sccfcug.org

California
Sacramento, CA CFUG
www.saccfcug.org

California
San Diego, CA CFUG
www.sdcfcug.org

Southern California
Southern California CFUG
www.sccfcug.org

Oregon
Eugene, OR CFUG
www.EugeneCFUG.org

Oregon
Portland, OR CFUG
www.pdxcfug.org

Alaska

Alaska
Anchorage, AK CFUG
www.akcfug.org

Hawaii

Hawaii, Honolulu
Hawaii CFUG
<http://cfhawaii.org>

Southwest

Missouri
Kansas City, MO CFUG
www.kcfusion.org

Missouri
St. Louis, MO CFUG
www.psiwebstudio.com/cfug

Nebraska
Omaha, NE CFUG
www.necfug.com

Ohio, Columbus
Ohio Area CFUG
www.oacfcug.org

Ohio
Mid Ohio Valley MMUG
www.movcfug.org

Wisconsin
Milwaukee, WI MMUG
www.metromilwaukee.com/usr/cfug

Arizona
Phoenix, AZ CFUG
www.azcfug.com

Arizona
Tucson, AZ CFUG
www.tucsoncfug.org

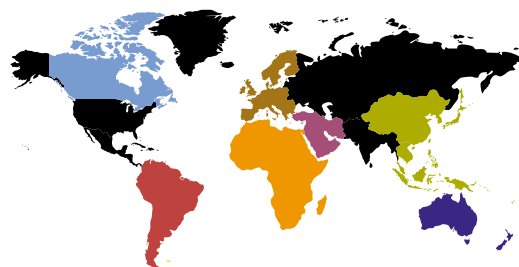
Texas
Austin, TX CFUG
<http://cftexas.net>

Texas
Dallas, TX CFUG
www.dfwcfug.org

Texas
San Antonio, TX CFUG
<http://www.samcfug.org>

About CFUGs

ColdFusion User Groups provide a forum of support and technology to Web professionals of all levels and professions. Whether you're a designer, seasoned developer, or just starting out - ColdFusion User Groups strengthen community, increase networking, unveil the latest technology innovations, and reveal the techniques that turn novices into experts, and experts into gurus.



Africa

South Africa, Cape Town
Cape Town, South Africa CFUG
www.coldfusion.org.za

South Africa, Johannesburg
Johannesburg, South Africa CFUG
www.coldfusion.org.za

Central Europe, Munich
Central Europe CFUG
www.cfug.de

Finland, Helsinki
Finland CFUG
www.cfug-fi.org

France, Valbonne
France CFUG
www.cfug.fr.st

Germany, Frankfurt
Frankfurt CFUG
www.coldfusion-frankfurt.de

Ireland
Belfast, Ireland CFUG
www.cfug.ie

Ireland
Cork, Ireland CFUG
<http://viewmylist.com/cork>

Italy, Bologna
Italy CFUG
www.ingenium-mmug.org

Sweden
Gothenburg, Sweden CFUG
www.cfug-se.org

Switzerland
Martin Bülmann, Switzerland CFUG
www.cfug.ch

United Kingdom, London
UK CFUG
www.ukcfug.org

United Kingdom
Northern England CFUG
www.cfug.org.uk

Asia

Malaysia, Kuala Lumpur
Malaysia CFUG
www.coldfusioner.com

Thailand
Bangkok, Thailand CFUG
<http://thaicfug.tei.or.th>

Japan, Urayasu-city Japan CFUG
<http://cfusion.itfrontier.co.jp/jcfug/jcfug.cfm>

Australia

Australia-Melbourne
Australian CFUGs
www.cfug.org.au

Canada

Canada
Edmonton, AB CFUG
<http://edmonton.cfug.ca>

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Montreal, QC CFUG
www.kopanas.com/cfugmontreal

Canada
Ottawa, ON CFUG
www.cfugottawa.com

Canada, Ottawa (HS group)
Ecole Secondary CFUG
www.escgarneau.com/gug

Canada
Toronto, ON CFUG
www.cfugtoreto.org

Canada
Vancouver, BC CFUG
www.cfug-vancouver.com

Canada
Vancouver Island CFUG
www.cfug-vancouverisland.com

Middle East

Pakistan Edu, Lahore Cantt
Pakistan Educational CFUG
www.cfugpakistan.org

Pakistan, Lahore
Pakistan CFUG
www.cfugpakistan.org

Saudi Arabia, Riyadh
CFUG Saudia
www.cfugsaudia.org

Turkey, Izmer
Turkey CFUG
www.cftr.net

Europe

Belgium, Brussels
Belgium CFUG
www.cfug.be

South America

Brazil
Rio de Janeiro CFUG
www.cfugrio.com.br

Macromedia ColdFusion MX Now 'Java Verified' for Portability Across J2EE Application Servers (San Francisco) – Macromedia has announced that Macromedia ColdFusion MX has achieved "Java Verified" status under the Sun Microsystems Java Verification Program. The Java Verification Program is designed to identify enterprise applications developed with Java 2 Platform, Enterprise Edition (J2EE) technology that are intended to be portable across different implementations of J2EE.

"One of our key objectives in bringing Macromedia ColdFusion MX to the J2EE platform is to give customers more



flexible deployment options," said Norm Meyrowitz, president of products, Macromedia. "Passing the Java Application Verification

Kit test suite ensures customers that they can enjoy the proven productivity of the ColdFusion environment on multiple J2EE application servers."

"The Java Verification Program enables vendors to demonstrate their commitment to standards, to minimize development costs, and maximize ROI," said Mark Bauhaus, vice president of Java Web services at Sun. "By attaining Java Verified status for ColdFusion MX, Macromedia is helping developers take full advantage of the power of the J2EE platform, delivering the true promise of 'Write Once, Run Anywhere.'"

Announcing MAX: the 2003 Macromedia User Conference

(San Francisco) – Macromedia has announced the Macromedia MAX 2003 conference, scheduled for November 18-21 at the Salt Palace Convention Center in Salt Lake City, Utah. MAX, a new conference that combines DevCon and UCON, is Macromedia's annual professional conference for Macromedia developers and designers.

This year, the Macromedia conference has a new name and a new venue, reflecting an exciting evolution for the conference and the company's customers. The name change is designed to express their goal – that MAX appeals to the broadest range of Macromedia customers. Stay tuned for details.



Toshiba Information Systems Licenses Macromedia Flash to Create Multimedia Platform for Devices (San Francisco and Tokyo) – Toshiba Information Systems (Japan) and Macromedia have announced a licensing agreement to integrate Macromedia Flash support into Toshiba's embedded software offerings. By integrating Macromedia Flash Player, the leading rich client on the Internet, with the company's design, engineering, and integration offerings, Toshiba Information Systems will create a comprehensive multimedia platform for devices.

"Through Macromedia Flash Player SDK licensing, companies like Toshiba Information Systems (Japan) can deliver unique solutions that build on the power of our rich client," said Peter



Meechan, vice president, Macromedia. "With Macromedia Flash as a core technology of its offerings, Toshiba Information

Systems will be able to develop user interfaces faster and more cost-effectively for their OEM projects."

"Macromedia Flash was the best choice for Toshiba Information Systems because of its ability to effectively build user interfaces for various device form factors," said Keiichi Ishikawa, general manager, embedded systems solution division, Toshiba Information Systems.

ACCESS Licenses Macromedia Flash SDK to Deliver Embedded Browser for Appliances (San Francisco and Tokyo) – Macromedia, Inc., and ACCESS Co., Ltd., have

announced a licensing agreement to integrate Macromedia Flash support into the ACCESS NetFront browser. The NetFront browser is one of the most popular and broadly deployed embedded browsers in the world. Macromedia Flash is the leading rich client on the Internet.

"Through Macromedia Flash Player SDK licensing, companies like ACCESS can deliver an embedded browser that includes all the leading Web standards," said Peter Meechan, vice president, Macromedia. "By integrating Macromedia Flash into its browser, ACCESS immediately enables developers to create rich content and user interfaces for its appliance customers."

"Macromedia Flash is the obvious choice for our NetFront browser because of

its ability to effectively deliver rich content in a compact format," said Toru Arakawa, president and CEO, ACCESS Co., Ltd. "By choosing an industry standard such as Macromedia Flash, we now have access to not only a huge developer community, but powerful authoring software, and a wide array of content types that we can deliver in our embedded browser platform."

Cast Your Vote for Your Favorite CF Products/Services

Vote for your favorite ColdFusion products/services in the 2003 **ColdFusion Developer's Journal** Readers' Choice Awards competition now in full swing at www.sys-con.com/coldfusion/readerschoice2003. Widely referred to as "The Oscars of the Software Industry," the Readers' Choice Awards program has become the most respected industry competition of its kind. Voting began on March 1, and will continue until August 30, 2003. Winners will be announced at Web Services Edge 2003 West, in Santa Clara, CA, September 30-October 2.



Feedback...



Getting into HomeSite+ – Bravo! [CFDJ, Volume 5, issue 6]

I'd just like to pass on my thanks for Charlie Arehart's excellent article in the June issue of **ColdFusion Developer's Journal**: "Getting into HomeSite+." In the two years of my subscription, this was the most beneficial article I've read.

Being quite disappointed with my Studio MX upgrade when it came to the IDE (and migration issues with SiteMinder in MX), I've gone back to ColdFusion 5. This really saved the day for me. Obviously I was surprised to learn it was on my CD all this time.

– Gene Marshall

gene_marshall@hp.com

MACROMEDIA

www.macromedia.com/go/drk3/

INTERMEDIA.NET

www.intermedia.net