

COLD FUSION Developer's Journal

ColdFusionJournal.com

August 2005 Volume:7 Issue:8

CF Everywhere

PART 3 14

Got Flash? 18

No-see-ums 20

**New Debugging
Features of CFMX7** 28

**ColdFusion and Crystal
Enterprise Integrations** 32

What's an Object? 36

**Profiling CFML at the
Tag Level, Finally!** 46

Event-Model Programming

with



**See what your code
is doing PG 24**

PLUS...

An Overlooked Great New Feature	5
The Next Programming Models	7
ColdFusion User Groups	44



One little box. A whole lot of power.

Put it to work for you.
The shortest distance between you
and powerful web applications.

Full speed ahead. The release of Macromedia ColdFusion MX 7 is changing the whole game. This groundbreaking release introduces new features to help address your daily development challenges. These features include:



› **Structured business reporting? Check. Printable web content? Check.**

ColdFusion MX 7 provides a structured business reporting solution that will have you creating detailed business reports for anyone who needs them. You can also dynamically transform any web content into high-quality, printable documents with ease. The days of needing a third-party application to generate dynamic reports are going, going, gone.



› **Make rapid J2EE development a reality.**

So, you're heavily invested in J2EE but would love to complete projects in less time? ColdFusion MX 7 is your answer: It delivers RAD productivity on top of the power and scalability of J2EE and deploys onto standard J2EE server environments.



› **New mobile applications are a go.**

Innovative new features enable ColdFusion MX 7 applications to reach beyond the web. So you can rapidly create new applications, or extend existing ones, to connect with mobile phones and IM clients using a variety of protocols. The new world of mobile communications is exciting, and this your invitation to the party.

To learn more or take ColdFusion MX 7 for a test drive, go to:
macromedia.com/go/cfmx7_demo

Brad Lawryk: "Before MX Kollection, I 'thought' I was doing a **pretty good job**, and now ...

Everyone tells me I'm doing a **brilliant** job!"

If you own Dreamweaver,

MX Kollection is a must!

List Database Information

- Create dynamic lists
- Order records within a list
- Insert, Edit, Delete field in list
- Automatic navigation bars
- Automatic list filter
- Sort column by clicking on the header
- Automatic row counter
- Delete multiple records directly from list
- Create master/detail lists

Manage Recordsets Visually

- Create, Edit recordsets visually
- Visually add tables to query
- Contextual menu to refresh fields
- Large query management Zoom In/Out
- JOINS between tables
- Define and apply complex conditions
- Automatically generate SQL conditions
- Add GROUP BY for aggregated fields
- Extract information from multiple tables
- Optimized SQL generation
- Smart SQL queries for list filters
- Automated database introspection

Rapid HTML Form Creation

- Generate Insert Record Form
- Generate Update Record Form
- Insert, Edit or Delete form field
- Redirect to page after form submit
- Table and CSS form generation

Form Validation and Error Handling

- Validate form fields
- Rich formats library
- Allow custom validation formats
- Error handling
- Preserve submitted values on error

Upload Files and Images

- File Upload
- Image Upload
- Resize and sharpen image on upload
- Show Dynamic Image
- Download Uploaded File
- Delete file from specified folder
- Show If File Exists on Server

Send E-mails

- Send e-mail after form submit

Online HTML Editor

- Edit HTML content visually
- Use your own CSS styles
- Upload and manage server images
- Format tables and align images

Enhanced HTML Form Controls

- Date Picker
- Dependent Drop-downs
- Editable Drop-down (Combo-box)
- Masked Textfield
- Numeric Textfield
- Dynamic Search
- n...1 Dependent field
- Smart Date

Security & User Authentication

- Login form generator
- Login with User Levels
- Remember me feature
- Encrypted password
- Restrict access to page
- Generate password with fixed length
- Update user password
- Send password by e-mail

DREAMWEAVER PRODUCTIVITY TOOL FOR DYNAMIC WEBSITE DEVELOPMENT

Download a trial version here - <http://interaktonline.com/go/MXKollection/>

Find out how easy is to use our tools - **save 30%** by using the coupon in the right.

Caution: InterAKT extensions are known to cause a high increase in productivity. For those of you who can't deal with the free time please continue to hand-code or refer to our competitors for help.



\$100 coupon - 199920002001

Interakt

macromedia
approved

macromedia
ALLIANCE PARTNER

<http://www.interaktonline.com/>



Dedicated Server Packages Starting at \$189/mo.

All dedicated servers include:

- ▶ FREE STATS SOFTWARE!
- ▶ No long term commitments!
- ▶ FREE SQL server access!
- ▶ FREE MAIL SOFTWARE!
- ▶ Fair and simple pricing!
- ▶ Optional server maintenance!

As one of the premier ColdFusion hosting community leaders, CFDynamics is constantly looking for ways to provide **better service** to ensure your satisfaction. Not only do we offer the **finest in shared hosting plans**, but we now offer the **finest in 100% dedicated server plans**! Now you can afford the freedom of having your own dedicated server!

When your needs have outgrown shared hosting look to CFDynamics for **total freedom**. With dedicated server packages they're not offering an oxymoron; "virtually private" or "virtually dedicated" is **NEITHER** private nor dedicated. CFDynamics offers a solution that is **100% completely dedicated**. They don't play games with the fake stuff; CFDynamics only offers the real deal. Real Service. Real Satisfaction. Real Value.

Real Freedom.



PaperThin Partner



Visit us online or call to order!

WWW.CFDYNAMICS.COM

1 - 866 - 233 - 9626
866 - CFDYNAMICS

editorial advisory board

Jeremy Allaire, *founder emeritus, macromedia, inc.*
Charlie Arehart, *CTO, new atlanta communications*
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**

Simon Horwith simon@sys-con.com

technical editor

Raymond Camden raymond@sys-con.com

editor

Nancy Valentine nancy@sys-con.com

associate editor

Seta Papazian seta@sys-con.com

research editor

Bahadir Karuv, PhD bahadir@sys-con.com

production**production consultant**

Jim Morgan jim@sys-con.com

lead designer

Abraham Addo abraham@sys-con.com

art director

Alex Botero alex@sys-con.com

associate art directors

Louis F. Cuffari louis@sys-con.com

Tami Beatty tami@sys-con.com

Andrea Boden andrea@sys-con.com

contributors to this issue

Charlie Arehart, Phil Cruz, Jason Heaslet, Simon Horwith,
Jeff Houser, Kevin Kazmierczak, Nik Molnar,
Jared Rypka-Hauer, Simeon Simeonov, Matt Woodward

editorial offices**SYS-CON MEDIA**

135 Chestnut Ridge Rd., Montvale, NJ 07645

Telephone: 201 802-3000 Fax: 201 782-9638

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 © 2005 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 LIST RENTAL INFORMATION:

Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com

Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

For promotional reprints, contact reprint
coordinator Kristin Kuhnle, kristin@sys-con.com.
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.

An Overlooked Great New Feature



By Simon Horwith

As a developer I tend to focus my attention on learning the tools that best meet my immediate project needs. As a Macromedia Certified Instructor I am in the position of having to learn all of the new features and tools, even those that


I may not personally use very often in development, in order to teach them to my students. My role as a developer is unique in that I spend the majority of my time dealing with application architecture and business objects, whereas the average developer focuses more on the presentation tier on a day-to-day basis. It's very important that I am able to teach Flash forms, PDF and FlashPaper generation, the report builder IDE, and new Dreamweaver features to name a few to people who really need and rely on them to get the job done when they leave the classroom and go back to the office.

I've spent a lot of time recently focusing on these new features, and have been pleasantly surprised to find that I really enjoy many of them much more than I thought I would. In this month's editorial I thought I'd share my insights regarding a new tool that seems to have received little attention even though it's terribly useful and reflects Macromedia's efforts to step in the right direction. It only takes 5 minutes for a developer to explore and learn how to use, so there's no excuse not to be aware of this little gem.

The feature I'm talking about is the new ColdFusion Component Query functionality in Dreamweaver. If you haven't already installed the CFMX 7 Dreamweaver extensions, you have to install those in order to use this new functionality. Once the extensions are installed, you're ready to use the new CFC Query functionality.

Dreamweaver MX has a query builder – you access it by going to the “Application” panel, clicking on the “bindings” tab, and then clicking the plus sign button and selecting recordset. The CFMX 7 Dreamweaver extensions add an enhancement to this recordset builder. If you launch the recordset builder while a CFC is the active file in Dreamweaver, the recordset builder allows you to select a function from the current CFC and have the query code written

into that function. There's also a “New Function” button there that allows you to specify a name for a new function to create. If you click this button and assign a name, then finish creating your query and submit it, the recordset builder will create a function with the name you specified. This function will default to not allowing output and to having a “query” return type. The function body will declare a private variable to hold your result set, contain the query you defined, and return that variable. This makes the recordset builder much more useful than it used to be – for the first time we can have the IDE create a query inside a method, which has been written the “right” way, rather than just writing a <CF-QUERY> block at the top of our file. It's a major step in the right direction in that it encourages developers to put their queries in CFC methods rather than browsable .cfm files. The code itself also encourages proper use of component variable scoping. Macromedia deserves a round of applause for extending the existing functionality in a way that encourages best practices.

The July issue went out fairly late due to the timing of the CFUnited Conference and the CF 10th birthday party in Newton – both of which I felt deserved a write-up. So, for this month's issue I played catch-up and am happy to say we're getting back on schedule – my apologies to any of you who were inconvenienced by the delay in getting the July issue into your hands. Our focus this month is on debugging and we've got many great articles focused on debugging techniques. Expect to see a few more debugging related articles that didn't make it into this issue in next month's issue of *CFDJ*. In addition to the debugging articles we also have some great articles on other topics in this issue including, among other things, an insightful article on programming models by Simeon Simeonov – the lead engineer on many versions of ColdFusion. Enjoy! 

About the Author

Simon Horwith is the editor-in-chief of ColdFusion Developer's Journal. Simon is a Macromedia Certified Master Instructor and a member of Team Macromedia. He has also been a contributing author of several books and technical papers. You can read his blog at www.horwith.com

simon@horwith.com

Complete source code and asset management in Dreamweaver MX—now possible with Surround SCM.

Dreamweaver users know a beautiful Web-based product is only skin deep. Underneath, it's a tangle of hundreds or thousands of ever changing source files. Without a good development process and strong tools, bad things happen. Surround SCM can help.

Surround SCM lets you...

Track multiple versions of your source files and easily compare and merge source code changes.

Check out files for exclusive use or work in private workspaces when collaborating on a team.

Automatically notify team members of changes to source files—push changes through your organization.

View complete audit trails of which files changed, why, and by whom.

Associate source code changes with feature requests, defects or change requests (requires additional purchase of TestTrack Pro).

Remotely access your source code repository from Dreamweaver MX.

Surround SCM adds flexible source code and digital asset control, powerful version control, and secure remote file access to Dreamweaver MX. Whether you are a team of one or one hundred, Surround SCM makes it easier to manage your source files, letting you focus on creating beautiful Web-based products.

Features:

Complete source code and digital asset control with private workspaces, automatic merging, role-based security and more.

IDE integration with Dreamweaver MX, JBuilder, Visual Studio, and other leading Web development tools.

Fast and secure remote access to your source files—work from anywhere.

Advanced branching and email notifications put you in complete control of your process.

External application triggers let you integrate Surround SCM into your Web site and product development processes.

Support for comprehensive issue management with TestTrack Pro—link changes to change requests, bug reports, feature requests and more.

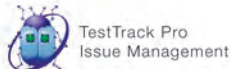
Scalable and reliable cross-platform, client/server solution supports Windows, Linux, Solaris, and Mac OS X.

Want to learn more?

Visit www.seapine.com/webdev to discover how Surround SCM can save you time and headaches. Be sure to download our white paper, **Change Management and Dreamweaver**, and learn why change management is the new must-have tool for web development.



www.seapine.com/webdev
1-888-683-6456



president & ceo

Fuat Kircaali fuat@sys-con.com

vp, business development

Grisha Davida grisha@sys-con.com

group publisher

Jeremy Geelan jeremy@sys-con.com

advertising

senior vp, sales & marketing

Carmen Gonzalez carmen@sys-con.com

vp, sales & marketing

Miles Silverman miles@sys-con.com

advertising director

Robyn Forma robyn@sys-con.com

advertising manager

Megan Mussa megan@sys-con.com

sales & marketing director

Dennis Leavey dennis@sys-con.com

associate sales manager

Dorothy Gil dorothy@sys-con.com

sys-con events

president, events

Grisha Davida grisha@sys-con.com

national sales manager

Jim Hanchrow jimh@sys-con.com

customer relations

circulation service coordinators

Edna Earle Russell edna@sys-con.com

Linda Lipton linda@sys-con.com

manager, jdl store

Brunilda Staropoli bruni@sys-con.com

sys-con.com

vp, information systems

Robert Diamond robert@sys-con.com

web designers

Stephen Kilmurray stephen@sys-con.com

Vincent Santaiti vincent@sys-con.com

online editor

Roger Strukhoff roger@sys-con.com

accounting

financial analyst

Joan LaRose joan@sys-con.com

accounts payable

Betty White betty@sys-con.com

accounts receivable

Gail Naples gain@sys-con.com

subscriptions

Subscribe@sys-con.com

Call 1-888-303-5282

For subscriptions and requests for bulk orders, please send your letters to Subscription Department

Cover Price: \$8.99/issue

Domestic: \$89.99/yr (12 issues)

Canada/Mexico: \$99.99/yr

All other countries \$129.99/yr

(U.S. Banks or Money Orders)

Back issues: \$12 U.S. \$15 all others

The Next Programming Models



By Simeon Simeonov

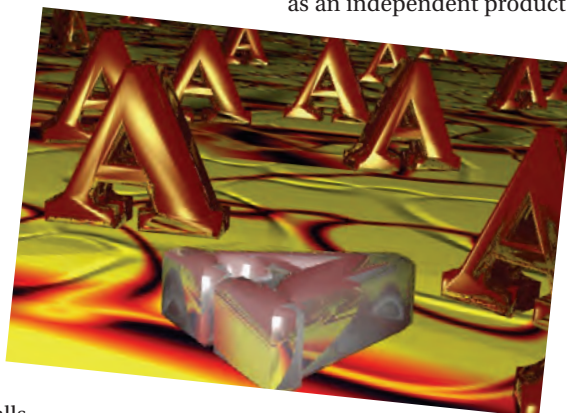
I've been around software for 20 years now. Looking back, I have mixed feelings about the progress we've made. The end results have been amazing, but the process of building software hasn't fundamentally changed since the 80s. In fact, I see us making some of the same mistakes over and over again. One of the common anti-patterns is over-relying on tools and frameworks instead of inventing new programming models.

Layers of abstraction are fundamental to software. Some layers are defined through programming models, e.g., machine language, assembly language, 3GLs, and JSP. Others are defined through a combination of tools and frameworks, e.g., MFC and Visual Studio on top of C++. There is a limit to how high we can raise a level of abstraction through tools and frameworks alone. At some point, a new programming model is the best way forward. Here are some examples. CASE tools on top of 3GLs never achieved the success of 4GLs. Tools and frameworks for Web application development, from CGI + your favorite language to WebObjects to HAHT, were demolished in the market by page-based Web application development models such as ColdFusion, PHP, JSP, and ASP. What we have seen time and time again is that it is often better to come up with a new programming model than to keep pushing an existing model forward by throwing ever more advanced tools and sophisticated frameworks on top. Think of a building. Programming models are the floors. Tools and frameworks are the walls. To build a tall building you need to strike a balance between the number of floors and the height of walls.

Beyond a certain point, an extra foot of room height adds very little to the quality of a room but increases the cost of the building substantially.

When should one create a new programming model as opposed to going with a framework and/or tool leverage? What is a programming model anyway? Tough questions, both of them... The first is impossible to answer perfectly or quickly. The second question is a little easier because you can often recognize a new programming model when you see it. One key observation is that you don't necessarily need a new programming language, as JSP and ASP demonstrate. Sometimes it is sufficient to create a domain-specific template or wrapper into which existing programming models fit. Also, new programming models may each come with its own set of frameworks and tools.

I have some first-hand experience in creating new programming models. At Allaire we defined the page-based Web application development model with ColdFusion and later helped the Java community get its act together with JSP and tag libraries. Later at Macromedia, we defined the model for building rich Internet applications (RIAs) with Flash & Flex, something Microsoft will try to catch up to with Avalon in Longhorn. In between, we did a lot of work on SOA programming models, though with the burst of the tech bubble we decided not to ship this as an independent product



but instead contributed the ideas to Apache and to existing products internally.

Here are some thoughts on two programming models that I hope we can significantly improve in the next few years.

Rich Internet Applications

You have to admit, we did take a step back in usability with the Web. We can build easily accessible applications quickly but wouldn't it be nice if we didn't have to go through 10 screens to make an airline reservation? What we need are applications that have the deployment characteristics of browser-based applications, but that have equivalent power and more interactivity than desktop applications. That's what RIAs are all about. They bring complexity on two levels. First, computing happens on both the client and the server over a potentially unreliable WAN. Second, they aim to deliver highly interactive user experiences (UEs). Don't blow that second requirement off. Research clearly shows that users respond better to these types of interfaces. Who wants to use old-style Web maps when you can go with Google maps or the Flash-based AbMap?

A good RIA programming model will protect developers from the details of location, i.e., the tasks associated with synchronizing data shared between the front and back end, invoking back-end services, dealing with online/offline operation, etc. It will also have an advanced rendering engine, preferably one that is cross-platform and device independent, and a presentation model that hides much of the hassle of resolution, screen orientation, and internationalization. I'm very biased in making this claim but the only commercially sound approach to RIAs nowadays is with Macromedia Flash and, better, with

Flash and Flex together. Microsoft Avalon is the closest competing technology. It has yet to ship. AJAX, contrary to what many believe, has been around since at least 1998 but it didn't have a cool acronym. AJAX + DHTML offer an alternative but there has been little success moving from specific cool apps to a generalized programming model. Java doesn't cut it, primarily for UE reasons. There is plenty of room for improvement.


Don't forget mobile applications. More than PC-based applications, they really need a makeover and there are a lot of dollars at stake. Microbrowsers are trying to find ways to bring AJAX + DHTML + WAP to devices. Java has deep market penetration but poor UE. Brew has the best device integration but is similar to Java on the UE front. Flash Lite is gaining traction here because of the great UE it enables.

Composite Applications

There is no question about it – you can build composite applications using Java, .NET, or any other programming language for that matter, just as you can build Web apps using C++ and write admin scripts in Cobol. Why would you, though? One of the cornerstones of SOA is that services can be implemented using anything. That's great, but traditional approaches for writing the glue code between services leave a lot to be desired. What we need are deeper and more declarative mechanisms for putting services together. BPEL and the WS-* standards are both too much and not enough. Do this: print all the specs and stack them together. Now, think about how much *ad hoc* work you had to do to build, deploy, and operate your last composite app. Do you feel comfortable with where the industry is going?

Building, deploying and operating composite applications requires dealing with issues such as policy definition and enforcement, service evolution/versioning, system/deployment architectures, and post-deployment management and monitoring. This goes into what traditionally has been considered to be the IT sphere of influence, often a taboo area for development. However, I deeply believe that a winning programming model has to begin to address these issues. Just consider some of the complexities. How do you maintain applications over time as services evolve? How do you debug them? When something doesn't work right in a production application how do you track down the root cause? If you don't address these issues during the architecture and design phases you're in for pain down the road.

Talk like this takes us into the realm of utility computing, whatever that means (definitions still vary). Perhaps this is what's necessary to make building, testing, deploying, and operating composite applications easy. There is plenty of innovation in this space. Unfortunately, much of it is in the form of add-on products as opposed to a comprehensive programming model-driven approach to the problem. This is bad news for customers who run the risk of experiencing the dubious pleasures of vendor lock-in.

My personal wish list for innovative programming models is longer, for example, covering ultra-scalable applications that run on large clusters (≥ 32 nodes). I even think that we can do a lot better with the decade-old Web application model. Just look at some of the work going on with Ruby on Rails and with the new capabilities in CF 7.0. As both a technologist and an investor, I'm excited about the future. 

About the Author

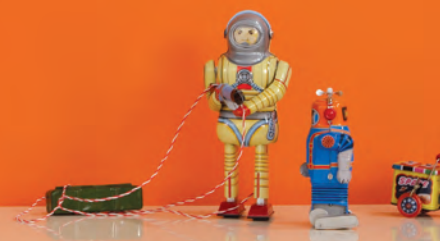
Simeon (Sim) Simeonov used to be chief architect at Allaire/Macromedia where he shipped more products than he can remember on a summer day. Now he is a technology investor at Polaris Venture Partners in Boston. He still codes on weekends.

sim@polarisventures.com

“I’m very biased in making this claim but the only commercially sound approach to RIAs nowadays is with Macromedia Flash and, better, with Flash and Flex together”



WEB DEV GURU seeks an Integrated Software Suite that speaks my language (XHTML) and won't cramp my style. Must play well with XML, CSS and others. I'm not superficial, but I like a nice code view. Clutter isn't cute.



Different people. Different needs. One suite solution.

With the latest versions of Macromedia Dreamweaver®, Flash® Professional, Fireworks®, Contribute™, and FlashPaper™, the new Studio 8 is quite a catch. To meet Studio 8 and find all the web design and development tools you need, visit www.macromedia.com/go/8_studio8.

macromedia®
STUDIO 8

How to Debug Your Applications

Problem solving and techniques for your debugging adventures



By Jeff Houser

I just spent four hours debugging an error for a client. The client is an application service provider, and they developed an administrator for internal use. The administrator allows for the user to switch between sites, at whim. The

“site switcher” is a drop menu, which loads up a different DNS depending on the selected site.

A certain set of code was working for all sites except one. The application was not throwing errors, but nothing was getting displayed on the page. How do you find the problem? I’m going to step you through the process I took to solve the problem, and along the way, I hope you’ll learn a technique or two that you can use in your own debugging adventures.

Two Types of Errors

There are really two types of errors that you’ll encounter in your code: syntax errors and logic errors. Syntax errors are those ugly ColdFusion errors that you are used to seeing every day as you write code (I know I’m used to seeing them). Generally these are easy to discover and address. These are usually squashed during your development.

The most common reason why I see syntax errors on production sites is due to undefined variables being accessed. Sometimes there is some link somewhere that is missing a URL variable, or perhaps someone changed the Query String and reloaded a page. Sometimes a checkbox form element is accessed, yet the user did not check anything before submitting the form, so it is undefined or a shared scope variable may be accessed before it is set. Other reasons that ColdFusion might throw syntax errors might be invalid tags or attributes, mismatched tags, or database timeout.

Syntax errors are easier to find than logic errors. Logic errors occur when the code that you’ve written doesn’t satisfy the business requirements you were setting out to imple-

ment. Some business requirements might be easy to fix, such as display 20 products per page instead of 15. Others might become murky to catch, such as people who buy at least five products from Product Line A get a 10 percent discount on all products from Product Line A in their order, unless they have three products of Product Line B attached to the same order, whereas they receive the bigger discount of 10 percent off Product Line A products or 5 percent off of all Product Line B products. The best way to avoid business logic errors like these is to have a clear set of business requirements before you start to code the application. Sometimes it’s hard for the client to see the big picture.

Examining the Debugging Output

Going back to my problem of the day, I started out by assuming that the code was fine, since it was working for all of the sites except one. The problem must reside in the data. The page in question was supposed to show a list programs from the database. If no programs existed, then a blank display is not a surprise. I tracked down the query code and executed it manually against the database using Query Analyzer (an SQL Server tool). The query worked fine and the proper data was returned. This query didn’t contain any CF variables, so the cause of this couldn’t have been invalid values.



Figure 1: The ColdFusion Administrator Login

Home Help
 [-] App Server
 ColdFusion Admin
 JRun MC
 Restart
 System Logs
 Heartbeat
 Configure
 [-] Site Management
 Domains
 Emails
 Horde Web Mail
 File Manager
 Site Builder
 Web Counter
 AWStats
 Web Logs
 [-] Database
 [-] Apache
 [-] Applications
 [-] Advanced
[more...](#)

testmycfm DNS,Domain Host, and Email Options:

[Domain Host and DNS Configuration](#) [Email Configuration](#)

Domain Host and DNS Configuration

Please refer to this topic in the [Help](#) section before proceeding.

DOMAIN

Select Domain: (1)

Dedicated Address: [testmycfm.webappcabaret.net] Server Address: [pointer.webappcabaret.net]

Domain Name (mydomain.com) ? : mycoolcfm.com

Set Domain

IP/Host Address: testmycfm.webappcabaret.net

Apache Document Root ? : /usr/ngasi/contexts/testmycfm/testmycfm

Apache Directory Index: index.html index.jsp index.do index.html.var index.

AppServer Virtual Path(s) ? : /*.cfm /CFIDE/

Customize Apache Virtual Host

DNS & SUB-DOMAINS

DNS Service ? : false Set DNS

MX Address (separate multiple MX with commas):

USE SERVER DEFAULT

Sub Domain	Apache Document Root	Directory Index	Virtual Path(s)	IP/Host Address	CNAME
	/usr/ngasi/contexts/testmy	index.html index.jsp index.		testmycfm.webapp	<input type="checkbox"/>
	/usr/ngasi/contexts/testmy	index.html index.jsp index.		testmycfm.webapp	<input type="checkbox"/>

[Email Configuration](#) ▲

Imagine a hosting company dedicated to meet the requirements for complex web sites and applications such as those developed with ColdFusion MX.

At WebAppCabaret our standards based process and tools make deploying ColdFusion MX applications as easy as a point-and-click.

We call it **Point-and-Deploy Hosting**.

Our advanced NGASI Web Hosting management Control was designed for the hosting and management of ColdFusion web sites and applications thus cutting down on maintenance time.

Backed by an experienced staff as well as a **Tier 1 Data center** and network. Our network is certified with frequent security audits by reputable security firms.

All ColdFusion hosting plans have separate and individual installation AND instances of ColdFusion MX 6.1 Enterprise with **full access to ColdFusion Administrator** and JRun Management Console; so there is virtually no restriction or customization required for your application.

Complete power and control at the tip of your fingers. We take care of the system and hosting infrastructure so you can concentrate on development and deployment of your application. That is **real ROI**.

Log on now at <http://www.webappcabaret.com/cdj.jsp> or call today at 1.866.256.7973



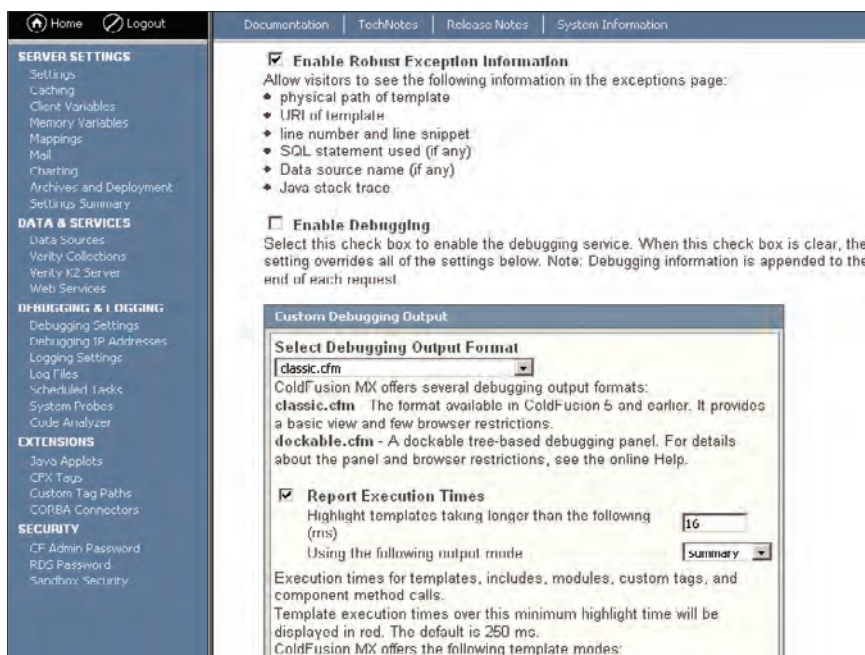


Figure 2: The Debugging and Settings Page

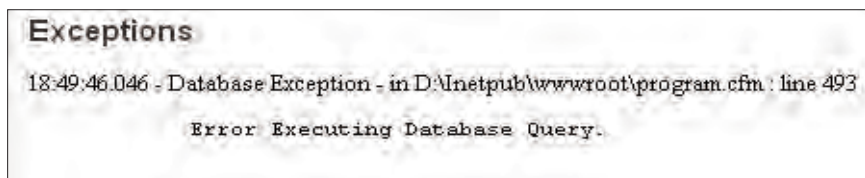


Figure 3: Database error

Just to be sure, I wanted to review the SQL actually being sent to the DB. For this I needed to turn on ColdFusion's debugging output. In normal cases, you wouldn't turn on debugging for a production site, but I decided this case was an exception. The site only has half a dozen users, none of whom could use the site until the error was fixed. You can turn on debugging from your ColdFusion Administrator using these instructions:

1. Go to your administrator. Most likely the URL is `www.yoursite.com/cfide/administrator`. You'll see a screen similar to Figure 1. It might look a little different for those who have already upgraded to CF7. Enter your password to login.
2. Under Debugging and Logging select Debugging Settings. You'll see the different things that can be displayed in debugging output, as shown in Figure 2.
3. Check the Enable Debugging box, if

not already done. I also like to click the Enable Robust Exception Information so that I can view the full SQL statement, and other information about the queries that run.

4. While you're in here, take some time to examine the list of other information that can be displayed in the debugging output. You can select the type of display, either dockable or classic. I prefer classic, the default method. You can also display various variable scopes, tracing information (used in conjunction with the `cftrace` tag), exception information, database activity, and general debug information. Most commonly I use the variables and database activity.

Back to the problem at hand: I enabled debugging and reran the template. I scrolled down and looked for the debugging output. The query I was looking for was not executed. Why was that?

cfdump and cfabort

The template in question is what I call a "case" template. Based on some mode variable, different code is executed within the template, often with completely different execution and results. I call it a case template because a different `cfcase` statement represents each mode. All of this inside the `cfswitch` tag. For more information, see: <http://livedocs.macromedia.com/coldfusion/7/html/docs/00000339.htm#1103819>.

This type of template is very similar to a fusebox approach to designing Web applications. The mode would be a "fuseaction" and the base template would be the "fusebox" template. I know by looking at the URL that a client-side redirect is not happening, which leaves two options. Either the wrong "mode" is being executed or a server side redirect is happening.

The first thing I wanted to do was to verify that the mode was correct. I turn to `cfdump` tag for that. `cfdump` is a very useful debugging tag that displays the contents of a variable. With simple variables, you could use a `cfoutput` to display the value of that variable, however for complex variables such as an array, structure, or XML document the `cfdump` is invaluable. `cfdump` has a few attributes, but the one we care about for the moment is the `var` attribute. You use the `var` attribute to specify which variable you want to display. Always make sure to enclose the variable name in pound signs, or else the variable name will be displayed instead of the variable value. You can check out the full documentation on `cfdump` at <http://livedocs.macromedia.com/coldfusion/7/html/docs/00000239.htm#3765824>.

I used `cfdump` to dump the value of the mode variable before entering the `cfswitch` tag. The code looked something like this:

```
<!-- code here -->
<cfdump var="#mode#">

<cfswitch expression="#mode#">
<!-- more code here -->
```

With debugging code in place, I reloaded the template. Nothing was displayed. I switched to a different site and tried again, and the mode was properly

displayed. It was empty, which was the default mode. That led me to believe there must be some sort of redirect that was being called in conditions that existed in the rogue site. In CF the most common method of redirect is to use a cflocation. I started pouring over the code looking for it. There was not an obvious redirect, but it may have been included in any one of a handful of includes or inside some CFC methods. Another tag came to the rescue, cfabort.

cfabort is used to stop the processing of a ColdFusion page. It has a single attribute, ShowError. The ShowError attribute accepts a string value, which is displayed to the screen before the templates execution is stopped. I added this line of code to the template before the switch statement:

```
<cfabort ShowError="It Got This Far">
```

I reran the template, but didn't receive the error message. That told me that the redirect was happening before the cfabort tag. By moving around the cfabort tag and I was able to isolate the line of code that caused the problem. It was a CFC method call that verified that the user was allowed to view this page. If the user didn't have access to view the page, then the system was actually operating properly (although some feedback would have been nice). I went back to the database to verify that my user had proper access to the database, and I did. Before the cfabort I added a dump of the user's access. Yes, the user should have

access. This obviously wasn't the problem.

At this point, I take another in-depth look at debugging output. I noticed something I hadn't noticed before, as shown in Figure 3. There was an error executing the database query. With this new bit of information, I jumped over to the admin and verified the data source. Sure enough ColdFusion could not connect to it. This was probably the cause of the problem. I opened up the data source to discover that the username was incorrect. I corrected the username, verified the data source, and then loaded the trouble page again. Everything started working as expected. Finally!

Catching Errors


So the problem was a failed database query and not a cflocation redirect. You might want to ask why I didn't see a ColdFusion syntax error if the database query failed, right? Good question, it shows that you are thinking. The answer is simple. The error was caught and the user was redirected to the "empty" index page. In this case the site was using the cferror tag to catch the error. cferror is a tag that you would normally put in the application.cfm to launch a specific error page when an error occurs. In CFMX7 you can also use the OnError event that is part of Application.cfc. More information about the cferror tag can be found at <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000242.htm#2022557>.

Information about the OnError method is found here: <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000697.htm#1188543>.

cferror usually redirects the user to a page that can process the error, usually displaying a nice error message to the user, logging the error, and/or sending an e-mail to notify a developer about the error. Unfortunately, this particular cferror tag just redirected the user to the site home page, which masked the error from my view. It is an internal site, which unfortunately, often gets fewer resources than the public-facing site, and I'm sure the error page was something that just hasn't been implemented for this site.

Where to Go from Here

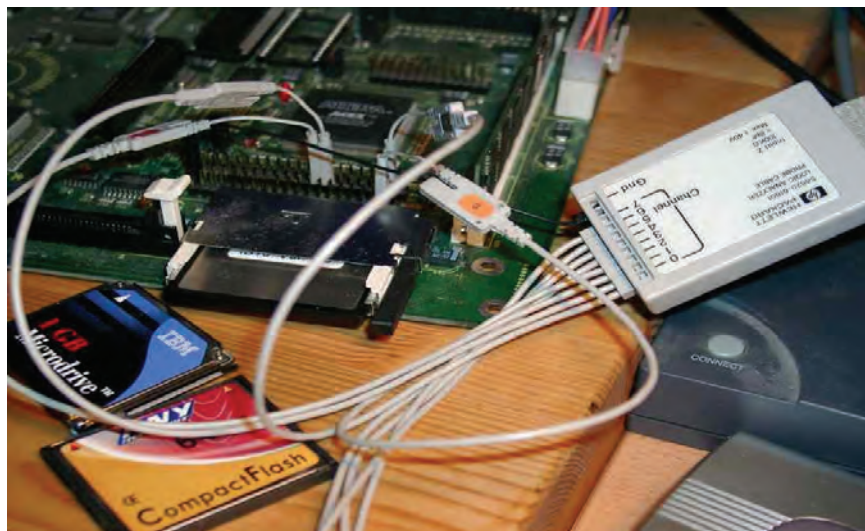
What do you want to look at next? If you're a BlueDragon user you can take a look at cfdebugger, which writes detailed information about the code that is executed to a file. More information about cfdebugger is in New Atlanta's enhancement guide, located here: www.newatlanta.com/products/bluedragon/self_help/docs/6_2/BlueDragon_62_CFML_Enhancements_Guide.pdf, and it was written about in the November 2003 article of *CFDJ*. At CFUnited, New Atlanta also announced the BlueDragon Profiler, which will allow you to get line-by-line details of a templates execution.

It is also worth noting that you can create your own debug templates for ColdFusion. The classic and dockable styles are located in the web-INF\debug directory of your ColdFusion installation. You can use them as a start to roll your own. Put your new debug template into the same directory and it will show up in the Administrator under debug settings. 

About the Author

Jeff Houser has been working with computers for over 20 years and in Web development for over 8 years. He owns a consulting company and has authored three separate books on ColdFusion, most recently *ColdFusion MX: The Complete Reference* (McGraw-Hill Osborne Media).

jeff@instantcoldfusion.com





CFEverywhere

Part 3

Conclusion



By Phill Cruz

After somewhat of a hiatus, we're back to continue and conclude our CFEverywhere journey. I should note that Dick Applebaum, my copilot for parts 1 and 2, was unable to join me for this last installment. It won't be the same

without his contribution but we'll have to make do. Moreover, as Dick provided the knowledge for the Mac side of things, I'll have to apologize in advance for the lack of detail on that front.

In the previous articles, we demonstrated how to deploy a CFML application into a self-contained package. We used BlueDragon, Jetty, and Derby to create a fully dynamic, database-driven application running locally, giving the appearance of a traditional desktop application. Now I'll deliver the final piece of the puzzle and demonstrate how to create an application launcher to deliver the ultimate user experience. To access your application, the user will be able to:

1. Download the application (or insert a CD)
2. Double-click an icon
3. That's it – there is no Step 3!

Up to now we've been starting Jetty and Derby from the command line by invoking the Java runtime and passing the appropriate JAR files. To make our own application bundle, we'll create some simple Java classes that wrap the necessary Jetty and Derby classes. If you're not familiar with Java don't

be intimidated. What we'll do is fairly straightforward and may be a good opportunity to get familiar with working with Java (in so far as it pertains to your CF projects). If you want to compile the Java source, you'll need the Java SDK installed on your system. You can get the SDK at <http://java.sun.com/j2se/1.4.2/download.html>. Please note that although you may have a Java Runtime Environment (JRE) installed, this is different from the SDK and doesn't have the necessary tools to compile Java code and create JAR files. It's also helpful to have a Java IDE at your disposal. Eclipse is a popular Java IDE that's readily available. You can download it from www.eclipse.org.

Create a new project folder called cfeverywherej to contain the files for the Java project. Create a file cfeverywere.java that looks like Listing 1. Let's take a look at the main method of this class. It begins by initializing some variables. It continues on to instantiate a JettyThread class and calls its run method. Looking at Listing 2, we see that JettyThread is a simple class that extends Thread. Its run method checks if the isStarted flag has been set. If not, it starts Jetty using the org.mortbay.jetty.Server class. After the JettyThread is created Derby is started in a similar fashion using the DerbyThread class (see Listing 3). After JettyThread and DerbyThread are created, we wait an amount of time defined by delayMilliseconds that allows time for the servers to initialize and get ready to accept requests. Finally, the Web browser is launched to the application URL using a platform specific command. In order to compile the class files, you will need to make sure the following jar files are on your build path:

```
cfeverywhere/server/lib/org.mortbay.jetty.jar
cfeverywhere/derby/lib/derbynet.jar
```

After you have compiled the Java class files, create a JAR file by using the JAR utility:

On Windows:

```
C:\j2sdk1.4.2_04\bin\jar.exe -cvf cfeverywhere.jar *.class
```

On Mac OS X:

```
C:\j2sdk1.4.2_04\bin\jar.exe -cvf cfeverywhere.jar *.class
```

If you are using Eclipse, you can create the JAR file by choosing Export from the File menu and choosing JAR file. Copy the cfeverywhere.jar file to the cfeverywhere project folder. We now have the Java classes required to create our application bundle. The process to create the application launcher is different for each platform so we'll discuss them separately.

To create our application launcher on Windows, we will use a utility called NativeJ. Visit their Web site at www.dobysoft.com/products/nativej and download the trial version of the program. Install the software and start a new project. Configure the project with these settings:

Executable

- *Application Type:* console
- *Allow application to be installed as a service:* no
- *Executable filename:* cfeverywhere.exe
- *Redirect system.out/system.err to:* popup message box/Event Log
- *Initial process priority:* normal

Java Runtime

- *JVM options:* -Djetty.home=server -Dderby.system.home=derby
- *Classpath:* .;server/lib/org.mortbay.jetty.jar;server/ext/commons-logging.jar;server/ext/xercesImpl.jar;server/ext/xml-apis.jar;server/lib/javax.servlet.jar;server/lib/org.mortbay.jmx.jar;server/ext/jasper-compiler.jar;server/ext/jasper-runtime.jar;cfeverywhere.jar

Application

- *Main class:* cfeverywhere
- *Default application arguments:* server/etc/cfeverywhere.xml

A sample NativeJ project file is included with the source code for this article. From the File menu click Generate Executable and you will, well, generate an executable.

Double-click the executable, and you should see a command line window appear that shows the server starting up, and then your CFML application should launch in your browser. Pretty cool, huh?

If your CF page didn't display properly it may be that the servers didn't complete the startup procedure before the browser was launched. You may have to increase the delayMilliseconds value in cfeverywhere.java to allow more time. When you want to quit your application you can't simply close the browser. You have to CTRL-C in the command line window and that will initiate a shutdown of the servers.

To create an application bundle on the Mac, you can use the same Java code to create the JAR file. Instead of NativeJ you would use the JarBundler application that is distributed with other developer tools for OSX, which are located at Developer/Applications. Configure JarBundler with the cfeverywhere.jar file and set the classpath similar to how NativeJ was configured; click "Create Application..." and you will have your Mac application

bundle. This process is described in more detail in this article:

<http://java.sun.com/developer/technicalArticles/JavaLP/JavaToMac3/>.

Launching from an HTA File

One aspect of the Java/NativeJ launcher that some folks might find undesirable is the appearance of the command window (and perhaps the license fee). On the CFEverywhere mailing list, Adam Haskell shared a great tip on how to use an HTA (HTML Application) file to create a CFE launcher that doesn't create a command line window. You can learn more about HTA at <http://msdn.microsoft.com/library/default.asp?url=/workshop/author/hta/overview/htaoverview.asp> (see Listing 4). When you double-click the cfeverywhere.hta file, the VBScript block executes, which instantiates an instance of the Wscript.Shell. This object allows you to execute the batch files to start Jetty and Derby. In order to give the servers time to start, we load a wait.html (Listing 5) in an iframe. Wait.html displays a loading animated gif and uses a meta-fresh to load the application page after a set time. When you want to quit, simply close the application/browser window and the onUnload() event will shutdown Jetty and Derby. It's a Windows-only solution but it works quite nicely and the price is right.

Creating a CFEverywhere CD

Creating a CD is really quite simple. For the entire process, we have been careful to only use relative paths when referring to files. All of the server components are designed to work in a read-only mode. Before we create the CD, let's set it up to automatically run our CFML application when the disc is inserted. Download AutoRun.exe and place it in the cfeverywhere folder. Create a text file called autorun.inf with these contents:

```
[autorun]
OPEN=autorun.exe cfeverywhere.hta
```

If you have a custom icon you can add:

```
icon=myicon.ico
```

To make a CD, simply create a project that contains everything in the cfeverywhere folder and burn the CD. Insert the CD on another computer (that has a Java runtime) and try it out. Note that starting a J2EE server and database server from CD is not the quickest thing in the world. You may have to set the wait time in wait.html to upwards of 20 seconds or more depending on the system. Regardless, it brings a smile to you face the first time you put a CD of your application in the computer and see your app launch.


Conclusion

Well, now you have all the tools and knowledge to create your own CFEverywhere applications. I hope this sparks your imagination and inspires you to try it out. If you do create a CFEverywhere application please share your experience on the CFEverywhere mailing list.

Resources

- To download the files used in this article go to: <http://philcruz.com>.

[com/cfeverywhere/downloads/cfeverywhere_files_part3.zip](http://www.cfeverywhere.com/downloads/cfeverywhere_files_part3.zip)

- *NativeJ*: www.dobysoft.com/products/nativej/
- *Bringing your Java App to Mac OSX*: <http://java.sun.com/developer/technicalArticles/JaVaLP/JaVaToMac3/>.
- *AutoRun*: www.tarma.com/products/index.htm#/products/autorun/
- *How to Launch CDs with HTML applications*: www.devx.com/webdev/Article/7023/0/page/1 

Listing 1: cfeverywhere.java

```
import java.io.IOException;

public class cfeverywhere {
    private static final String WIN_ID = "Windows";
    boolean isJettyStarted;
    String[] args;

    public static boolean isWindowsPlatform(){
        String os = System.getProperty("os.name");
        if ( os != null && os.startsWith(WIN_ID))
            return true;
        else
            return false;
    }

    public static void main(String[] args) throws IOException {

        boolean windows = isWindowsPlatform();
        int delayMilliseconds = 20000;
        String cmd = null;
        String applicationUrl = "http://localhost:8080/derbyemployees-
drilldown.cfm";

        System.out.println("Starting CFEverywhere...");
        // start jetty in a new thread
        new JettyThread(args).start();
        // start Derby in a new thread
        new DerbyThread(args).start();

        System.out.println("..waiting to launch browser...");
        java.lang.Object sync = new java.lang.Object();
        try{
            synchronized(sync){
                sync.wait(delayMilliseconds);
            }
        } catch (InterruptedException e){}

        System.out.println("Launching browser to " + applicationUrl);
        if (windows){
            cmd = "rundll32 url.dll,FileProtocolHandler " + applica-
tionUrl;
        } else {
            cmd = "open " + applicationUrl;
        }
        //execute the command
        Runtime.getRuntime().exec(cmd);
    }
}
```

About the Author

Phil Cruz is a Macromedia Certified Advanced ColdFusion developer and has over 12 years of experience in the computing industry. He is responsible for www.mach-ii.info, a community site for the Mach-II framework. As a micro-ISV, he created Tracking Tools, an easy-to-use bug tracking application built with Mach-II (www.tracking-tools.com).

phil@philcruz.com

Listing 2: JettyThread.java

```
public class JettyThread extends Thread{

    boolean isStarted;
    String[] myargs;

    public JettyThread(String[] args){
        isStarted = false;
        myargs = args;
    }

    public void run() {
        if (!isStarted){
            org.mortbay.jetty.Server.main(myargs);
        }
        isStarted = !isStarted;
        try {
            sleep((long)(1000));
        } catch (InterruptedException e) {}
    }
}
```

Listing 3: DerbyThread.java

```
import org.apache.derby.drda.NetworkServerControl;

public class DerbyThread extends Thread{

    boolean isStarted;
    String[] myargs;

    public DerbyThread(String[] args){
        isStarted = false;
        myargs = args;
    }

    public void run(){
        if (!isStarted){
            try{
                NetworkServerControl server = new NetworkServerCon-
trol();

                server.start(null);
            }catch (Exception e){}
        }
        isStarted = !isStarted;
        try {
            sleep((long)(1000));
        } catch (InterruptedException e) {}
    }
}
```


Listing 4: Cfeverywhere.hta

```
<head>
<script language="JavaScript">
newHeight = 480;
newWidth = 640;
self.resizeTo(newWidth,newHeight);
</script>

<script language='vbscript'>
Set WshShell = CreateObject( "WScript.Shell" )
WshShell.Run "start.bat",0,false
WshShell.Run "startdb.bat",0,false
sub closeOut
    WshShell.Run "stop.bat",0,false
    WshShell2.Run "stopdb.bat",0,false
End sub
</script>
<HTA:APPLICATION ID="WbrtChange"
icon="ttools_32.ico"
BORDER="dialog"
BORDERSTYLE="normal"
maximizeButton="no"
minimizeButton="no"
showInTaskbar="yes"
SCROLL="no"
SHOWINTASKBAR="yes"
SINGLEINSTANCE="yes"/>
<style>
iframe{
    border: none;
    margin: 0;
    padding: 0;
    height:480;
    width:640;
}

body{
    border: none;
    margin: 0;
    padding: 0;
    overflow : hidden;
}

</style>
</head>
<body onload="closeOut()">
<iframe scrolling="No" src="wait.html"></iframe>
</body>
```

Listing 5: wait.html

```
<html>
<head>
    <title>Loading CFEverywhere...</title>
    <meta http-equiv="refresh"
content="30;url=http://127.0.0.1:8080/derbyemployeesdrilldown.
cfm">
</head>
<body>
    
</body>
</html>
```

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

What's your PDF?



Precise Document Formatting



With activePDF Server, you gain full control over your PDF output with conversion options that allow you to specify page size, compression and resolution options, embed text, create bookmarks, concatenate to existing files, and more. Licensed per server, you can easily add PDF generation to virtually any Windows application.



Populate Dynamic Forms



With activePDF Toolkit's form-filling capabilities, you can dynamically populate PDF forms with data and images from a database, allow users using only Adobe Reader to fill-in and save forms and use PDF forms as document templates to precisely control image placement and resizing. With Toolkit's robust API, the automation of virtually any PDF manipulation task becomes possible - append, stamp, stitch, merge, paint, secure PDF and more.



Promote Digital Fidelity



Do you need to standardize PDF output within your enterprise? With DocConverter, you can easily use built-in support for "watched" folders to implement server-side PDF generation in a matter of minutes, with full control over the PDF output at the server level. Or, use DocConverter's programmable COM object to integrate convert-to-PDF functionality within your enterprise application.



Present Data Fashionably



Ensuring precise layout of an HTML document can be a nightmare, especially when printing. PDF guarantees pixel-perfect layout every time as what you see is what you print. With activePDF WebGrabber, you can dynamically convert any URL, HTML stream, or HTML file to PDF on the fly, while maintaining embedded styles.

Download your
free trial version today at
www.activePDF.com

Copyright © 2004, activePDF, Inc. All Rights Reserved.
"ACTIVEPDF", "Leading the iPaper Revolution" and the
activePDF logo are registered trademarks of activePDF,
Inc. All activePDF product names are trademarks of
activePDF, Inc.



Got Flash?

Macromedia teaches RIA development like never before



By Simon Horwith

Several years ago I reviewed DRIA – “Developing Rich Internet Applications.” At the time it was the first official Macromedia Curriculum course to teach Rich Internet

Application development. I thought I’d do a follow-up on that, as things have changed in the past couple of years.

Macromedia Training has been very busy keeping the courses up to date with Macromedia’s product offerings, and the DRIA class has gone through some revisions as well as new permutations.

For those developers new to Flash who simply want to learn it as a development environment, Macromedia now offers the “Macromedia Flash MX Application Development” course – a three-day class that teaches you the fundamentals of using the Flash authoring environment and ActionScript to build applications the right way.

Those of you interested in learning Rich Internet Application development (i.e., building Flash front ends that can interact with server-side resources) now have two class options for Flash development. Macromedia Training now offers a three-day course entitled “Macromedia Flash MX Training 2004: Advanced Application Development.” This course teaches ActionScript and OOP concepts, RIA UI elements, communication with servers (Web services and Flash Remoting), data persistence, and techniques for linking class files (business logic) with visual elements. In other words, everything you need to know to build Rich Internet Applications with Flash. The course is designed for people with some Flash experience.

For those of you who are new to Flash and want to learn how to develop Rich Internet Applications, Macromedia Training now offers “Macromedia Flash MX Training 2004: Application Development for Programmers.” This course is five-days long and, in addition to teaching everything that the “Advanced Application Development” course teaches, it

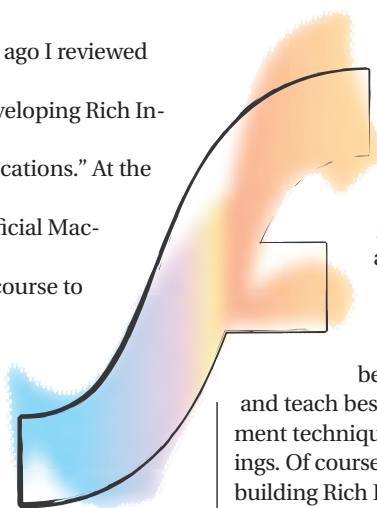
also teaches the UI components, movie clip objects, and other topics that somebody who already knows Flash would be expected to know prior to the “Advanced Application Development” class. This course is designed for people with some programming background with a language other than Flash (but no actual Flash experience).

Those are the new and improved Macromedia Training options for people who want to build RIAs with Flash. The courses have been updated for the most recent versions of Flash

and teach best practices and advanced concepts and development techniques better than any previous Flash course offerings. Of course, these days, Flash isn’t the only tool we have for building Rich Internet Apps. Unless you’ve been asleep for two years, you’ve probably already heard about Flex. For those of you who haven’t been paying attention, Flex, in a nutshell, is a server product designed to allow programmers to develop Flash front ends to their applications using a tag-based language. It encourages design pattern best practices and allows developers to create these “flashy” UIs in a language and environment that they’re more likely to be comfortable with. Basically, I like to think of it as “the real programmer’s Flash authoring tool” though stating that does open up a can of worms in certain audiences.

Macromedia has two classes that teach Flex 1.5 to developers and, let me tell you, they’re the best courses that I’ve seen from Macromedia Training to date. The “Developing Rich Internet Applications with Macromedia Flex” class is an intense 14 units in three days that’s geared for developers who already have a solid understanding of object-oriented programming concepts. In this class, students learn the fundamentals of the Flex Server, of MXML (the language used to develop Flex apps), and of Flex application architecture. They spend a lot of time studying various complex data and object types and how to use them, Flex UI components and layout, communication between Flex and Web services and Java Objects, using XML data, event handling, and data exchange between ActionScript and Java. They even learn how to combine several of these techniques and create drag-and-drop interfaces. The class uses Java Objects for its remote functionality in all of its labs.


The “Macromedia Flex for Web Application Developers” course teaches all of the material that is covered in the “Developing Rich Internet Applications with Macromedia Flex” course, plus two additional units. This course is four days in length and is comprised of 16 units. The additional



units add additional lessons that teach the navigation component options in Flex as well as manipulating the size, look, and feel of Flex UI components. Unlike "Developing Rich Internet Applications with Macromedia Flex," this course is ideal for ColdFusion and other Web developers who don't have a very solid Java or other object-oriented programming language background. Rather than Java class files, the "Macromedia Flex for Web Application Developers" course uses ColdFusion for all of its server-side functionality.

Both Flex classes are a cut above any course I've ever taught or attended. The amount of material is staggering. Of course, that's not so impressive – anyone can fill a class with tons of material. What's truly impressive is the quality of the material in both courses. The courseware authors have gone to great lengths to ensure that code and architectural best practices are also strongly emphasised in this course. Like all Macromedia training classes, students build an application over the duration of either class during hands-on walkthrough and/or lab exercises. One other thing that makes the Flex courses unique is Macromedia's control over which Macromedia Certified Instructors are authorized to teach the classes. Instructors must show Macromedia a real understanding of object-oriented programming concepts as well as a mastery of Flex before they are permitted to teach the class. Flex does for RIA development what ColdFusion does for dynamic Web application development and is, in simple terms, Flash development for developers. I highly recommend the Macromedia Flex classes to anyone who has an interest in learning how to develop applications with Flex, and to anyone who

is interested in Rich Internet Application development but finds Flash development and/or the Flash IDE unapproachable.

If you are interested in taking the Macromedia Flash or Flex classes, you can find a training partner near you by submitting the form found at <http://spectral5.macromedia.com/findaclass.cfm>. You can read the course outlines for the Flash classes by visiting www.macromedia.com/support/training/instructor_led_curriculum and clicking on the Flash classes that most interest you. Follow the links at www.macromedia.com/support/training/instructor_led_curriculum/curriculum_bumper.html to read the Flex course outlines. Alternatively, you can read/download the course outlines for these classes by visiting <http://training.aboutweb.com/> and selecting the classes that interest you from the menu. 

About the Author

Simon Horwith is the editor-in-chief of ColdFusion Developer's Journal and is the CIO at AboutWeb, LLC, a Washington, DC based company specializing in staff augmentation, consulting, and training. Simon is a Macromedia Certified Master Instructor and is a member of Team Macromedia. He has been using ColdFusion since version 1.5 and specializes in ColdFusion application architecture, including architecting applications that integrate with Java, Flash, Flex, and a myriad of other technologies. In addition to presenting at CFUGs and conferences around the world, he has also been a contributing author of several books and technical papers. You can read his blog at www.horwith.com. simon@horwith.com

Develop Powerful Web Applications with ColdFusion MX 7

- ▶ Easily build and deploy multi-step data entry forms with CFML tags
- ▶ Solve business reporting problems with Integrated Reporting
- ▶ Manage your account with HostMySite's Control Panel
- ▶ Call 24x7x365 for expert ColdFusion support

Visit hostmysite.com/cfdj for a special offer



HostMySite.com

1-877-215-4678



No-see-ums

Don't forget your bug spray!



By Nik Molnar

“ Sorry about that Adam, we'll have to take a deeper look at it. Uh-huh.

Yeah. No, I totally understand. Okay then, I'll give you a call if I still can't replicate the problem.”

Many of you recognize this phone call: the client has an issue and now you have to figure out why. I usually find issues like this much more frustrating than a cold hard error. At least with an error you get something concrete, you get facts about the problem. But these types of bugs, often problems with the application's logic, are the ones that sneak through to the client and are difficult to track down. I call this kind of bug No-see-ums, named after the teeny tiny mosquitoes commonly found in the Everglades.

Luckily for me, ColdFusion provides great tools for tracking down bugs like these. So armed with a fly swatter (error handling) and a can of Off (debugging), away I work at tracking down Adam's problem.

Off

I liken debugging to Off bug spray. Debugging tools will help you keep the bugs away, but they can still get through and bite you. ColdFusion provides two great ways to debug applications. The first is code based, through the use of tags like `<cfdump>`, `<cftrace>`, and `<cf timer>` and the function `isDebugMode()`; the second is through site-wide debugging.

Any developer who uses ColdFusion and another language no

doubt misses the `<cfdump>` functionality in their other development environment. `<cfdump>` provides a color-coded graphical representation of complex objects, as well as the standard output of simple ones. This includes red components, blue structures, green arrays, purple queries, orange functions, gray XML, and black WDDX packets. `<cftrace>` displays and logs debugging data about an application at runtime. It can track logic flow, variable values, and execution time. `<cftrace>` can display its output at the end of the request or in the debugging section at the end of the request, or in Dreamweaver MX, in the Server Debug tab of the Results window. ColdFusion also logs all `<cftrace>` output to the file logs\cftrace.log, in the ColdFusion installation directory. `<cf timer>` calculates execution time for any chunk code and displays it along with any output produced by said code. `isDebugMode()` is a simple function that returns a Boolean expression indicating whether site-wide debugging is enabled or not.

ColdFusion makes it easy for developers to turn on site-wide debugging information. When site-wide debugging is enabled, pertinent information is appended to the end of every request, allowing the developer to see much of what is going on under the surface. What information the debugger will show can be set in the ColdFusion Administrator.

Even though ColdFusion debugging gives developers an enormous head start on creating bug-free applications, it's only a good defense. Many times you will still need to anticipate and take action to accurately eliminate bugs.

Ray Romano

Ray Romano has said, "I don't care how brave you are; you don't just rush in and kill an unknown bug. You got to figure it out first. I don't know if it has the gift of flight. What if it's a flyer? Then you got to close up hatches and stuff."

Bugs that generate a ColdFusion error are actually easy to identify, you just have to know what to look for.

When there is a ColdFusion error, it provides information about the error in an easy-to-access structure. The structure contains many keys, a lot of which are optional. (So be sure to check for their existence before referencing them.) The most recognizable key in the structure is the message key. It contains the default error message, usually shown in bold on a default error screen. If no error message is available, the key will contain an empty string. The detail key contains a detailed HTML message from ColdFusion and will help you determine which tag caused the error. ErrorCode will most likely contain an empty string, but you can supply a value for it through the errorCode attribute of <cfthrow> tags. ExtendedInfo is similar to ErrorCode in that it's usually an empty string. The exception to this rule is when the Type key, which contains the error's type, is the string application. Since ColdFusion is running Java under the hood, all errors are really Java errors. When an error is thrown, the Java Virtual Machine reports a servlet exception, which is stored in the RootCause key.

TagContext is one of the most robust error keys. It contains a stack of structures, where each element in the stack represents an open tag, and the structure in that element contains more detailed information about that tag. This structure will contain an ID key that represents the open tag. If the open tag is a custom tag, the value will read cfmodule, and if it is <cfscript>, it will read "??". The structure will also tell you which line of code the open tag can be found on, the Line key, and in which ColdFusion file it is in, the Template key. The raw Java stack trace for the error can be found in the Raw_Trace key. The structure also contains two peculiar keys. The first is Column, which is always 0, but is retained for compatibility with older versions of ColdFusion. The second is Type, which is always a string equal to CFML.

Other keys in the error structure show up depending on the type of error that has been thrown. For database errors, an additional five keys may be present. NativeErrorCode contains an error code submitted by the database driver. Because of the varying drivers that ColdFusion supports, the values for this will vary. However, if the driver does not submit a value, the default is -1, and if the query is a query of queries, the value will be 0. SQLState is very similar to NativeErrorCode in that it is derived from the database driver. It is code given to assist in the diagnosis of failing database operations, and its default is also -1. Sql contains the parsed SQL statement that was sent to the DBMS, while the string representation of the DBMS error is stored in the QueryError key. If <cfqueryparam> tags were used in the query, the Where key will contain the name-value pair of each.

Another special key that may appear is the ErrNumber key, only found for expression exceptions. It represents an internal expression error number. Missing include errors add a MissingFileName key that gives the name of the missing file. Locking errors add an additional two keys: LockName and LockOperation. LockName contains the name of the affected lock. If the affected lock is unnamed, the value will be anonymous. LockOperation will contain the operation that failed or unknown.

Now that we know what we are dealing with (read "All hatches closed"), we can get offensive.

The Fly Swatter

ColdFusion provides a handful of mechanisms for handling errors. There are tags in the form of <cferror>, <cftry>, <cfcatch>, <cfthrow>, and <cfrethrow>; site-wide management in the ColdFusion Administrator; and (new to CFMX 7.0) there is an onError() function in the Application.cfc. Each method of error handling has its own strengths and weaknesses, but a combination of the methods provides the most complete solution.

Tags

While the scope of this article will not cover the ins and outs of the tags ColdFusion gives us to handle errors, a quick overview is necessary. <cferror> allows you to tell ColdFusion to run a particular template when an error of any given type is thrown. This is a great way to beautify the default error screen and give all pages, even ones that error, a consistent look and feel. <cftry> and <cfcatch> are used hand in hand. If an error occurs within a <cftry> construct, ColdFusion will immediately run the code in the corresponding <cfcatch> construct. Inside of a <cfcatch> construct you may also call <cfrethrow>, which will continue to throw the error as if there was no <cftry> and <cfcatch> clause at all. To make your own custom errors occur at any time, you can use the <cfthrow> tag. This is great for creating errors that pertain to just your application, not all applications in general, such as a business logic error. The attributes that make up <cfthrow> allow you to control all the main parts of an error, as explained earlier.

Administrator

ColdFusion also provides a site-wide error handler, set in the ColdFusion Administrator (under Server Settings/Settings at the bottom of the page). This setting allows you to provide a relative template path, which ColdFusion will execute when an error occurs. This template could do things such as send the sites administrator an e-mail with the error details, or just be used to dumb-down the errors that an end user sees.

Application.cfc

In ColdFusion MX 7.0 Macromedia introduced us to the Application.cfc. This upgrade to the Application.cfm/onRequestEnd.cfm also gave us developers ways to control many aspects of our applications. This control is implemented through specially named functions that ColdFusion automatically calls at the appropriate time. Of the most interest to us today is onError(). Any code placed in the onError() function in Application.cfc will run when the application errors. From this central location, developers can forward errors to the appropriate parties, redirect users to superficial error pages, and log errors. ColdFusion passes two arguments to the onError() function when it calls it (thus you should always have two <cfargument> tags in your onError() function). The first argument is an error structure, just like the one described earlier. The second argument is a string that contains the name of the Application.cfc method, if any, in

which the error occurred. This argument has special implications for errors that happen in the `onSessionEnd()` and `onApplicationEnd()` functions. `onSessionEnd()` is fired when a session time-out setting is reached for a user's session, or, if using J2EE sessions, the user closes the browser. The `onApplicationEnd()` function is run when the server times out or is shut down. Because users are not requesting data from ColdFusion when these functions run, they cannot see any errors the functions may throw, and, because of this, logging errors that come from these functions is crucial.

In addition to `onSessionEnd()` and `onApplicationEnd()`, `Application.cfc` implements their opposites, `onSessionStart()` and `onApplicationStart()`. `onSessionStart()` fires when a new session is created by a user, including ColdFusion event gateway sessions. `onApplicationStart()` is triggered when the application first starts – either when the first request for a page is processed or the first CFC method is invoked by an event gateway instance, Flash Remoting request, or a Web service invocation. It's great for setting application-wide variables, such as the names of data sources or the location of mail servers.

As stated before, `Application.cfc` contains a replacement for `Application.cfm` and `onRequestEnd.cfm`. The `onRequestStart()` function runs when ColdFusion receives an HTTP request, a message to an event gateway, a SOAP request, or a Flash Remoting request. Following the completion of `onRequestStart()`, `onRequest()` will run. This function can contain the main display of a page. For example, this would be a good place to put all the core files for a Fusebox application. (For more information on Fusebox see <http://www.fusebox.org/>.) `onRequestEnd()` runs after all pages and CFCs in the request have been processed.

Order of Operations

Between all the new functions of the `Application.cfc`, error handling techniques, and debugging, it gets a little difficult to follow what ColdFusion is going to process and when. Here is a quick rundown of how ColdFusion processes requests:

1. CFC initialization code at the top of `Application.cfc`
2. `onApplicationStart()`, if not run before for current application
3. `onSessionStart()`, if not run before for current session
4. `onRequestStart()`
5. `onRequest()`, or the requested page if there is no `onRequest()` method
6. `onRequestEnd()`

ColdFusion could at anytime have an event fired that will run the `onSessionEnd()` function or `onApplicationEnd()` function. If `onApplicationEnd()` is triggered, it doesn't fire events to run `onSessionEnd()`.

Also ColdFusion could have an error at anytime.

ColdFusion executes its error handling in the following order:

1. `<cfcatch>`, if present and if an error happens within a `<cftry>`
2. `onError()`, if the application has an `Application.cfc` file with `onError()` within it
3. `<cferror>`, if present and if exception type is not "request"

4. ColdFusion Administrator's Site-wide Error Handler, if defined
5. `<cferror>`, if present, with exception type of "request"
6. Display standard error

Putting It All Together

I have finally found the bug that Adam called me about earlier. Let me tell you, it was difficult to track down. Among all the tools I have in ColdFusion, it still took me a long time to find the problem.

The reason Adam's problem was difficult to track down was because of what I will refer to as a "chain with nesting." I call it that because I found the error in a sequence of CFC function calls (many of which make a request to the database based on the value passed into them) in which each function receives arguments that are part of the results of a previous function call. To make matters worse, some of the functions in the chain call upon other functions internally, which are located in different components and sometimes create their own chain of calls. If one function returns the wrong result due to a bad query or perhaps some bad math, it could completely throw off everything all the way down the chain. This scenario often happens in large projects, especially those that are object-oriented or pseudo object-oriented.

To help myself out of these predicaments, I have put together

a little framework using many of the components we have learned about today. The end result is a stack (array) of each function call, placed in the order they were processed. Each function call gets its passed-in arguments logged (important because ColdFusion, as does many programming languages, destroys a functions arguments once the function finishes executing), its results logged, as well as basic information such as when the function was executed and how long it took to execute.

Since Adam's application is very complex, I have created a much smaller, simpler version of the problem, one you can look at, and, if so desire, easily implement in your own application.

This example randomly picks one of 10 phrases and reverses it. It's actually quite silly, much like all my bug jokes so far, but should give you a good idea of where I'm coming from.

Listing 1 shows the file the user will request. It's very straightforward and creates a small chain of function calls on line 4; it also instantiates the CFC shown in Listing 2. (Listings 2 and 3 can be downloaded from <http://coldfusion.sys-con.com>.)

Listing 2 is a component with two functions. The first, `getPhrase()`, will return a phrase based on the passed-in `phraseNumber`. If no `phraseNumber` is passed in, it will randomly pick a number. `reversePhrase()` asks `getPhrase()` for five random phrases and reverses and returns the fifth. The code of interest is the `super.onCall()` calls on lines 6 and 30, as well as the `super.onResult()` calls on lines 23 and 36. In this case "super" is a keyword that refers to `Application.cfc` (see Listing 3), since `test.cfc` extends `Application.cfc` (line 1, Listing 2).



Listing 3 is where the magic happens. When the user makes a request, based on the order of operations discussed before, the first bit of code ColdFusion processes is the onRequestStart() function. This creates an empty one-dimensional array called the cfcStack that will be accessed later in the request. Next index.cfm is run, which calls on my getPhrase() and reversePhrase() functions. Each of those functions call onCall() and onResult(), which log and stack what is going on in each function call. ColdFusion then processes the onRequestEnd() function, which checks to see if debugging is enabled and, if so, <cfdump>s the cfcStack.

This allows me to see each function that was called, in the order that it was called, what parameters it had to work with, and what results it returned. Notice the use of reserved functions described above onRequestStart() and onRequestEnd() as well as my own functions onCall() and onResult() within the Application.cfc.

I also added an onError() function, which in the case of an error would e-mail me the cfcStack as well as all the error information, and, if debugging is on, <cfthrow> the error so I can see it immediately on the screen.

Just by taking a quick look at the cfcStack that is being dumped at the end of the request, I can tell if each function individually is behaving properly, as well as see if each one is fitting in with the parts of the whole request.

This simple <cfdump> is what led to this phone call: "Hey Adam, yeah, should be good to go. Alright man, thanks a lot.

I hope you have a good weekend. Oh you're going there? Well have fun, and bring your bug spray; it can be pretty nasty this time of year."

About the Author

Nik Molnar is a ColdFusion/Flex developer with over seven years experience. He has led teams through the development of enterprise applications for the mortgage, sports ticketing, and stock industries. He is an amateur Iron Chef and posts regularly at his blog: foodDuo.com. He lives with his wife Katy and his dog Jacques in Orlando, Florida.

nik@truetechnology.com

Listing 1

```
<cfobject type="component" name="testObj" component="test">

<!--- A small chain of CFC function calls --->
<cfset backwardsPhrase = testObj.reversePhrase(testObj.getPhrase())>

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

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

Efficient Web Content Management

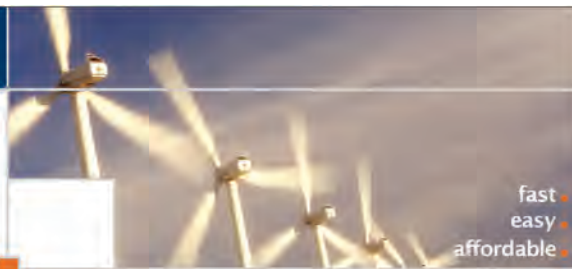
CommonSpot™ Content Server is the ColdFusion developer's leading choice for efficient Web content management.

Our rich Web publishing framework empowers developers with out-of-the-box features such as template-driven pages, custom content objects, granular security, and powerful ColdFusion integration hooks (just to name a few), allowing you to rapidly build and efficiently deploy dynamic, high performance Web sites.

CommonSpot's open architecture and extensive APIs enable easy customization and integration of external applications. With CommonSpot, you can design and build exactly what you need. Best of all, CommonSpot puts content management in the hands of content owners. With non-technical users responsible for creating and managing Web content, developers are freed to focus on strategic application development.

Evaluate CommonSpot today.

To see your site *running* under CommonSpot, call us at 1.800.940.3087.



features.

- 100 % browser-based
- Content object architecture
- Template driven pages
- 50+ standard elements
- Content reuse
- Content scheduling
- Personalization
- Flexible workflow
- Granular security
- Mac-based authoring
- 508 compliance
- Full CSS support
- Custom metadata
- Taxonomy module
- Extensible via ColdFusion
- Web Services content API
- Custom authentication
- Replication
- Static site generation
- Multilanguage support

1.800.940.3087
www.paperthin.com

Paper | Thin

© Copyright 2005 PaperThin, Inc. All rights reserved.

Event-Model Programming with CF

See what your code is doing



By Jared
Rypka-Hauer

If there's one word that summarizes the core of a successful software development project, it's organization. A lot has been written about the organization of documentation and organization of code, but not a lot has been written regarding the organization of the way code is executed.

We're going to take a look at one way to do exactly that using the concept of events to control how code is executed within an application. The term event should be fairly familiar to anyone who's used JavaScript with HTML (raise your hand if you haven't) and seen terms like `onClick` or `onSubmit`. Simply put, an event model is a way of organizing not the code, but the way that code is executed (or execution path – the order in which code is executed to respond to a particular request).

This method of organization often goes hand in hand with using frameworks like ModelGlue, Mach-II, or Fusebox. In each of these cases, events (Fusebox calls them `fuseactions`) are given arbitrary names and passed along on the URL or in a form post, each named event causing sections of code to be executed. As frameworks become a larger part of how we work, an in-depth examination of using events to organize and control execution within an app is important.

Event-Based Flow Control

In the case of Model-Glue, Fusebox, and Mach-II, the actions taken by the application when the user clicks a link are guided by an XML file based on parameters passed on the URL. Among other things, the XML file typically includes several sections that outline the flow of application logic for all actions the application may take (saving data, altering data, retrieving data for display, etc.). The various requirements for any given page can

be provided by structuring your events from high-level (or will cause lots of different actions to be taken) to low-level (or very specific – it will only do one thing).

While the examples shown are using ModelGlue, the same structure can be applied to Fusebox and Mach-II. The terminology and a few of the specifics may change, but the ideas are applicable to any of the three big HTML-interface frameworks. If you aren't familiar with ModelGlue, head over to www.model-glue.com and download it. It includes a QuickStart Guide that covers how to use the whole XML file and how to use the framework to build applications. Because this article is intended to highlight the use of tiered event models to structure Web applications, a complete explanation of ModelGlue is beyond its scope.

We also need to give a few definitions before we get started:

- *Model* (sometimes called a domain model) is a collection of code that connects your application to databases, other Web sites, Web services, and any other external tools. The model also contains all the code for dealing with data, setting variables... it's the workhorse of the application.
- *View* is what shows up on the browser and is pretty straightforward. Keep in mind, however, that browser can also mean HTML-enabled e-mail client, which means that your view could take limited action such as sending itself to an e-mail recipient instead of the user's browser.
- *Controller* is a collection of code that passes messages back and forth between the view and the model. It's the traffic cop in the Model-View-Controller (MVC) design style.

Back to Our Regularly Scheduled Event Modeling...

Any good construction project begins with a solid foundation, and I've found a three-tier event model to be the best. It provides clear separation between user input, data interactions, and UI results. It's also undoubtedly the easiest way to start and maintain an event-driven application as it grows in complexity. It becomes a matter of choosing actions from a menu (your low-level events) that you can then plug into your middle-tier events and even if you have to add a tier in some cases, keeps things very consistent, clean, and even simple.

You want to know what a high-level event is? It's an event that shows up in the URL, either as part of a hyperlink or in a form field. It's expressed via URL syntax that, for example, says `index.cfm?event=showMeTheMoney` and tells the framework what action to perform. Back in the day when I used page-based development, I would have a page called `listUsers.cfm`. Now, I have an event called `listUsers` and the associated URL looks like this: `index.cfm?event=listUsers`. It's not untrue that you can, in general, just replace the word page with event and have a fairly clear picture of what we're up to here.

Take a look at Listing 1. We have three events listed to get one page. Undoubtedly the first thought that crosses your mind is: "Why on earth should I use three events to get one page on the screen!?" Trust me, I understand! Although you don't need three events, this article is about tiered event models and the idea is to create your events in layers so you can later add to the basic model quickly and with a minimum of fuss.

Listing 2 is where the tiered event model really starts to click and the idea should really take hold. If you look at the top-tier event (home), you'll see that we've added a broadcast tag (which, in the case of ModelGlue, is how you tell the application to do things like interact with a database or the file system), and we've added result tags as well. Result tags are used to branch events. Like an implied condition (if-else), the result tags say, "If the controller sets a result named 'loggedIn', do this action, otherwise ignore me. If the controller sets a result named 'notLoggedIn', then do this other action." You can include the same result multiple times. For instance if you wanted to log views by logged-in users, you could have `<result name="isLoggedIn" do="logAuthUserAction" />` and `<result name="isLoggedIn" do="showView" />`. This is how the ModelGlue controller sends information back to the framework's kernel.

From Simple to Complicated

By now it's fairly apparent that even simpler views should use this method for the sake of consistency and eventual maintenance. We can count on functionality expanding, and using a method like this allows you to easily plug new functionality into events. For a peek at how this can set the application up for the next level of complexity, see Listing 3.

Look at the middle-tier event ("homePage"). Notice that I've added a new section to the `homePage` event: broadcasts. This includes a message tag and an argument tag. This new section tells ModelGlue to look in the `<controllers .../>` section of the XML config file and find the message named `getContent`, then execute the associated method. In this case, we're using the `<argument .../>` tag to specify the name of the page for which to fetch content. It would be just as easy to specify the page being requested using a URL variable, or use these two in combination by using conditional logic to apply a default value in the absence of an explicit one.

For each of these events, the views are rendered into a variable named `body`, a fact that you can see by looking at the `<view .../>` tags. This makes it possible to have a single event named `layout` to process all the views. All three of the major frameworks support content variables to capture the content of screens for later output to the browser. In the file `layout.cfm`, you'll see `<cfoutput> #view-State.getView("body")# </cfoutput>` in ModelGlue. In the other frameworks you'll see similar syntax for doing the same thing. In Fusebox, you'll see a simple `<cfoutput>#contentVariable#</cfoutput>`. In Mach-II, you'll see the term `contentKey` instead of `content`

variable, although the idea is still the same: collect the output of a particular event request and use a single `layout.cfm` template to draw the screen for return to the browser.

In general, however, your `layout.cfm` file provides a basic layout with header, footer, and navigation as shown in Listing 4. This file can be expanded upon, adding `cfif` tags to check for various content variables that may or may not be present. Some things will change based on the login status of the user, certain application conditions, and so on. It is recommended that each of these page fragments be generated into their own content variables and displayed by a layout template like the one shown in Listing 4. It's not uncommon to have more than one layout template for different site sections as the layouts change, allowing you to minimize the number of places in your application that need to be kept up during maintenance cycles.

Event-Driven Grid Layouts

Grid layouts are an effective technique for breaking complex pages down into manageable pieces. Portal sites often use a grid layout, breaking complex pages down into several columns and each column into small sections often called pods. This method makes complex pages easier to maintain while facilitating complex sites that are also easy to use. It also provides a good opportunity to demonstrate a slightly deeper event model because we can create a collection of events that will build our pods and tie them into any other middle-tier event. This creates a very modular design that couldn't be easier to maintain or expand upon.

Figure 5 illustrates a static collection of pods. The nice thing about this particular layout technique is that there is one event responsible for adding pods, and it spawns a self-contained process for creating one view fragment called pods that can be displayed by `layout.cfm`. It makes adding new pods incredibly easy. Simply add a new event for a new pod, then add the new pod to the `getPods` event. Nothing outside this process needs to know of the change because the `getPods` event simply creates a view fragment that is output in the `layout.cfm` file. Unfortunately, this doesn't allow for dynamically created pods, and if you have an application in which users are given the ability to choose the pods they wish to see once they've authenticated (e.g., see <http://my.yahoo.com>), there are just a few more changes that need to be made. Let's look at Listing 6.

It's not as complicated as it may seem at first:


- `homePage` event spawns the `getPods` event and asks the model for a list of pods.
- If the results of this request indicate that pods are in need of rendering, it spawns the `getPod` event to handle the first pod in the `podListEventKey`.
- `getPod` asks the model for the data it needs to render the first pod in the list.
- Off we go to the `getPodLayout` event for insertion in a view (probably via `cfinclude`, because the pod's name isn't static; it's stored in the event object).
- Rendered pods are saved in a content key called `pod`.
- Return to `getPods` to see if the last pod has been rendered or not. If it has, we head back up to `homePage` for rendering via the `layout` event. If not, we go through the `getPod/getPodLayout` process, removing the just-processed pod name from the list until there aren't any pods left to handle.

I know you have to be thinking why on earth should I work in the XML file, in the model, in the view, and add the framework's overhead

to the mix? It's going to bog down, run slow, and cause problems right? Wrong! If that were the case, then Macromedia.com, which runs on a mix of Mach-II with a little ModelGlue for flavor, would die at peak times every single day...but it doesn't. Why?

Because of well-planned architecture, design, and testing. Because using these techniques allows the creation of a pod system that can be plugged in anywhere in the application: written once, maintained in one place, but used wherever it may be needed or even just wanted. Once the initial design is working, there's testing and refactoring into faster and more efficient designs. Rendered content can be used with RAM-based caching or disk-based caching to accelerate response times. Hardware, server settings, and application code can be tweaked and tuned, but without very clearly structured and well-organized code you're going to start fighting an uphill battle from the beginning. It's cheaper in the long run to put more effort into design and architecture than to erect a mass of spaghetti code that will eventually turn into a maintenance nightmare.

So organize your code. Use events to create tight little routines that do as little as possible themselves and plug them into each other to create larger routines, then use high-level events

to tie those together. For me it works well to start with the event model like we saw here, because it helps me see what my code will be doing. If you take anything away from this article, it should be that by using solid design principles we can create applications that are not just scalable, but stable under high load; easily maintained and upgraded; provide extremely high value to our clients; and just plain fun to build. 

About the Author

Jared Rypka-Hauer is founder and CEO of Continuum Media Group, LLC, based in Minneapolis. Jared's articles are regularly featured in his blog, CF_Objective, which is known for its focus on object-oriented Web development. Jared is dedicated to furthering the cause of best practices and better Web development in the ColdFusion and Web communities. He is organizing a technology conference in Minneapolis on object-oriented development and enterprise integration. He seeks to help restore Minneapolis to its former glory as the "other Silicon Valley."

jared@web-relevant.com

Listing 1

```
<event-handlers>
  <event-handler name="showHomePage">
    <results>
      <result do="homePage" />
    </results>
  </event-handler>
  <event-handler name="homePage">
    <views>
      <include name="body" template="dsp.home.cfm" />
    </views>
    <results>
      <result do="layout" />
    </results>
  </event-handler>
  <event-handler name="layout">
    <views>
      <include name="layout" template="template.cfm" />
    </views>
  </event-handler>
</event-handlers>
```

Listing 2

```
<event-handlers>
  <event-handler name="showHomePage">
    <broadcasts>
      <message name="needLoginStatus" />
    </broadcasts>
    <results>
      <result name="loggedIn" do="homePage" />
      <result name="notLoggedIn" do="loginForm" />
    </results>
  </event-handler>
  <event-handler name="homePage">
    <views>
      <include name="body" template="dsp.home.cfm" />
    </views>
    <results>
      <result do="layout" />
    </results>
  </event-handler>
  <event-handler name="loginForm">
    <views>
      <include name="body" template="dsp.loginForm.cfm" />
    </views>
    <results>
```

```
      <result do="layout" />
    </results>
  </event-handler>
  <event-handler name="layout">
    <views>
      <include name="layout" template="template.cfm" />
    </views>
  </event-handler>
</event-handlers>
```

Listing 3

```
<event-handlers>
  <event-handler name="showHomePage">
    <broadcasts>
      <message name="needLoginStatus" />
    </broadcasts>
    <results>
      <result name="loggedIn" do="homePage" />
      <result name="notLoggedIn" do="loginForm" />
    </results>
  </event-handler>
  <event-handler name="homePage">
    <broadcasts>
      <message name="getContent">
        <argument name="page" value="home" />
      </message>
    </broadcasts>
    <views>
      <include name="body" template="dsp.home.cfm" />
    </views>
    <results>
      <result do="layout" />
    </results>
  </event-handler>
  <event-handler name="loginForm">
    <views>
      <include name="body" template="dsp.loginForm.cfm" />
    </views>
    <results>
      <result do="layout" />
    </results>
  </event-handler>
  <event-handler name="layout">
    <views>
      <include name="layout" template="template.cfm" />
    </views>
```



```

    </event-handler>
</event-handlers>

Listing 4
#viewCollection.getView("header")# #viewCollection.getView("menu")#
#viewCollection.getView("body")# #viewCollection.getView("footer")#

```

```

Listing 5
<event-handlers>

    <event-handler name="Home">
        <views>
            <include name="body" template="dsp.home.cfm"/>
        </views>
        <results>
            <result do="getPods"/>
            <result do="layout"/>
        </results>
    </event-handler>

    <event-handler name="getPods">
        <results>
            <result do="getPodOne"/>
            <result do="getPodTwo"/>
            <result do="getPodThree"/>
            <result do="podLayout"/>
        </results>
    </event-handler>

    <event-handler name="getPodOne">
        <broadcasts>
            <message name="needPodContent">
                <argument name="forPod" value="podOne"/>
            </message>
        </broadcasts>
        <views>
            <include name="pods" template="dsp.podOne.cfm"
append="true"/>
        </views>
    </event-handler>

    <event-handler name="getPodTwo">
        <views>
            <include name="pods" template="dsp.podTwo.cfm"
append="true"/>
        </views>
    </event-handler>

    <event-handler name="getPodThree">
        <broadcasts>
            <message name="needUserInfo">
                <argument name="myself" value="true"/>
            </message>
        </broadcasts>
    </event-handlers>

    </broadcasts>
    <views>
        <include name="pods" template="dsp.podThree.cfm"
append="true"/>
    </views>
</event-handler>

    <event-handler name="podLayout">
        <views>
            <include name="pods" template="dsp.podLayout.cfm"
append="true"/>
        </views>
    </event-handler>

    <event-handler name="layout">
        <views>
            <include name="layout" template="dsp.podlayout.cfm"/>
        </views>
    </event-handler>
</event-handlers>

```

Listing 6
<event-handlers>

```

    <event-handler name="showHome">
        <broadcasts>
            <message name="checkForPods" />
        </broadcasts>
        <results>
            <result name="hasPods" do="getPods"/>
            <result do="homePage"/>
        </results>
    </event-handler>

    <event-handler name="homePage">
        <views>
            <include name="body" template="layout.main.cfm"/>
        </views>
        <results>
            <result do="layout" />
        </results>
    </event-handler>

    <event-handler name="getPods">
        <broadcasts>
            <message name="checkRemainingPodCount">
                <argument name="podListEventKey" value="podList"/>
            </message>
        </broadcasts>
        <results>
            <result name="hasAnotherPod" do="getPod"/>
            <result name="noMorePods" do="podLayout"/>
        </results>
    </event-handler>

    <event-handler name="getPod">
        <broadcasts>
            <message name="needPodContent">
                <argument name="podListEventKey" value="podList"/>
                <argument name="podDataEventKey" value="podData"/>
            </message>
        </broadcasts>
        <results>
            <result do="getPodLayout"/>
        </results>
    </event-handler>

    <event-handler name="getPodLayout">
        <broadcasts>
            <message name="decrementPodList" />
        </broadcasts>
        <views>
            <include name="pods" template="dsp.podTemplateWrapper.cfm"
append="true">
                <value name="podDataEventKey" value="podData"/>
            </include>
        </views>
        <results>
            <result do="getPods"/>
        </results>
    </event-handler>

    <event-handler name="podLayout">
        <views>
            <include name="pods" template="dsp.podLayout.cfm"
append="true"/>
        </views>
    </event-handler>

    <event-handler name="layout">
        <views>
            <include name="layout" template="dsp.podlayout.cfm"/>
        </views>
    </event-handler>

    <event-handler name="Exception">
        <views>
            <include name="body" template="exception.cfm"/>
        </views>
    </event-handler>
</event-handlers>

```

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



New Debugging Features of CFMX7

Making some classic debugging tasks a lot easier



By Kevin
Kazmierczak

Debugging code is one of those tasks that we hate because it means that something is wrong in the code. Previous versions of ColdFusion have provided tools to assist us in finding problems with

our code, but CFMX7 has introduced some new tools and features that make some of our old debugging techniques easier.

The introduction of the new `cftimer` tag has simplified the use of `GetTickCount()` to track execution time for blocks of code. Other important new debugging features of CFMX7 are the new result attribute of certain tags, especially `cfquery`, and the ability to use the `GetMetaData()` function to return detailed information on the executed query.

<cftimer>

The `cftimer` tag allows programmers to track execution time of specific sections of code. Previous to this tag, most of us used something similar to the following:

```
<cfset starttime = GetTickCount()>
<!--- CODE TO TIME HERE --->
<cfoutput>#GetTickCount() - starttime#</cfoutput>
```

This would total up the number of milliseconds it took to run the code between the `GetTickCount()` functions. This method worked fine but was clumsy for a few reasons – it left unnecessary timing variables floating around your code and there was no easy way to turn it on or off without building special wrappers around it. Using `cftimer` solves these problems by allowing us to wrap our code within a `cftimer` tag to track the execution time in milliseconds. The tag is very simple to use and it only has two optional attributes:

1. **Label:** Text to name the execution block to time
2. **Type:** Output format of the execution time with label
 - *Inline:* Displays execution time after code block on page
 - *Outline:* Displays an outline around code block on page

with execution time

- *Comment:* Displays execution time as an HTML comment in source
- *Debug (default):* Displays execution time in debugging section in the Timer section

Depending on your output selection, it will display the execution times for each of the code blocks you decide to track in different locations. Using the default type of *debug* is great because it will leave the text out of the page and in a nice clean order with all of the `cftimer` labels in the debugging output. The *outline* output option is the most visually appealing and puts the code output in an outlined box on the page with a label name and the execution time. In order for this to display properly, your browser must be able to handle the `<fieldset>` and `<legend>` tags. The `cftimer` tag can also be nested to track overall execution time of a larger block of code versus smaller subsections within:

```
<cfoutput>
<cftimer label="Large Code Block" type="outline">
  <cftimer label="Small Sub Block 1" type="outline">
    <cfloop from="1" to="10000" index="x">
      <cfif x mod 2000 eq 0>#x# </cfif>
    </cfloop>
  </cftimer>
  <cftimer label="Small Sub Block 2" type="outline">
    <cfloop from="1" to="10000" index="x">
      <cfif x mod 2000 eq 0>#x# </cfif>
    </cfloop>
  </cftimer>
</cftimer>
</cfoutput>
```

In order to get `cftimer` to work, you need to enable debugging in the ColdFusion Administration along with turning on the “Timer Information” option. One of the benefits of being able to turn on and off the “Timer Information” setting speci-

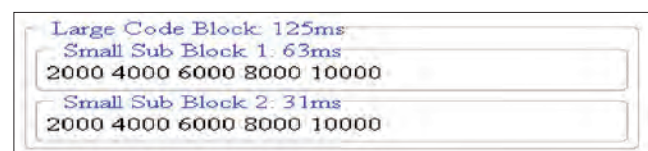


Figure 1: Output with `type="outline"`

CFTimer Times

- [15ms] *Small Sub Block 1*
- [63ms] *Small Sub Block 2*
- [94ms] *Large Code Block*

Figure 2: Output with type="debug" in debugging section

cally is that you can put all of your cftimer tags in your code during your development and when you copy them live on a production server you can turn the "Timer Information" setting off and the code will run just as it would without the cftimer outputs.

Debugging Queries

Debugging queries is easier in CFMX7 thanks to two new features: the result attribute on cfquery and the ability to use GetMetaData() on a query. Together these features give us better access to information coming back from the database and what query was run.

There is a new attribute named "result" that many tags now have such as cffile and cfstoredproc, but the most notable tag for this feature is cfquery. You use this attribute to specify the name of the structure you want to return information from the query on. This structure created from the cfquery will contain the following attributes:

- **sql:** The executed SQL statement
- **recordcount:** The number of records returned
- **cached:** Boolean value for if the query was cached
- **sqlparameters:** An array of cfqueryparam values (Only if cfqueryparam was used)
- **columnlist:** A list of the columns returned
- **executionTime:** Amount of time it took to run the query

It is worth noting that the execution time includes the amount of time it took for ColdFusion to process the cfquery tag itself plus the amount of time the SQL query actually took. You will see a different execution time for the query in the debugging information than what this structure will return, due to its tracking the processing of the actual cfquery.

Listing 1 Simple query on the book club example database

```
<cfquery name="getmembers" datasource="cfbookclub" result="strQuery">
    SELECT memberid, firstname, lastname, email, isadmin
    FROM members
</cfquery>
```

In addition to getting more information on a query with the result attribute, you can now get more detailed column-level information by using the GetMetaData() function on a query. This function will return the following information on each of the columns returned:

- **IsCaseSensitive:** Boolean value for whether or not the data is case sensitive
- **Name:** The column name
- **Type:** The SQL data type of the column

This function allows you to view the data type of the column inside your application. Depending on your database system, you could have achieved this by other means such as querying the information schema views in Microsoft SQL Server, but this func-



> **Developers:**
Finally, web content
management that's
**Powerful and
Affordable.**

Our newest partners:

CFDynamics, Lumenbrite Training, Pixelspace,
Sandstorm Design, roundpeg, Inc. and Uvault.com
— Sign up today!

> See us at CFUnited
1.866.870.6358

www.BeSavvy.com
Savvy Software Inc.



macromedia®
ALLIANCE PARTNER

struct	
CACHED	false
COLUMNLIST	EMAIL, FIRSTNAME, ISADMIN, LASTNAME, MEMBERID
EXECUTIONTIME	47
RECORDCOUNT	3
SQL	SELECT memberid, firstname, lastname, email, isadmin FROM members ORDER BY lastname, firstname

Figure 3: Output of strQuery result structure from Listing 1

array	
1	struct IsCaseSensitive NO Name memberid TypeName INT
2	struct IsCaseSensitive NO Name firstname TypeName VARCHAR
3	struct IsCaseSensitive NO Name lastname TypeName VARCHAR
4	struct IsCaseSensitive NO Name email TypeName VARCHAR
5	struct IsCaseSensitive NO Name isadmin TypeName BIT

Figure 4: Output of GetMetaData(getmembers) from Listing 1

tion makes a simple solution to work across multiple database platforms. Combined with the data from the other debugging methods such as the result attribute, you are able to get almost any piece of information you need from a query.

The array of structures is presented in order of the column names from the query, which differs from using the `queryname.columnlist` attribute from `cfquery` where the list is in alphabetical order. Having the column names in ordinal order provides simple ways to create reports or output from queries with minimal amounts of code. All you need to do is create your query and put the column names in order of your desired report and give them optional aliases. Then you can loop over information

gathered from `GetMetaData()` to dynamically create an output based upon your query.

Conclusion

The new debugging features of CFMX7 make some of the classic debugging tasks we did before a lot easier. While these features could have been achieved in previous versions by alternative methods, these new features make it easier and cleaner to achieve much more. Not only will they help debugging but they will improve and enhance application development by giving developers access to information that was previously hidden.

About the Author

Kevin Kazmierczak is the senior software developer for Citynet, a telecommunications company in Bridgeport, West Virginia. He is a Macromedia Certified Advanced ColdFusion Developer and has been using ColdFusion since 1999.

kevin.kazmierczak@citynet.net

www.linuxworld.com

Connect

Subscribe Today!

Connect online for fastest service... don't miss another issue of LWM!

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE \$71.76

YOU PAY ONLY \$49⁹⁹

12 ISSUES/YR

LOG ON TO

SYS-CON MEDIA

The World's Leading i-Technology Publisher

www.linuxworld.com

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
ACTIVEPDF	WWW.ACTIVEPDF.COM	866-468-6733	17
BLOG-N-PLAY	WWW.BLOG-N-PLAY.COM	888-303-5282	37
CFDYNAMICS	WWW.CFDYNAMICS.COM	866-233-9626	4
COLDFUSION DEVELOPER'S JOURNAL	WWW.SYS-CON.COM/COLDFUSION	888-303-5282	41
HAL HELMS, INC	WWW.HALHELMS.COM		33
HOSTING.COM	WWW.HOSTING.COM		15
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM/CFDJ	877-215-4678	19
INTERAKT ONLINE	http://www.interaktonline.com	4031 401.68.19	3
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800-379-7729	COVER IV
JAVA DEVELOPER'S JOURNAL	JDJ.SYS-CON.COM	888-303-5282	43
LINUXWORLD MAGAZINE	WWW.LINUXWORLD.COM	888-303-5282	30
MACROMEDIA	WWW.MACROMEDIA.COM/go/cfm7_demo	415.252.2000	COVER II
MACROMEDIA MAX	WWW.MACROMEDIA.COM/MAX	415.252.2000	COVER III
MACROMEDIA	WWW.MACROMEDIA.COM/go/8_studio8	415.252.2000	9
MX DEVELOPER'S JOURNAL	WWW.SYS-CON.COM/MX/SUBSCRIPTION.CFM	888-303-5282	31
PAPERTHIN	WWW.PAPERTHIN.COM	800-940-3087	23
SAVVY SOFTWARE	WWW.BESAVVY.COM	866-870-6358	29
SEAPINE SOFTWARE	WWW.SEAPINE.COM/WEBDEV	888-683-6456	6
SYS-CON E-NEWSLETTERS	WWW.SYS-CON.COM	888-303-5282	31
SYS-CON REPRINTS	WWW.SYS-CON.COM	201-802-3024	39
SYS-CON.COM	WWW.SYS-CON.COM	888-303-5282	35
WEBAPP CABARET	WWW.WEBAPPCABARET.COM/cdj.jsp	866-256-7973	11

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.

Looking to Stay Ahead of the *i*-Technology Curve?

Subscribe to these **FREE** Newsletters >

Get the latest information on the most innovative products, new releases, interviews, industry developments, and *i*-technology news

Targeted to meet your professional needs, each newsletter is informative, insightful, and to the point.
And best of all – they're FREE!

Your subscription is just a mouse-click away at www.sys-con.com

IT solutions JOURNAL

NET JOURNAL

JDA
JOURNAL OF DEVELOPERS

WebServices JOURNAL

information
STORAGE + SECURITY JOURNAL

LINUXWORLD MAGAZINE

wireless BUSINESS TECHNOLOGY

LINUX BUSINESS WEEK

XML JOURNAL

MX developer's journal

WebSphere JOURNAL

wldj THE LEADING SOURCE FOR WEBSITE PROFESSIONALS

ColdFusion Developer's Journal

SYS-CON MEDIA

The World's Leading *i*-Technology Publisher

A new tool for MX professional developers and designers...

ADVERTISE

Contact: Robyn Forma
robyn@sys-con.com
(201) 802-3022
for details on rates
and programs

SUBSCRIBE

[www.sys-con.com/
mx/subscription.cfm](http://www.sys-con.com/mx/subscription.cfm)
1 (888) 303-5282

SYS-CON MEDIA



MX
developer's journal

ColdFusion and Crystal Enterprise Integrations

A primer



By Jason Heaslet

This document is a primer. It's designed to inform you, the reader, of what can be done and what to consider before attempting integration. It's not riddled with proprietary code that describes exactly how to

do integrate between ColdFusionMX and Crystal Enterprise.

Why ColdFusion and Crystal Enterprise?

Reporting is not just some set of documents with some data on it. It can be so much more. Reporting is at the core of every business. Most developers believe that their applications drive their business. However, reporting *is* the core deliverable and it's the very heart of every business. Reporting is the intelligence of how your business is running and how you can streamline it to run more efficiently. With that in mind, your applications are mechanisms of what your business does. They allow you to perform your job more efficiently. They are tools that allow you to capture your data.

Now, more than ever, the development community is moving into a new era of rich media. The general consensus of the ColdFusionMX and Legacy ColdFusion community is that reporting is a beast best left undisturbed, or that they can produce metrics faster and quicker via a few well-written `<CFQUERY>` and `<CFOUTPUT>` tags that will deliver a sparse few details in html. However, contrary to the popular belief of most developers, reporting is a vast and useful topic. Companies spend millions on reporting solutions that provide everything from Web metrics, to trending and analysis, to financial reports. This is not something that most ColdFusionMX or Legacy ColdFusion developers are comfortable with. Allaire's (now Macromedia) attempt at including Crystal Reports through the `<CFREPORT>` tag was all but futile, as most developers wouldn't use it, choosing to write their metrics pages through `<CFQUERY>` and `<CFOUTPUT>` tags.

With Business Objects (previously Crystal Decisions) pushing its Crystal Enterprise product, this merely exacerbated the subject. Crystal Decisions licensed Microsoft's ASP technology and created a new twist on it,

delivering CSP (Crystal Server Pages) within its Crystal Enterprise product. The vast majority of Internet developers, especially ColdFusionMX developers, have yet to successfully integrate their application environments and their reporting environments. It has, for the most part, always been that you're either in the application environment or in the reporting environment.

ColdFusionMX v7 was released with a new banded report writer as well as various other reporting oriented features. The CF community rejoiced. However, those that rejoiced because they can now abandon Crystal for CFMX 7 will have to continue to use Crystal. The reason being: Crystal is an expensive undertaking and most companies will not abandon a solution that they have so much invested in.

Corporate and Application History

As most ColdFusion developers know, ColdFusion was originally developed by Allaire and Allaire was later acquired by Macromedia. ColdFusion is in its seventh version and has moved from a C++ platform to a Java platform. Crystal on the other hand has a longer history. It's currently in its 11th version and has changed hands several times. The part of the story that I know, beginning from the time I started interfacing with the tool, is that it was sold to Seagate. It was then sold back to the original company (who renamed itself Crystal Decisions) and subsequently acquired by Business Objects.

The Difference Between Crystal Reports and Crystal Enterprise

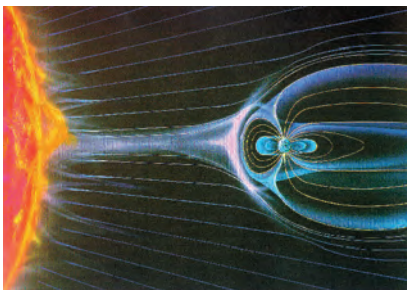
When people start talking about "Crystal" they refer to it in two different aspects:

1. Crystal Reports
2. Crystal Enterprise

Crystal Reports is a banded report writer in its 11th version. Its features are numerous and robust. Crystal Enterprise is a culmination of server-based services designed for the deployment and

delivery of reports and reporting, primarily developed through either the Crystal Reports IDE or "on the fly" using CSP. As I said previously, Crystal Decisions licensed Microsoft's ASP technology and created a new twist on it, delivering CSP (Crystal Server Pages) within its Crystal Enterprise product.

I have found that Crystal Reports developers and Crystal Enterprise developers don't normally travel in the same



circles. Report developers, in my experience, know a lot about displaying data and how Crystal Reports, as a banded report IDE, functions. In general they have not been exposed to the server-side aspect of development. Conversely, I've found that Crystal Enterprise developers are well versed in the Crystal Enterprise APIs, server architectures, and performance tuning.

I have run across a few that are very good at both, but they just seem to be few and far between. My personal experience leans toward Crystal Enterprise, rather than actual report development. I prefer server-side API development and performance tuning. I'm fairly well versed in Crystal Report development, but I have found that there are many other developers who know the IDE better than I do. I consider it a presentation layer and prefer to continue my development from a server-side aspect. However, knowing its strengths and weaknesses is definitely an asset.

Enter the ColdFusion Developer

Cold Fusion developers have always been told either you can't do ColdFusion and Crystal, or that they need to use the <CFReport> tag. Both are total disasters, of course. The new reporting-oriented features of CFMX 7 are really nice. I've got to hand it to Macromedia, Tim Buntel, Damon Cooper, and team. They have put together a rock-solid set of features (such as CFDocument) that I have put to good use and look forward to them strengthening in the future.

The reason I mention it in this way is, while the banded report tool is nice and robust and integrated, it is still a 1.0 release. Business Objects, on the other hand, has had 11 versions to get it right. Although I feel they still could improve, it's a far cry from version 1.0. That leads me to the statement that my father drilled into me at a young age: "Use the right tool for the job."

If I am going to write a report that's not too complex, or maybe has a little drill through or drill down, I'll likely use CFMX 7 and one of the new report features, like the banded report writer or the CFDocument tag. I'll wrap it up into a PDF or FlashPaper. However, if I need to write a report that has many layers, sub layers, sub reports, or I want to use a code repository and I need to manage my reports, or any combination thereof, I'm more than likely going to use Crystal. I can probably work around those issues. What I can't work around is something I mentioned earlier. Often, I need to deploy through a system that the company has a huge investment in. I evaluate each deployment strategy individually. I use Crystal when I need to do some heavy lifting. I use CFMX 7 when the reporting isn't so intensive. As I said, use the right tool for the job. Crystal Enterprise has a huge strength in report deployment. It has been designed around crunching reports that, in some cases, may take days to complete. It can deliver you the first page in a report while it is still building the remaining pages. It schedules, maintains, manages, and secures documents and reports and does it all natively.

Consider Your Licensing

Licensing is, of course, a very serious matter with all software. However, it's especially important when integrating two different pieces of software. In this specific case, we need to consider the licensing aspect of Crystal Enterprise. There are

three types of licensing you can use:

1. Named User
2. Concurrent User
3. CPU

If you purchased Crystal Enterprise prior to January 1, 2005, and you own concurrent licenses for it, you can still purchase them. However, if you don't already own concurrent licenses, you can no longer buy them. Concurrent licensing was designed around the fact that two users of the same account are unlikely to be requesting data at the same time. Therefore, the serviceable ratio was approximately one concurrent user per 10 real users.

A named user is just that, one named user. There can only be one user per account and that user can only be logged in from one machine at any one time. CPU licensing licenses the servers per CPU. A CPU license needs to be purchased for every CPU on the machine. When installing a CPU license, you can consider it to be unlimited concurrent licensing. Setting up CPU licenses for performance is a bit of a trick. It can be done, but really needs to be done by someone who not only knows how Crystal Enterprise behaves, but also, after evaluating the reports, what needs to be delivered. Business Objects does have a document that explains all this but they will force you to sign an NDA to get it.

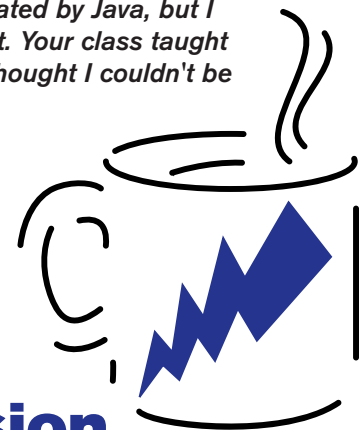
Crystal Enterprise has about 10–14 services that can be

"I was totally intimidated by Java, but I knew I had to learn it. Your class taught me what I honestly thought I couldn't be taught." - Sharon T

Java for ColdFusion Programmers?

Java for ColdFusion Programmers, the five-day Hal Helms, Inc. training class is designed for ColdFusion programmers; no previous Java experience is needed. You'll learn how to think in objects and program in Java.

For class information and registration, come to halhelms.com.



installed. The services are considered servers in their own right by Business Objects. The system is designed to be easily scaled outward depending on your growing needs. The trick is that if you purchase CPU licenses, you need to ensure you have enough to manage your scalability; last I checked they were approximately \$60,000.00 USD. ColdFusionMX licensing on the other hand is fairly simple. You can purchase CFMX as standard or enterprise packages.

Standard edition gives you the standard CF Server, like you've always used. In its current version, it's a modified version of JRun 4 that drives only the CFMX tag set.

The Enterprise package yields not only the core server, but also a full version of JRun 4, multiple server instances, J2EE application server deployment, event gateways, enterprise server security, sourceless deployment, archive and deployment services, high-performance e-mail delivery, high-performance reporting and document generation, asynchronous processing, and server/instance clustering.

Deployment Strategies

When deploying a CFMX to Crystal Enterprise integration, I recommend seriously evaluating the needs of both environments. A Performance Analysis and Tuning engagement (PA&T) from Macromedia would be a good idea. The reason I recommend this is that integration will *appear* to stress your server.

The stress that you will see will be in relation to the communication between the two applications. It could be your network, connectivity, virus software, firewalls, a few other items, and or any combination of the previous. A major reason why people have said previously that it could not be done was due to these issues and not that the applications wouldn't talk to each other.

Actually, in its simplest, they don't really talk to each other. Crystal talks to itself. CFMX grabs the Crystal services and shakes it around and tells it what it's going to do... like a bully. The first key is that everything requires authentication. The second key is to know how to quiz Crystal Enterprise to give it what you want, after it's authenticated.

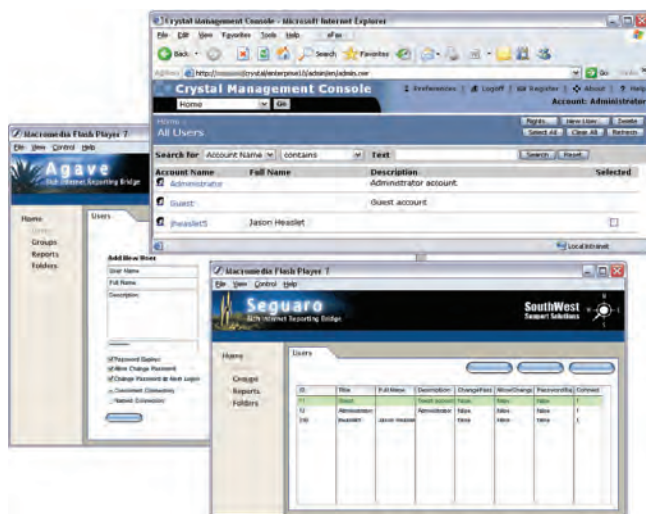


Figure 1: Agave and Seguario Prototypes with Crystal Enterprise's Management Console


Agave and Seguario

The results of this are two application prototypes: Agave and Seguario. I've only mentioned my development of these prototypes to a handful of people, until now.

Agave was the original prototype that integrated ColdFusion 4.0 to a Crystal Enterprise 8.0 implementation. The goal was to re-create Crystal Enterprise's ePortfolio using ColdFusion. With my development of Agave, I can add/edit/delete users, folders, and groups. I can add users to groups and/or folders. I can modify a user's security in any combination thereof. More important, I can authenticate that user to deploy a report through Crystal Enterprise. Crystal Enterprise then gets to do what it does best: render and deliver the report. I stopped there with Agave; I had new aspirations with what I could do with this success. The testing stretched from CF 4.0 through CFMX 7 into Crystal Enterprise versions 8.0 through 11.0.

Seguario was originally designed utilizing CFMX 6.0 and, later, 6.1. It's been integrated into Crystal Enterprise versions 8.5 through 11.0. My development of Seguario was built using a homegrown MVC model. I started using CFCs in the testing and, later, in Web services. My intention was to use Flash to rebuild ePortfolio. Using ColdFusionMX I can back-end a Flash UI. Using Flash data grids and XML connectors (I actually put that stuff I learned at MAX2003 to use), I successfully replicated the same features I had developed in Agave.

There are still many features that have not yet been built. However, some of the functionality that have been built thus far are some functions utilizing Crystal Enterprise's RAS server to provide some ad hoc reporting, report scheduling, and report template manipulation. I have even been able to pull a report deployed by Crystal Enterprise and render it in Flash using FlashPaper (it's not exactly a recommended practice but it can be done).

These two prototypes (Agave and Seguario) are now the foundation I use when building most integrations. The reason is, as I consulted to provide this type of service, I learned that each integration is unique. As you all know, everyone has their own ideas about how each application should get implemented. It's amplified when you start integrating existing applications. Everyone has built applications differently. Thus each set of tools that needs to be deployed is deployed on an as-needed basis. The result is that I provide Agave and Seguario licenses as part of the service I provide. 

About the Author

Jason Heaslet is a long-time ColdFusion application developer and consultant. He has been working with ColdFusion since his days at NASA as the Web Master for Space Shuttle Vehicle Engineering Office (SSVEO). Jason now architects and builds "best of breed" applications and currently spends most of his waking hours either fiddling with CFMX 7, its reporting features, and how to integrate it into Crystal Enterprise, or tinkering with some new Flash Lite application (that is, when his lovely wife and son aren't pulling him kicking and screaming from the keyboard).

jason@heaslet.net



Visit the *New*
www.SYS-CON.com
 Website Today!

The World's Leading *i*-Technology
 News and Information Source

24/7

FREE NEWSLETTERS

Stay ahead of the *i*-Technology curve with
 E-mail updates on what's happening in your industry

SYS-CON.TV

Watch video of breaking news, interviews with industry leaders, and how-to tutorials

BLOG-N-PLAY!

Read web logs from the movers and shakers or create your own blog to be read by millions

WEBCAST

Streaming video on today's *i*-Technology news, events, and webinars

EDUCATION

The world's leading online *i*-Technology university

RESEARCH

i-Technology data "and" analysis for business decision-makers

MAGAZINES

View the current issue and past archives of your favorite *i*-Technology journal

INTERNATIONAL SITES

Get all the news and information happening in other countries worldwide

JUMP TO THE LEADING *i*-TECHNOLOGY WEBSITES:

<i>IT Solutions Guide</i>	<i>MX Developer's Journal</i>
<i>Information Storage+Security Journal</i>	<i>ColdFusion Developer's Journal</i>
<i>JDJ</i>	<i>XML Journal</i>
<i>Web Services Journal</i>	<i>Wireless Business & Technology</i>
<i>.NET Developer's Journal</i>	<i>Symbian Developer's Journal</i>
<i>LinuxWorld Magazine</i>	<i>WebSphere Journal</i>
<i>Linux Business News</i>	<i>WLDJ</i>
<i>Eclipse Developer's Journal</i>	<i>PowerBuilder Developer's Journal</i>

What's an Object?

An introduction to object-oriented programming for ColdFusion developers



By Matt Woodward

On New Year's Eve, 2004 I declared 2005 to be the "year of object-oriented programming for ColdFusion developers," and since the year is approaching its final quarter it's a good time to focus our atten-

tion on OOP in ColdFusion and see how we're doing.

Based on talking with developers both in person and virtually, reading blogs, and looking at some of the newer ColdFusion code that people have been sending my way over the last few months, I'm really excited about the increase in interest and use of OOP in ColdFusion. The ability to even do object-oriented programming in ColdFusion is a relatively new addition to the technology, so it's not surprising that it's taken a bit of time to catch on. From what I'm seeing these days the tide is definitely turning, so this is great to see.

What I also see occurring, however, is an increasing gap between the ColdFusion developers who are embracing OOP wholeheartedly and those who want to dive into OOP but don't quite know how to go about doing so, or have gotten started but get frustrated or discouraged and go back to their old ways of doing things. This is lamentable but understandable. Coming from the procedural mindset that many ColdFusion developers have, making the move to OOP is not something that will happen instantly, and in many cases developers are under deadlines and just can't afford to ditch their tried-and-true methods for OOP if they don't yet have a comfort level with it.

With that in mind, this article is designed to give ColdFusion developers an extremely gentle introduction to some OOP fundamentals. In working with other developers I've found that these concepts have helped many of them make the transition from ColdFusion 5-style procedural coding into OOP in CFMX 6.1 and 7. This is a transition that in my opinion ColdFusion developers can't afford not to make. Object-oriented programming has been around since the 1960s, all modern languages support OOP to varying degrees, and the ubiquity of OOP is not accidental; it really is a better way to build applications. I hope by the end of this article you'll agree with me and start to see the OO light.

ColdFusion's OO Conundrum

If you're reading this chances are I don't need to convince you of the tremendous strengths of ColdFusion: ease of use, huge feature set, rapid development, integration with Java.... These are all great things, and ColdFusion 7 made things even better with the addition of the cfdocument tag, Flash forms and XForms, event gateways, and a great deal of other new features that help us build better applications more quickly and easily.

With all that having been said, and I don't want to scare anyone, I truly believe that if ColdFusion – and by extension ColdFusion developers – is going to survive over the long haul, we absolutely must get with the times and start down the path of truly understanding OOP and using it as the default way we do ColdFusion development. Every other major Web application development platform is fully OO or rapidly getting there, so the days for excuses are over. OOP has become king of the software development world for a reason, and I promise that once you truly understand it you'll wonder how you ever got along without it. It's quite simply a better way to solve real-world problems through programming.

First and perhaps foremost, I think it's quite unfortunate that most of the ColdFusion literature dives right into CFML syntax and spends a great deal of time illustrating how to build entirely page-based, procedural applications, only discussing ColdFusion Components (CFCs) and OOP as an afterthought. Since OOP wasn't really even possible prior to the introduction of CFCs in CFMX, and even then a lot of the huge quirks weren't worked out until CFMX 6.1, I suppose this situation shouldn't be surprising.

However let's consider where we are today. CFMX 6.1 is now the "old" version and we have a fantastic new release with ColdFusion 7, so it's high time that we start using the tools available to us to their full potential. I only hope that the ColdFusion literature begins to reflect this mode of thinking as well, and I think that will happen given the huge upswing in interest in OOP I've seen over the last few months.

As a bit of an aside, I'm going to forgo the "Is ColdFusion/ Are CFCs fully OO?" argument because I think it's largely irrelevant to the discussion at hand. While it is true that CFCs lack certain characteristics of objects in fully-OO languages such as Java, that's certainly not a reason to not use them to do OOP in ColdFusion. First we all need to understand what OO is and start using CFCs to do OO programming in ColdFusion, then we can talk about how CFCs differ from something like a Java object and address any limitations, tricks, and traps related to these issues.

If you pick up any good Java book (I'll have a few good recommendations at the end of this article) it will start with a discussion of objects and basic OOP concepts before you even see much Java code. Better still, there are a couple of good language-agnostic OO books that can provide a solid grounding in OO concepts before you even start writing any code. With OOP it's extremely important that you have a firm foundation in the fundamental concepts before you jump right into development, especially if you have a completely procedural programming background.

At the outset, thinking in objects is in my estimation far more important than understanding how to implement these concepts in code. In the first Java class I took at Sun Microsystems we didn't write a single line of code, we just spent an entire week talking about objects and approaching problem solving in an object-oriented fashion. This was all rather abstract and didn't involve a single line of Java code, but it's still the best Java class I ever took.

While ColdFusion's audience is admittedly a bit bifurcated when compared to Java's, I still believe it's of the utmost importance both for us as developers and for the reputation of ColdFusion as a technology that we make OOP the de facto way we develop applications in ColdFusion. One of ColdFusion's biggest strengths is that it's easy to learn and allows even non-programmers to get results very quickly. This is also one of its biggest weaknesses. The challenge, then, is to somehow keep the ease of use/rapid development reputation while strengthening the notion that CF is an extremely capable development platform that can be used to build enterprise-level applications using the same methodologies as the two giants of the Web application development world, namely Java and .NET. I don't have a solution for this conundrum yet, but I do believe that making OOP an inextricable part of CF development is a big step in the right direction.

As I climb off my soapbox, try to put your mind in a place that might be new to some of you. Imagine if you will a world where there is no procedural programming. In this world all software is built with objects, and a software

application is nothing more than objects communicating with one another. (I'll focus more on the distinctions between procedural programming and OOP in a future article.) At the heart of this style of development of course is the concept of what an object is, so let's address that question before we go any deeper.

What's an Object?

I'm glad you asked that question! I've talked with a lot of ColdFusion developers who are intrigued by the idea of objects and OOP, but they just don't even know where to begin. To quote Maria von Trapp, "Let's start at the very beginning, a very good place to start." The beginning of object-oriented programming is of course the concept of the object. To be completely truthful there is a bit of a chicken-and-egg situation between the concept of a class and the concept of an object. For the moment let's simply focus on explaining what an object is, not only because they don't call it "Class-Oriented Programming," but because I think you'll find it easier to understand the concept of a class if you first understand what an object is.

To put it extremely simply, you already know what an object is. Objects are all around you. If you look at the computer you spend countless hours a day using, that will serve as a good example of what an object is in the real world. One of the amazingly powerful things about software objects is that they're designed to model real-world objects, and do quite a good job of it!

Your computer is an object, and as an object it has certain characteristics such as brand, processor type and speed, amount of RAM, hard drive size, etc. In object lingo, these characteristics are called *attributes*. Your computer can also do things (sometimes it doesn't do what you want it to do, but it's still doing something!). It can power up, launch applications, automatically run tasks such as a virus scan, and power down. In object parlance these things that your computer can do would be referred to as *methods* (or sometimes they're referred to slightly less accurately, at least from an OOP standpoint, as *functions*).

In addition to attributes and methods, perhaps the most important characteristic of an object is that it's *self-con-*

THREE REASONS TO

blog-n-play.com

1 Get instantly published to 2 million+ readers per month!

blog-n-play™ is the only **FREE** custom blog address you can own that comes with instant access to the entire i-technology community. Have your blog read alongside the world's leading authorities, makers and shakers of the industry including well-known and highly respected i-technology writers and editors.

2 Own a most prestigious blog address!

blog-n-play™ gives you the most prestigious blog address. There is no other blog community in the world that offers such a targeted address, and comes with an instant targeted readership.

3 Best blog engine in the world...

blog-n-play™ is powered by **Blog-City**™, the most feature rich and bleeding-edge blog engine in the world, designed by Alan Williamson, the legendary editor of **JDJ**. Alan kept the i-technology community bloggers' demanding needs in mind and integrated your blog page to your favorite magazine's Web site.



www.TAMI.linuxworld.com

"Many blogs to choose from"

PICK YOUR MOST PRESTIGIOUS ADDRESS

IT Solutions Guide	MX Dev. Journal
Storage+Security Journal	ColdFusion Dev. Journal
JDJ: Java	XML-Journal
Web Services Journal	Wireless Business &Tech.
.NET Dev. Journal	WebSphere Journal
LinuxWorld Magazine	WLDJ: WebLogic
LinuxBusinessWeek	PowerBuilder Dev. Journal
Eclipse Dev. Journal	

3 MINUTE SETUP

Sign up for your FREE blog Today!

blog-n-play.com
i-Technology Blogs Read by Millions

www.blog-n-play.com

— This site will go beta February 15, 2005!

tained. As I said earlier I'm not going to dive into the differences between procedural programming and OOP at this point because I think that's a bit difficult to understand without really understanding what an object is, but the notion of a self-contained object is important to address. In procedural programming there basically exists data and functions that work with this data, but the data and functions are typically separate and distinct from one another. With an object, the data (attributes) and functions (methods) are all contained within a single, tidy little software component known as the object. This offers numerous advantages that we'll delve into in a future article.

I'm a Person, Not an Object!

Let's consider another, very commonly used example to shed more light on the concept of objects, and this one you'll probably use more regularly in your applications: a person. A Person object, like the Computer object, also has *attributes* (for example, first and last name) and things they can do, or *methods* (for example, "walk"). So while you might not prefer to be thought of as an object, as far as the software world is concerned a person is a rather ideal example of an object. If you don't take too much offense let's proceed with this example for now.

Recalling that the three most important aspects of an object are attributes, methods, and that the object is self-contained, we're now going to look more specifically at how we'd go about modeling a person as an object in our applications. First, let's make a basic list of some of the more important attributes that a person has:

- first name
- last name
- birth date
- sex
- height
- weight

Now, let's make a list of some of the things a person can do. These translate into our methods:

- walk
- run
- sleep
- work

- talk
- write ColdFusion applications

You get the idea. Attributes and methods are what define an object.

A Touch of Class

Earlier I mentioned the term *class*, and before we go much further we need to touch briefly on what a class is. Now that you understand at least in a basic sense what an object is you'll grasp the concept of class pretty easily. A class can be thought of as a generic version of an object. Continuing with our Person object discussion, think of the Person *class* as a generic person. You don't know if this person is male or female, young or old, short or tall, it's just a nameless, faceless, abstract person.

In OOP you use this generic Person class as the blueprint for creating specific Person objects. The Person *class* knows that a person has the attributes and methods listed above but knows nothing specific about a particular person. In other words, the Person class knows that a person has a first name, but it doesn't know any particular person's first name. The Person *object* will know specific details about a particular person, such as "This particular Person's first name is Matt."

You create a specific Person object, which is also known as an *instance* of the Person class, by *instantiating the class*. This is a fancy OO way of saying, "Let's take this generic Person class and make it into a specific Person object." For argument's sake let's assume we want to create a Person object called "matt." (I could always use another one of myself to help me get things done.)

In terms specific to ColdFusion, objects are defined using ColdFusion Components (CFCs). The Person class would be a CFC called Person.cfc. To instantiate this class in ColdFusion, you

would either use the cfoject tag or the CreateObject function as follows:

Using the cfoject tag:

```
<cfoject component="Person" name="matt" />
```

Using CreateObject (this could be within a cfscrip block or you could do this within a cfset tag as well):

```
matt = CreateObject("component", "Person");
```

There are some details you'll need to know regarding where ColdFusion looks to locate your CFCs, but don't concern yourself with the specifics of implementation at the moment. Just repeat this phrase over and over again: "I'm thinking in objects. I'm thinking in objects." We'll get to a specific implementation at the end of this article.

So at this point we have a specific Person object called matt with which we can start working. When the matt object is first created, however, it doesn't necessarily know anything about itself. There are a couple of standard ways to get data into our object. You might think we can just start setting our object's attributes using dot notation as follows:

```
matt.firstName = "Matt";
matt.lastName = "Woodward";
```

Not so fast! You may recall from the discussion above that one of the hallmarks of an object is that it's self-contained. What does this mean? In addition to meaning that the object's data and the methods for working with the data are both contained within the object, it also means that you should never (and in most cases, well-designed objects won't let you) set an object's data directly as illustrated above. A good object will contain methods that will allow other objects to access its data as needed, and the polite OO behavior is to use these methods – and only these methods – to interact with the object. You don't want to be seen as rude on your first visit to ObjectLand, do you?

Don't Touch My Data... At Least Not Without Asking

Only using an object's methods to access its data brings up another OO concept called *encapsulation*. In brief, encapsulation refers to



the situation I described above whereby you only interact with an object through the methods it exposes to you. For something simple such as retrieving a Person object's first name this may seem like overkill (why not just grab the data directly?), but let's consider another example. If you were to have an object that contained a method that had to perform relatively complex operations to return the requested data to you, you shouldn't have to know (and probably don't even want to know!) what the object is doing to return that data.

In this sense an object and its use of encapsulation are like a telephone. You just plug the phone into the wall, pick up the receiver, and you get a dial tone. You don't know or care about what's going on in the wiring, switching, etc., you just want to get a dial tone. This is exactly what encapsulation does. It provides an interface for you to use to get what you need without requiring you to know what's going on behind the scenes. This is important for simplicity and ease of use, to protect the object's data, and to keep things from breaking in the future. Imagine if you had to reconfigure your telephone every time the phone company got a new piece of hardware on their network!

Encapsulation protects you from having to modify your code if the implementation of a particular operation in an object changes. This could be as simple as changing the way a value is calculated or as complex as changing from a relational database to an XML-based data storage system. Regardless of the specifics involved, encapsulation will keep your OO systems humming right along even if a relatively drastic change has to be made to an object within the system.

Beans: They're Not Just a Breakfast Food (with Apologies to Johnny Cash)

To summarize the idea of encapsulation, any code that interacts with an object should never modify the object's data directly. In procedural programming variable values such as someone's name are modified directly, but as we'll see in a future article this leads to problems that we can avoid with good OOP. This is why a best practice in OOP involves the creation and use of methods within objects known as *getters* and *setters* (also referred to as *accessors* and *mutators*). These are simple methods that allow us to retrieve and set values of data within objects through method calls, as opposed to accessing the data directly.

If an object has getters and setters for each of its attributes, it is actually a specific type of object known as a *bean*. The bean concept comes from the Java world, where everything is related to coffee in one way or another, so I suppose that's as good an explanation as any for the name. A bean is a type of object that is very commonly used to represent real-world constructs such as a Person, Computer, or Car. There are of course other types of objects you'll be using in OOP, but in my experience if people grasp real-world objects (which in turn are usually beans in OO systems), it's a lot easier to start working with other types of objects as well. The CFC example we'll see at the end of this article will technically be a bean.

Returning to the example above, after we create our matt object we'll want to set the values for matt's attributes so

REPRINT!

Once you're in it...



...reprint it!

- ColdFusion Developer's Journal
- Java Developer's Journal
- Web Services Developer's Journal
- Wireless Business & Technology
- XML Journal
- PowerBuilder Developer's Journal
- .NET Developer's Journal

Contact Dorothy Gil
201 802-3024
dorothy@sys-con.com

REprints

SYS-CON
MEDIA

the object is no longer an empty shell of a Person. You've probably already surmised that since we can't access the object's data directly, we're going to be using the object's get and set methods to interact with the matt object. Let's tell matt what his first and last names are by using the object's setters for these attributes:

```
matt.setFirstName("Matt");
matt.setLastName("Woodward");
```

As you can see, the set methods take a single argument, a string, to set the values of first and last name. Once these values are set, we can then turn around and ask the matt object what its first and last name is by using the object's get methods:

```
<cfoutput>The matt object's first name is #matt.
getFirstName()#, and
the matt object's last name is #matt.getLast-
Name()#.</cfoutput>
```

I hope you're already starting to see how this can simplify many of your programming tasks. Once an object is instantiated and the data is populated within the object (and even that step isn't terribly complex), you simply ask the object for its data and can use it wherever it's needed. This makes for clean, flexible code that's easy to use and maintain.

Object Constructors

The final concept I'll address in this article before moving on to the specific code for our Person CFC is the concept of a *constructor*. A constructor is a method within a class that provides the means to construct the object. What this construction operation usually involves

is setting any default data that's absolutely necessary for the object to function correctly once it's instantiated.

In the case of the bean object we're discussing, you can use this constructor method to create the object and set all of the attribute values rather than setting each attribute's value individually. Most objects will have what's called a "no-arg constructor," which means that if you want to instantiate the object but don't have the object's attribute values right at that particular moment, you can simply create the object, pass the constructor method no arguments or data, and the constructor will handle anything that's absolutely essential for that object to exist.

At this point we've hit upon one of the first differences of CFCs as compared to other OO languages such as Java. CFCs do not have the concept of a constructor. In Java, to create a new object you typically do something like this:

```
Person matt = new Person();
```

Because in Java the constructor method shares the same name with the object itself, that Person() call will invoke the object's constructor method. Let's revisit how we created our CFC objects earlier:

```
matt = CreateObject("component", "Person");
```

At this point all we've done is create an object, but we haven't called a constructor. Or have we? Although CFCs don't have formal constructors, they do have what's usually referred to as a "pseudo-constructor." When you create a CFC, any code that's placed after the opening cfcomponent tag but outside

any cffunction tags will get executed. We'll look at the code for our Person CFC in a moment, but here's a short example:

```
<cfcomponent>
    <!-- this is the pseudo-constructor -->
    <cfset variables.firstName = "Matt" />
    <cfset variables.lastName = "Woodward" />
    <!-- end of pseudo-constructor -->

    <!-- object methods begin here -->
    <cffunction name="doSomething">
        etc. ...
    </cffunction>
</cfcomponent>
```

Now obviously you aren't going to want to set all of your Person objects' names to "Matt Woodward" as they're created, but I hope you understand the concept. The CFML code in the area after the opening component tag but before the first cffunction tag will get executed when you create an instance of this CFC by using cfoject or CreateObject().

What has become considered a bit of a best practice in OO development with CFCs is not to rely on this pseudo-constructor, but rather to provide a method in the object called init() that is used as a constructor for the CFC. This allows for an explicit call to a constructor that is more along the lines of what happens in other OO languages. So while you could still create the object without calling the init method, in many cases you would do something like this:

```
matt = CreateObject("component", "Person").
init("Matt", "Woodward");
```

This would create the object and immediately call the object's init method. By passing the init method a first and last name, in this case the object subsequently calls the setters for the firstName and lastName attributes in the object (we'll see specifically how this works in a moment). Bear in mind you could also call the init method and pass it no arguments and the object would be created; it just won't know its name or anything else about itself until we set those values. This will all be coming together shortly!

“Object-oriented programming has been around since the 1960s, all modern languages support OOP to varying degrees, and the ubiquity of OOP is not accidental; it really is a better way to build applications”

Subscribe Today!

SAVE 16%

12 Issues for **\$89⁹⁹**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE



- Exclusive feature articles
- Latest *CFDJ* product reviews
- Interviews with the hottest names in ColdFusion
- Code examples you can use in your applications
- *CFDJ* tips and techniques

That's a savings of \$29.89 off the annual newsstand rate. Visit our site at www.sys-con.com/coldfusion or call 1-800-303-5282 and subscribe today!

ColdFusion Developer's Journal



The Person CFC

If you're with me so far, you're well on your way to understanding objects and being able to start using them in your ColdFusion applications. Armed with all your newfound OO knowledge, let's take a look at the specific implementation of the Person CFC. What I won't cover in this article is how to use this CFC in the context of a larger application; I don't want to muddy the waters with how you will throw objects around in your applications just yet.

Let's briefly review the important object concepts before taking a look at the specific CFC code:

- Objects have attributes (e.g., first name and last name)
- Objects have methods (e.g., `getFirstName()`, `setFirstName()`)
- Objects are self-contained, meaning the object's data and the methods that work with the data are all contained within the object itself
- Objects use encapsulation, meaning you will only interact with the object via the methods that it exposes to you
- Objects contain a constructor method that is called to construct the object (set default attribute values, etc.)
- Objects are your friend and you will use them whenever possible in your ColdFusion applications (sorry, had to throw that in here!)

And now, without further delay, let's take a look at the `Person.cfc` code (see Listing 1)! We're going to limit our CFC's attributes to just first name and last name, and there's going to be some "stuff" in here that we won't concern ourselves with at this stage, but we'll look over the CFC's anatomy after you peruse the code for yourself. See if you can pick out all of the concepts we've discussed and how they're implemented in this CFC.

If we want to create an instance of this object with the first name "Matt" and the last name "Woodward," we'd just use the code we outlined earlier, but now that you've seen the CFC code you'll have a better understanding of what's happening:

```
matt = CreateObject("component", "Person").
init("Matt", "Woodward");
```

This creates the object, and since

we're calling the `init` method and passing it a first and last name, the `init` method in turn calls the setters for these attributes within the object.

Finally, since the `init` method returns the object itself, the "matt" variable becomes an instance of the Person object with the first and last name attributes set. How many items from our object concept list did you pick out? Let's walk through a few of them.

1. Our object has attributes, specifically `firstName` and `lastName`. While a real-world object would typically have a lot more attributes than this, these suffice to illustrate the concept.
2. Our object has methods, five to be exact. We have an `init` method that serves as a constructor, and a pair of `get` and `set` methods for each of our two attributes. The `get` methods simply return a string for their respective attribute, and the `set` methods take a string as an argument and set the attribute's value to the value of the argument passed to the method.
3. Our object is self-contained. The object's attributes (data) and the methods that interact with this data are all contained within the object itself.
4. Because we have getters and setters and no publicly accessible data (we'll discuss public, private, and surrounding issues in a future article), we're using encapsulation. You can only get and set the attribute values via the methods that the object provides to you for this purpose.
5. Our object contains a constructor, which in this case is our `init` method. The two arguments `firstName` and `lastName` are marked as optional so we can either call `init()` and not pass it any arguments, in which case we'd get an object back with no value for `firstName` and `lastName`, or we can pass it arguments and the `firstName` and `lastName` attributes are set accordingly.

Last, objects clearly *are* your best friend and you *will* be using them whenever possible in your ColdFusion applications! Even if you're not convinced quite yet, I hope you can at least see the tremendous potential for the use of objects


in your applications. Once you really understand objects and get the hang of how to use them, it will make your development work much easier, your applications far easier to maintain, and you'll be writing your ColdFusion applications using the same methodologies as the rest of the software development world.

Looking Ahead...

I hope this relatively high-level overview of what an object is and how you apply object concepts in the ColdFusion world has piqued your interest in the wonderful world of object-oriented programming. While I can't promise that it will bring you vast wealth and make you more attractive to others, I can promise that it's a better way to develop ColdFusion applications. Even if it might seem a bit complex at first, after you've been working with objects for a bit you'll start to see the complexity fall away and you won't be able to imagine how you ever developed without them.

Please feel free to e-mail me if there are specific things you'd like to see addressed in future articles, but I think a logical next step will be to address some of the specific gotchas you'll need to be aware of as you start using CFCs as objects, and to show you how to use this Person CFC in a real-world application. Stay tuned for another installment in the near future!

Resources

- Sierra, Kathy and Bert Bates. *Head First Java*. O'Reilly, 2004.
- Taylor, David A. *Object Technology: A Manager's Guide*. 2nd ed. Addison-Wesley, 1998.
- Weisfeld, Matt. *The Object-Oriented Thought Process*. 2nd ed. Sam's Publishing, 2004. 

About the Author

Matt Woodward is Technical Architect and Lead Developer for Realty InfoLinks in Dallas, Texas. He is a Macromedia Certified ColdFusion Developer, a member of Team Macromedia, and has been using ColdFusion since 1996. In addition to his ColdFusion work Matt also develops in Flex, Java, and PHP.

mpwoodward@mac.com

Listing 1

```
<cfcomponent displayName="Person" output="false"
  hint="I am a Person object">
  <!-- constructor method -->
  <cffunction name="init" access="public" output="false"
    returnType="Person" hint="Constructor for the Person CFC">
    <!-- arguments for the constructor; these are all optional
    so this can function as a no-arg constructor -->
    <cfargument name="firstName" type="string" required="false"
      default="" />
    <cfargument name="lastName" type="string" required="false"
      default="" />

    <!-- call the setters for firstName and lastName -->
    <cfscript>
      setFirstName(arguments.firstName);
      setLastName(arguments.lastName);
    </cfscript>

    <!-- return this object -->
    <cfreturn this />
  </cffunction>

  <!-- getters and setters (a.k.a. accessors and mutators) -->
  <cffunction name="getFirstName" access="public" output="false"
    returnType="string"
```

```
    hint="Returns this object's firstName">
    <cfreturn variables.firstName />
  </cffunction>

  <cffunction name="setFirstName" access="public" output="false"
    returnType="void" hint="Sets this object's firstName">
    <cfargument name="firstName" type="string" required="true" />

    <cfset variables.firstName = arguments.firstName />
  </cffunction>

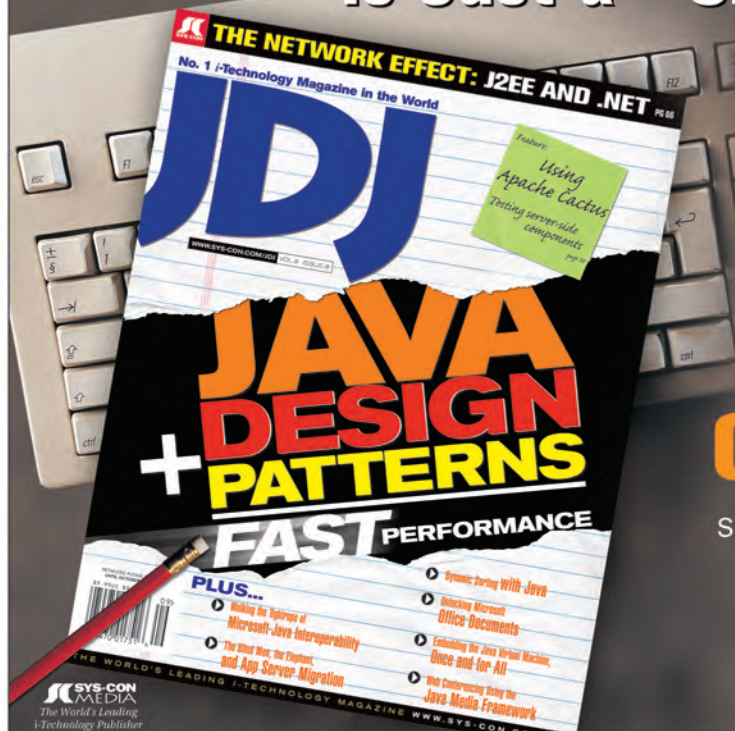
  <cffunction name="getLastName" access="public" output="false"
    returnType="string" hint="Returns this object's lastName">
    <cfreturn variables.lastName />
  </cffunction>

  <cffunction name="setLastName" access="public" output="false"
    returnType="string" hint="Sets this object's lastName">
    <cfargument name="lastName" type="string" required="true" />

    <cfset variables.lastName = arguments.lastName />
  </cffunction>
</cfcomponent>
```

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

The World's Leading Java Resource Is Just a >Click< Away!



JDJ is the world's premier independent, vendor-neutral print resource for the ever-expanding international community of Internet technology professionals who use Java.

Only **\$69⁹⁹** ONE YEAR
12 ISSUES

Subscription Price Includes **FREE** JDJ Digital Edition!

www.SYS-CON.com/JDJ
or **1-888-303-5282**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE



ColdFusion U

For more information go to...

U.S.

Alabama
Huntsville
Huntsville, AL CFUG
www.nacflug.com

Alaska
Anchorage
Alaska Macromedia User Group
www.akmmug.org

Arizona
Phoenix
www.azcfug.org

Arizona
Tucson
www.tucsoncfug.org

California
San Francisco
Bay Area CFUG
www.bacflug.net

California
Riverside
Inland Empire CFUG
www.sccflug.org

California
EL Segundo
Los Angeles CFUG
www.sccflug.org

California
Irvine
Orange County CFUG
www.sccflug.org

California
Davis
Sacramento, CA CFUG
www.saccflug.org

California
San Jose (temporary)
Silicon Valley CFUG
www.siliconvalleycfug.com

California
San Diego
San Diego, CA CFUG
www.sdcflug.org/

California
Long Beach
Southern California CFUG
www.sccflug.org

Colorado
Denver
Denver CFUG
www.denvercfug.org/

Delaware
Kennett Square
Wilmington CFUG
www.bvcfug.org/

Delaware
Laurel
Delmarva CFUG
www.delmarva-cfug.org

Florida
Jacksonville
Jacksonville, FL CFUG
www.jaxfusion.org/

Florida
Winter Springs
Gainesville, FL CFUG
www.gisfusion.com/

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

Florida
Tallahassee
Tallahassee, FL CFUG
www.tcfug.com/

Florida
Palm Harbor
Tampa, FL CFUG
www.tbmmug.org

Georgia
Atlanta
Atlanta, GA CFUG
www.acflug.org

Illinois
East Central
East Central Illinois CFUG
www.ecicflug.org/

Indiana
Avon
Indianapolis, IN CFUG
www.hoosierfusion.com

Indiana
Mishawaka
Northern Indiana CFUG
www.ninmug.org

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

Kentucky
Louisville
Louisville, KY CFUG
www.kymug.com/

Louisiana
Lafayette
Lafayette, LA MMUG
www.cflib.org/acadiana/

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

Maryland
Rockville
Maryland CFUG
www.cfug-md.org

Massachusetts
Quincy
Boston, MA CFUG
www.bostoncfug.com

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

Minnesota
Brooklyn Park
Twin Cities CFUG
www.colderfusion.com

Missouri
Overland Park
Kansas City, MO CFUG
www.kcfusion.org

Missouri
O'Fallon
St. Louis, MO CFUG
www.stlmmug.com/

New Jersey
Princeton
Central New Jersey CFUG
<http://www.cjcfug.us/>

Nevada
Las Vegas
Las Vegas CFUG
www.snfcug.com/

New York
Albany
Albany, NY CFUG
www.anycfug.org

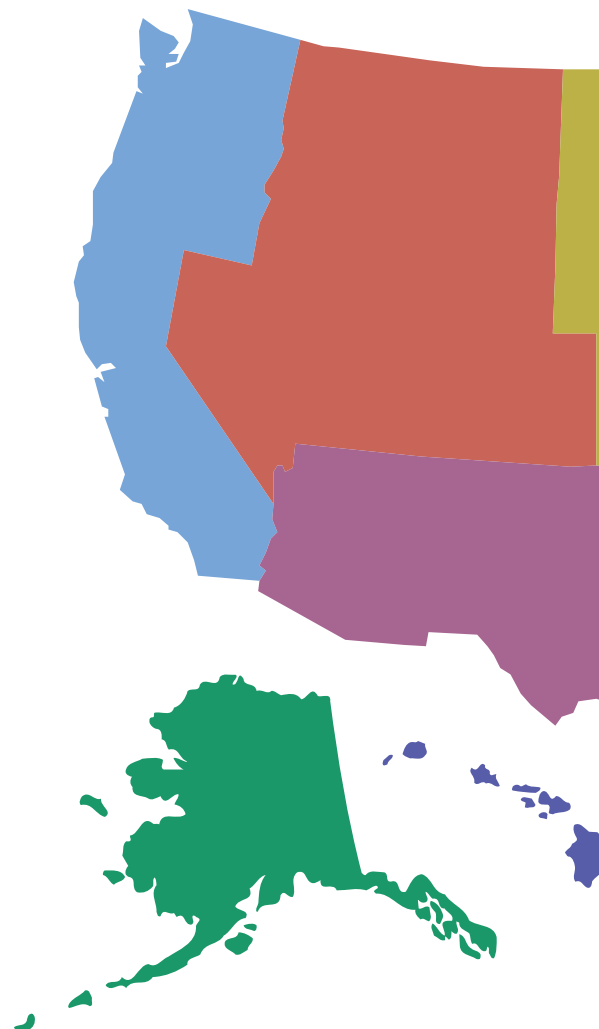
New York
Brooklyn
New York, NY CFUG
www.nycflug.org

New York
Syracuse
Syracuse, NY CFUG
www.cfugcny.org

North Carolina
Raleigh
Raleigh, NC CFUG
www.ccfug.org

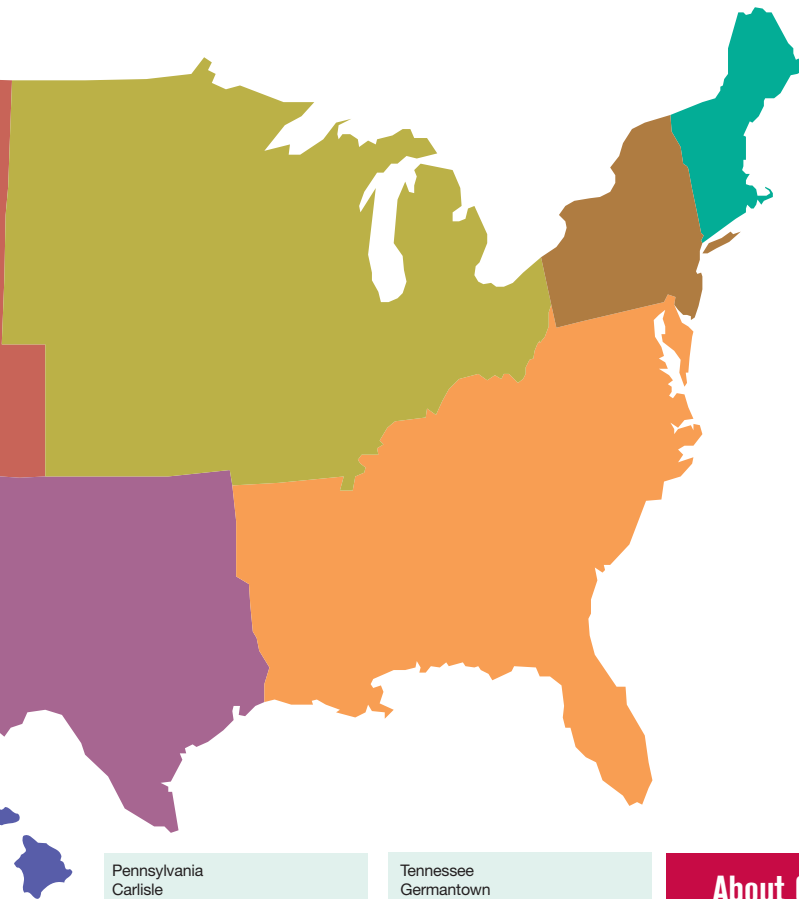
Ohio
Dayton
Greater Dayton CFUG
www.cfd Dayton.com

Oregon
Portland
Portland, OR CFUG
www.pdxcfug.org



User Groups

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



Pennsylvania
Carlisle
Central Penn CFUG
www.centralpenncfug.org

Pennsylvania
Exton
Philadelphia, PA CFUG
www.phillycfug.org/

Pennsylvania
State College
State College, PA CFUG
www.mmug-sc.org/

Rhode Island
Providence
Providence, RI CFUG
www.ricfug.com/www/meetings.cfm

Tennessee
LaVergne
Nashville, TN CFUG
www.ncfug.com

Tennessee
Germantown
Memphis, TN CFUG
www.mmug.mind-over-data.com

Texas
Austin
Austin, TX CFUG
www.cftexas.net/

Texas
Corinth
Dallas, TX CFUG
www.dfwcfug.org/

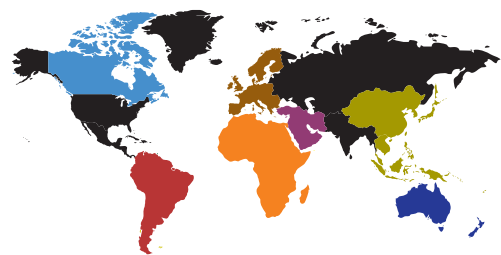
Texas
Houston
Houston Area CFUG
www.houcfug.org

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

INTERNATIONAL



Australia
ACT CFUG
www.actcfug.com

Australia
Queensland CFUG
www.qld.cfug.org.au/

Australia
Southern Australia CFUG
www.cfug.org.au/

Australia
Victoria CFUG
www.cfcentral.com.au

Australia
Western Australia CFUG
www.cfugwa.com/

Italy
Italy CFUG
www.cfmentor.com

Japan
Japan CFUG
cfusion.itfrontier.co.jp/jcfug/jcfug.cfm

Scotland
Scottish CFUG
www.scottishcfug.com

South Africa
Joe-Burg, South Africa CFUG
www.mmug.co.za

South Africa
Cape Town, South Africa CFUG
www.mmug.co.za

Brazil
Brasilia CFUG
www.cfugdf.com.br

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

Brazil
Sao Paulo CFUG
www.cfugsp.com.br

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfugtorenton.org

Ireland
Dublin, Ireland CFUG
www.mmug-dublin.com/

Spain
Spanish CFUG
www.cfugspain.org

Switzerland
Swiss CFUG
www.swisscfug.org

Thailand
Bangkok, Thailand CFUG
thaicfug.tei.or.th/

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org





Profiling at the Tag

CFML Level, Finally!

Make short work of tuning



By Charlie Arehart

The ability to view tag-level execution profiling (the amount of time spent on each tag in a request) is no longer a dream, and it opens up powerful new forms of debugging and performance tuning.

The Problem: Tools We Lack

Most CFML developers know that both ColdFusion and BlueDragon can be configured to show debugging info at the bottom of a page, which shows (among other things) the total time spent on a request and in each template within that request.

But how often have you wished you could see the time spent on each **tag** within the request? This would really help in determining where specific bottlenecks exist. While seeing that timing information for each individual tag would be powerful enough, we really need to start a higher level. What we need is a way to view all the requests against the server for a given time period, viewing their total execution time in aggregate, so we can identify just which requests are indeed the longest-running and therefore deserve our closer attention.

All these kinds of profiling/tuning tools have long been available to other developers, but not to us CFML developers. Users of BlueDragon, though, will soon have just this kind of tool in the BlueDragon CFML Profiler (scheduled to be released later this year), which has been designed to address this very kind of challenge.

Solving a Real Problem with the Profiler

Rather than just talk about the tool, let's instead dive right into using it to solve an actual problem. To help both identify the problem and confirm the result of code changes, I've used a load testing tool to run a small set of requests executing a subset of the Macromedia example applications that come with ColdFusion.

It's not important to discuss the load testing tool, nor will I bother detailing the particular requests I ran in the test. And I certainly don't mean to disparage the examples. They weren't written to be demonstrations of well-tuned code but rather very simple examples which beginning CFML developers could easily understand. Even so, they do represent rather typical practices that many developers employ which can hold up performance.

With the Profiler enabled, I ran a simple short load test to run through some of the example pages. (The details of how to start and configure the profiler may change between now and its final release, so I won't elaborate on how it's started now.)

The profiler is an extension to the BlueDragon engine, using a very low-overhead mechanism (aspects) to create a stream of profiling data available on a particular port, which can then be analyzed by a tool that retrieves the stream of profiling information.

Even though it's low-overhead and won't add much burden during profiling, the profiler can still generate a very large amount of information if enabled for too long a test (it's tracking every tag in every request while enabled). Profiling tools aren't really meant to be used to analyze full bore load tests. Again, I was just using a small sample generating multiple requests against several pages running for about 30 seconds.

The current version of the profiler is a web-based interface (HTML, generated from CFML), but a Flash-based version could be a powerful adaptation and is under consideration.

Viewing the Profile of All Requests

With the load test completed (or indeed even while it's running), I opened the web-based interface to view the requests currently being profiled. See Figure 1. The “get latest profiles” button caused the profiler to gather all the current profiling information being generated by the latest set of requests.

The profiler collects all the information since the BlueDragon engine has started, but I can clear the profiled requests at any time before running another set of tests, in order to focus only on those most recently run requests. I'll be doing this later, after performing some code changes based on the information presented from this first load test.

Since profiling may be generated for many requests (even hundreds in my small test), the interface provides for scrolling to

show just 10 requests at a time. The list of requests can also be sorted by any of the available columns, and I have chosen to sort it by the “Render Time” column, listing the requests by total execution time per request.

Further, you'll notice that many of them are highlighted in yellow. That's not something I added to the screenshots for this article but rather it's what the profiler will do for you to help isolate the most troubling requests, based on a “filter” field available on the display. I've entered a value of 100 (milliseconds), but you can set it to whatever value you prefer depending on the kind of data shown in your profile display.

We can see that all the requests shown as being of greatest duration are for the page `/cfdocs/exampleapps/personneldirectory/results.cfm`. The load test ran many other requests, but these are the ones taking the most time, so that's where I can start to focus my tuning attention.

To be fair, the kind of information shown in Figure 1, viewing total time per request, can be obtained from other tools, including simple web server log analysis tools. Indeed, yet another tool offering this sort of aggregated CFML page request information (for both CFMX and BlueDragon/J2EE) is the excellent SeeFusion tool, available at seefusion.com.

Even so, there is at least one hint of the greater power of the profiler in that the display also shows the number of tags run within the request. Clearly that's not reported by a web server log analysis tool (nor even by SeeFusion, though it has great value as an adjunct to the profiler which I'll discuss later).

Drilling Down to View All Tags In The Request

At this point, most CFML developers would at least be delighted to be able to take this information and focus on the one page that's showing to cause the most execution time. They would open the code and start eye-balling to see what's up, but the BlueDragon Profiler can do much more for you!

You can click on a given request, and as Figure 2 shows it will drill down and view details for that request, displaying several interesting kinds of detail that CFML developers have simply never seen. First, notice that it displays all the tags that were executed in the request, and further it shows the number of times they were executed, how long they took in aggregate, as well as the average time spent per execution of that tag. Very cool!

Note as well that this information is across all templates executed within the request, meaning included files and CFML custom tags, whose names are all shown at the bottom of the display. That portion of the screen also shows how many times each of those files was used in the request as well as the number of tags in each file and the total time spent in each file.

The display of tags and their execution time also supports the same filter time, so we see in this particular example that the CFQUERY tag is clearly the longest running tag in the request—indeed it is the same time shown for the “template as a whole”, so it's clearly the biggest thing the template does. (The highlighting of CFMODULE reflects an anomaly which will be resolved before the final release.)

Drilling Down to View All Executions of a Given Tag in the Request

Again, at this point one may be tempted to go edit the template and find the CFQUERY tag (it does say there's only one

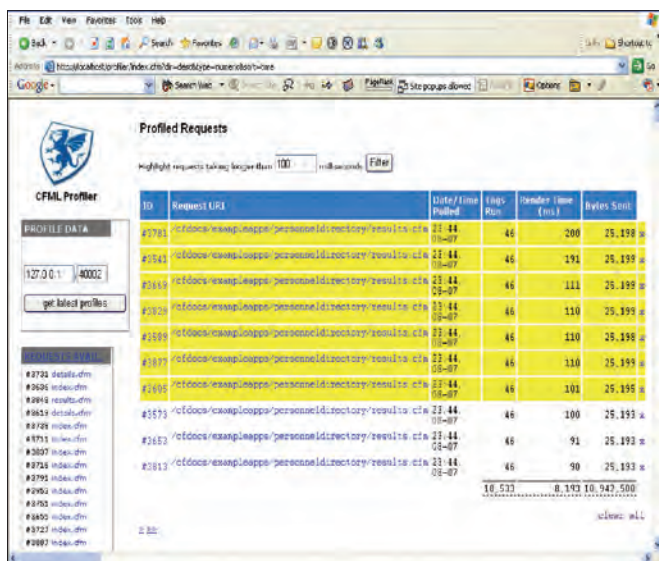


Figure 1: Display of Longest Running Requests

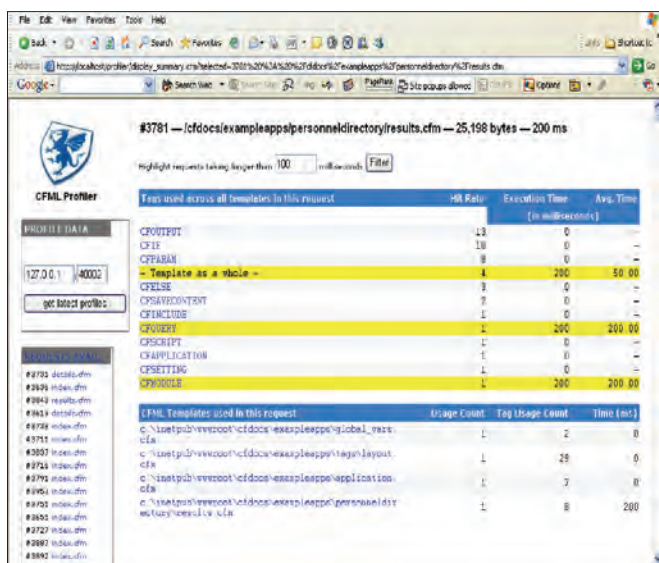


Figure 2: Display of All Tags and Their Total Execution Time In A Given Request

in the template), but we don't technically know which file it occurs in (there are four files making up this request). Still, most CFML developers would again happily rely on their traditional experience at this point to root through the files to find the tag and attack it. But they don't have to waste their time!

The profiler helps save that time and hassle in a couple more ways, providing still more powerful profiling information. While the CFQUERY tag is clearly our focus, let's take a moment instead to view what we could see about a tag that shows having been executed in several places within a template (or across many files within the request).

For instance, notice that the CFOUTPUT tag shows at the

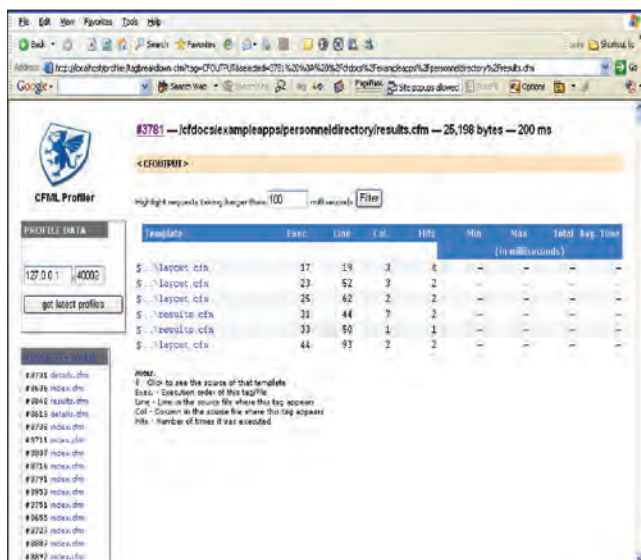


Figure 3: Details of All Uses of a Given Tag Across the Entire Request

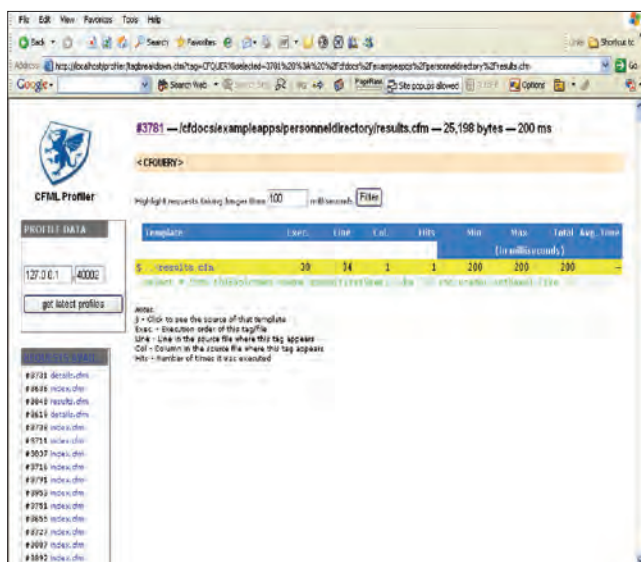


Figure 4: Display of All Uses of CFQUERY Across the Entire Request

top of the list of Figure 2. The list is sorted by default by the "hit rate" or number of times a given tag is executed within the request (across all files). Even though it's not a performance concern, let's take a look at drilling down to view the details about all uses of CFOUTPUT (since there's only one occurrence of CFQUERY, which we'll come back to in a moment).

In Figure 3, we see that the 13 executions of CFOUTPUT are spread across two templates (layout.cfm and results.cfm), and notice further that we see the specific line number and number of times executed for that tag for each line (totaling to the 13 that were listed for CFOUTPUT at the top of Figure 2), as well as the time spent (min, max, total, and average) for that tag on that line in that file, across the entire request. Again, potentially powerful stuff!

But the CFOUTPUT tag doesn't show much in terms of the time values displayed (their values weren't even enough to register). I just wanted to diverge for a moment to just show this additional level of functionality available in the profiler. It could come in handy in other circumstances.

Indeed, we could drill down one more time into any one of those files listed (clicking on the file name) to see a further detail display for just that file (for this request), showing the same columns but listing all the tags in that one file. In the interest of time, I'll skip showing that screen.

Going back to the CFQUERY tag identified in Figure 2, if we were to drill down on that page to view the details about the use of CFQUERY across all files in the given request, it would show the details in Figure 4.

Notice that there's a slight (and useful!) difference in Figure 4 compared to Figure 3. Since it is a CFQUERY being displayed, the profiler goes a step further and shows the specific SQL statement associated with that CFQUERY on that line. Nice.

Displaying the Selected Tag's Source Page Context

So at this point we know that we have a rather long-running query at line 34 of results.cfm. Again most CFML developers would be itching to open their editor at this point to get to work on the code! But you may be tickled to learn that if we were still doing analysis at this point (or didn't have ready access to the source, such as on a remote box), we don't *have* to open the template in an editor to view the code.

The profiler itself offers a simple and useful display of the given tag, in the context of its source. Notice the \$ symbol displayed to the left of the filename shown in Figure 4. That acts as a hyperlink (different from clicking on the filename) which when clicked will display the CFML for that file, highlighting specifically the line number selected. Figure 5 shows how it opens to displaying line 34, as expected.

Turning to the Code to Tweak and Tune: Attempt 1

So, that's enough of the overview of how to use and understand the profiler displays. Let's use the information we've seen to now actually tune this application and review the results with the profiler. Even without opening an editor, we see in Figure 5 that there are a couple of issues with the CFQUERY and its SQL. Can you spot the possible opportunities for improvement?

We could also turn our attention to them. As is always the case, tuning is an iterative process.

Conclusion: The Profiler Made Short Work of Tuning

Clearly, the Profiler has been a powerful tool. It's made quick work first of identifying the longest running requests for a given period of time, then we were able to drill down to view the longest-running tag(s) within a given long-running request. Further, we could then drill down to view the uses of that tag across all files in that long-running request. Finally, we could even choose to look at the CFML source to view a selected long-running tag in context of its source.

It took a while to explain it all, but clearly in normal operation this could have taken just minutes to possibly identify and resolve a key bottleneck. This was a query problem, and I mentioned SeeFusion earlier which also has a powerful feature for specifically identifying query bottlenecks. I do recommend you check it out. Even so, I could just as well have found a problem with something other than a query problem. In any case SeeFusion wouldn't have targeted the specific line of code at issue. There's certainly a place for both tools.

If you're looking forward to this powerful profiling capability, I'll just remind you again that it's a BlueDragon-only feature, scheduled to be released later this year (the interface and feature set are subject to change). We already have ideas of still more profiling information we'd like to show (memory used

ID	Request URI	Date/Time	Tags	Header Time (ms)	Bytes Sent
#1345	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	240	25,179 x
#1346	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	80	25,169 x
#1347	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	80	25,169 x
#1348	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	71	25,167 x
#1349	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	70	25,167 x
#1350	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	70	25,167 x
#1351	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	70	25,169 x
#1352	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	70	25,169 x
#1353	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	70	25,172 x

Figure 6: Display of Requests After First Code Change

ID	Request URI	Date/Time	Tags	Header Time (ms)	Bytes Sent
#1345	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	56	140	24,892 x
#1346	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	70	25,159 x
#1347	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	43	60	22,673 x
#1348	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	42	50	24,001 x
#1349	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	64	50	25,820 x
#1350	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	41	25,260 x
#1351	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	40	25,261 x
#1352	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	40	25,261 x
#1353	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	40	25,260 x
#1354	/cfdocs/examp1oapps/personalidirectory/results.cfm	22:56:08-07	46	40	25,260 x

Figure 7: Display of Requests After Second Change

for the objects in the page, and more.) Keep an eye on us.

The best place to learn about BlueDragon and its advantages is to visit the web site, <http://www.newatlanta.com/bluedragon/>, which has ample self-help information. To hear when the profiler is released, join the BlueDragon Interest list (http://www.newatlanta.com/products/bluedragon/selfhelp/archive_search/index.cfm), a mailing list staffed by our engineers and customers.

About the Author

Charlie Arehart, co-technical editor of ColdFusion Developer's Journal and a Macromedia Certified Advanced ColdFusion developer and trainer, is CTO of New Atlanta Communications, makers of BlueDragon. In this role he continues to support the CFML community, contributing to several CF resources and speaking frequently at user groups throughout the country.

charlie@newatlanta.com

Don't Miss
CFDJ's
Next
Issue!



Debugging, logging/auditing, and testing – focus on the Various ways to use cfml, the coldfusion administrator, and other Technologies to debug and test code as well as techniques for Implementing auditing and logging features in your applications



MAX is Angela Buraglia.

Angela Buraglia is a web developer and work-from-home mom who somehow finds time to co-author technical books, including her latest **Dreamweaver MX 2004 Killer Tips**. She is one of thousands of leading designers and developers that will gather at MAX 2005 this October to learn new skills, explore emerging technologies, share techniques with peers, and put exciting new ideas in motion.

Learn

Choose from over 90 different hands-on and workshop sessions – in five tracks – to create a schedule to meet your specific needs. Hear Angela Buraglia and other industry leaders speak on best practices and coming trends and technologies.

Connect

Exchange ideas with other designers and developers at networking sessions. Attend “birds-of-a-feather” sessions to connect with like-minded peers.

MAX 2005 happens October 16-19 in Anaheim, California. Please join us.

Register Now

Save \$200 and get the best session selection with early-bird registration at macromedia.com/max (ends August 26).



Ideas in motion. The 2005 Macromedia® Conference.

© 2005 Macromedia, Inc. All rights reserved. Macromedia and the Macromedia logo are trademarks or registered trademarks of Macromedia, Inc., in the United States as well as in other countries. Other marks are the properties of their respective owners.



Intermedia.NET...

Now serving the hottest ColdFusion.



At Intermedia.NET we go beyond the industry standard by supporting the hottest new Coldfusion software, offering power like never before. For nearly a decade, we've been providing reliable, secure hosting to thousands of companies across the globe. We can do the same for you.

Intermedia.NET's premier hosting services include:

- ColdFusion MX hosting
- Custom tag registration
- Competitive plans
- Verity collections search engine
- Security sandboxes
- Guaranteed service levels

Unprecedented power, unmatched reputation...
Intermedia.NET is your hosting solution.



INTERMEDIA.NET

Call us at: 1.888.379.7729

e-mail us at: sales@intermedia.NET

Visit us at: www.intermedia.NET