

COLD FUSION Developer's Journal

ColdFusionJournal.com

August 2006 Volume:8 Issue: 8

CFUnited Review 9

Faster Debugging with CFTIMER and CFTRACE 10

Multiple-File Uploads with ColdFusion 26

Errors in Your Code 18

Developing Flex 2 Applications with CF and XML Without Needing FDS or Mystic 32

Power Your Productivity 46



Presorted
Standard
US Postage
PAID
St. Croix Press





ColdFusion Hosting is our Complete Focus

➤ **POWERFUL HOSTING PLANS**

FREE SQL server access | FREE account setup | Unlimited email accounts | Generous disk space & data transfer
30 day money-back guarantee | Great value

➤ **RELIABLE NETWORK**

99.99% average uptime | State-of-the-art data center with complete redundancy in power, HVAC, fire suppression,
bandwidth and security | 24/7 network monitoring

➤ **FANTASTIC SUPPORT SERVICES**

24/7 support services | Knowledgeable phone support | We focus on your individual needs

CFDynamics

866.233.9626 ➤ CFDYNAMICS.COM

For years we have been involved in the Cold Fusion community and have come to know what developers and project managers look for in a web host. The combination of our powerful hosting plans, reliable network, and fantastic support sets us apart from other hosts.

Real service. Real satisfaction. Real value. Real support. Real Freedom.



<Guru:Logic/>

A shortcut doesn't take away the fun.

It is the fun.

Get the free Adobe® Flex™ trial beta at adobe.com/flex.

The fastest way to create rich Internet applications.

Real time. Cross-platform.

Expressive. Standards-based programming model. Data intensive app.

Extend Ajax. And go beyond.

<Download Adobe Flex now/>





**For the greatest hits
of the 70's, 80's and 90's
call your web host's
tech support.**

For answers call us at 1-866-EDGEWEB
3 3 4 3 9 3 2

When calling your web host for support you want answers, not an annoying song stuck in your head from spending all day on hold. At Edgewebhosting.net, we'll answer your call in two rings or less. There's no annoying on-hold music, no recorded messages or confusing menu merry-go-rounds. And when you call, one of our qualified experts will have the answers you're looking for. Not that you'll need to call us often since our self-healing servers virtually eliminate the potential for problems and automatically resolve most CF, ASP, .NET, SQL, IIS and Linux problems in 60 seconds or less with no human interaction.

Sleep soundly, take a vacation, and be confident knowing your server will be housed in one of the most redundant self-owned datacenters in the world alongside some of the largest sites on the Internet today and kept online and operational by one of the most advanced teams of skilled Gurus, DBAs, Network and Systems Engineers.



For a new kind of easy listening,
talk to EdgeWebHosting.net

<http://edgewebhosting.net>

By the Numbers:

- 2 Rings or less, live support
- 100% Guarantee
- 99.999% Uptime
- 2.6 GBPS Redundant Internet Fiber Connectivity
- 1st Tier Carrier Neutral Facility
- 24 x 7 Emergency support
- 24 Hour Backup
- Type IV Redundant Datacenter



2003 - 2006

◦ Shared Hosting ◦ Managed Dedicated Servers ◦ Managed Colocation ◦ Semi-Private Servers
◦ ColdFusion ◦ BlueDragon ◦ ASP ◦ .NET ◦ .Linux ◦ .Java ◦ SQL Server ◦ .MySQL ◦ Self-Healing Servers

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

executive editor

Nancy Valentine nancy@sys-con.com

research editor

Bahadır Karuv, PhD bahadir@sys-con.com

production**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

editorial offices**SYS-CON MEDIA**

577 Chestnut Ridge Rd., Woodcliff Lake, NJ 07677

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

577 Chestnut Ridge Rd., Montvale, NJ 07677

@copyright

Copyright © 2006 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 Megan Mussa, megan@sys-con.com.

SYS-CON Publications, Inc., reserves the right to

revise, republish and authorize its readers to use

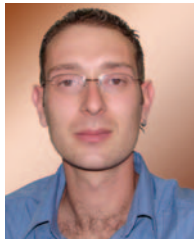
the articles submitted for publication.

All brand and product names used on these pages

are trade names, service marks, or trademarks

of their respective companies.

CFDJ – The Ultimate Resource for ColdFusion Developers



By Simon Horwith

When Cold-Fusion

Developer's Journal was

launched eight years ago,

it was the first printed

periodical exclusively by and for ColdFusion de-

velopers. Over the years there have been many

changes in format, authors, and publishing/edi-

torial staff. One thing has remained constant,

however: the focus of the magazine has always

been to serve the CF development community

with the best content possible.

Over the course of this summer, we have formed a CFDJ Editorial Board that consists of community leaders and very experienced developers. Many of the board members are seasoned CFDJ authors, some are relatively new to the magazine itself, but all bring years of experience and community leadership, as well as vision and a strong desire to help contribute to making CFDJ the best that it can be.

Care was taken to ensure that we also have experts who have specialized in specific topic areas, and that the collective specialties of all board members create a very well-rounded panel capable of offering insight into every aspect of ColdFusion development. In addition to contributing articles, editorial board members are responsible for soliciting and tech-editing articles for those issues of CFDJ

that they do not choose to contribute articles of their own.

This well-rounded coverage of specialties will allow CFDJ to bring content covering a wide array of topics like never before. CFDJ will continue to cover "traditional" topics dealing with writing ColdFusion code, but will now cover much more than just that. On a regular basis CFDJ will include articles that focus on topics including but not limited to:

- Next generation RIA software and other industry trends
- Developer careers and the ColdFusion job market in general
- Frameworks and coding/architectural styles
- CF Server performance tuning, testing, and troubleshooting
- Server and application security
- Regular puzzles and solutions
- ColdFusion community related topics
- Case studies
- Open source ColdFusion
- Object-oriented CF

About the Author

Simon Horwith is the editor-in-chief of Cold-Fusion 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

CF_UNDERGROUND VIII PRE-MAX EVENT



DATE: SUNDAY, OCTOBER 22, 2006

LOCATION: THE BEACH, LAS VEGAS

*Speakers include:
Simon Horwith
Michael Smith
Glenda Vigoreaux
and more...*

365 Convention Center Drive, Las Vegas, Nevada

Meet with old friends and new before the MAX event. A full day of networking, ColdFusion, great talks, drinks, food, and fun.

WWW.CF-UNDERGROUND.COM



president & ceo

Fuat Kircaali fuat@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

associate sales manager

Kerry Mealia kerry@sys-con.com

Lauren Orsi lauren@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 coordinator

Edna Earle Russell edna@sys-con.com

sys-con.com

vp, information systems

Robert Diamond robert@sys-con.com

web designers

Stephen Kilmurray stephen@sys-con.com

Wayne Uffelman wayne@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 gailn@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



- Getting started with CF
- IDEs and other utilities that aid CF developers
- And many, many more (write me at simon@horwith.com if you have specific ideas of your own that I haven't mentioned)

These articles and topics will be written and chosen in such a way so that we can offer invaluable material to developers of all skill levels and varied topics of interest.

I am remaining as editor-in-chief to assist our editorial board any way that I can, helping them to solicit articles, and will have a focus

Editorial Board as follows:

- Simon Horwith (simon @ horwith.com)
- Charlie Arehart (charlie @ carehart.org)
- Ben Forta (ben @ forta dot com)
- Shlomy Gantz (shlomy @ bluebrick.com)
- Hal Helms (hal @ halhelms.com)
- Jeffry Houser (jeff @ farcryfly.com)
- Doug Hughes (dhughes @ alagad.com)
- Ed Sullivan (esulliva @ adobe.com)
- Jeff Peters (jeff @ grokfusebox.com)
- Brian Rinaldi (brinaldi @ remotesynthesis.com)
- Robi Sen (robisen @ gmail.com)
- Matt Woodward (matt @ mattwoodward.com)



of my own - making sure that the overall content remains well rounded. Some readers expressed concern about the amount of attention and focus that CFDJ has given Flex 2 lately, for example, and I am happy to lay those concerns to rest. We'll have one article every month related to creating next-generation Web applications using Flex 2, AJAX, SPRY, or what-have-you for those interested readers, but the overall focus of each issue will remain completely targeted at discussing topics specific to a larger majority of ColdFusion developers.

It is with great excitement that I announce the 2006-2007 ColdFusion Developer's Journal

- Brandon Harper (brandon @ gmail.com)

You can read the editorial board member bios online at <http://coldfusion.sys-con.com/general/editboard.htm>.


There is still one key ingredient necessary to making CFDJ the best resource available: feedback and editorial contributions from you, our readers. I strongly encourage all of our readers to contact me and/or any of the CFDJ editorial board members listed above, with any questions or concerns, as well as with any interest you may have in contributing content to CFDJ, the ultimate resource available to ColdFusion developers. 

TABLE OF CONTENTS



editorial

CFDJ - The Ultimate Resource for
ColdFusion Developers

By Simon Horwith

5

show report

CFUnited

Reviewed by Ryan Hartwich

9

debugging

Faster Debugging with CFTIMER
and CFTRACE

Hidden gems

By Shlomy Gantz

10

tips

ColdFusion to the Rescue

A stopgap solution
saves the day

By Adedeji Olowe

12

blogs

Handling 404 Errors for
a Migrated Blog

The technology behind
the solution

By Joshua Curtiss

16

tools

Multiple-File Uploads
with ColdFusion

By Dave Shuck

26

cfugs

ColdFusion User Groups

Power Your Productivity

By Tom Schreck, Byron Bignell &
Nolan Dubeau

46

CF United

The chance to learn new techniques

Reviewed by Ryan Hartwich

I last attended CFUnited in 2003 when it was called CFUN-03 and it was a far smaller, less professional conference. At the time, there were around 300-350 attendees and the conference was just passing the point in which it felt like a small, developer-organized affair. This year, it has grown to approximately 900 people and feels very much like a corporate conference.

TeraTech, a ColdFusion consulting and training firm based in the Washington, DC, suburbs, is the organizer of the conference and a long-time sponsor of the Maryland ColdFusion User Group. TeraTech dedicates two full-time staff members to organizing the conference and, based on what I see, they are obviously working hard to put on a good show.

As a whole, the conference was well organized. With over 60 speakers over four days, this is the leading ColdFusion conference worldwide. Adobe is a large sponsor and it is quite clear from their commitment that they want to support the market. Adobe sent at least 10 ColdFusion developers to the conference as speakers, community support (user groups), and to man their "help desk" booth in the community area. One significant change from previous years was the addition of a fourth day of presentations, all repeats of the most popular sessions from the first three days. This made it far easier for attendees to schedule the key sessions we wanted to attend. In previous years, you could miss a key presentation if it was scheduled at the same time as another you wanted to attend. (Note that a few months before each conference, registrants can indicate session preferences and later sign up for them. This allows the organizers to modify the schedule to minimize overlap of popular sessions and to guarantee that the more popular sessions are in the largest rooms.)


The conference was held in a very nice, large conference center that is obviously new and well maintained. It was large enough to cope with the crowd most of the time; the only space problems were during casual lunches (buffet-style held in the common lobby areas with sparse seating at peak times) and in a few presentations that were heavily attended (there is additional conference space available on another floor). Each technical session was professionally recorded by a third-party firm and the audio is being made available for free to attendees. The video is being made available to non attendees at a fairly high price (\$650 U.S.) with a discount price of \$200 for attendees. To be fair

to Michael, this is the first time he has had the sessions video recorded and to do so over four days for all concurrent sessions costs tens of thousands of dollars. This was an experiment and some experimentation with the supply and demand curve is to be expected. Michael pointed out to me that the primary goal of the recording was not to replace the need for attendees to attend or for their casual viewing, but was for employers in distant states (and countries) who could not send their staff to the conference and wanted a way for their whole team to benefit from the speakers at their convenience (hence, the relatively high individual cost). I'm looking forward to previewing the recordings in the next week or two when they are available.

To the average attendee, the problems were few and far between. There were some minor snafus with the complimentary shuttle bus to the overflow hotel as well as some minor delays in registration, Internet access, etc. As a whole, none of these were significant and shouldn't scare you away.

How can the conference grow and improve in future years? TeraTech needs to increase their staffing slightly (which I am told they have done) to cope with the increased attendance, organizational responsibilities, and minor areas that were lacking. I think they need to bring in a wider assortment of vendors (approximately 20 were represented) to display their ColdFusion products and also to increase the community networking features of the conference. Networking is a big benefit of the conference and the opportunity to network was constantly available, though not actively assisted.

The most obvious area that was lacking was electrical and Internet connectivity for my laptop. Conference centers are not normally equipped to handle hundreds of developers with laptops. I recommend that the presentation halls be equipped with long power strips on the sides of the presentation rooms farthest from the entry doors (the entry door areas typically have a few plugs and the largest density of attendees anyway). The Internet access should have been adequate, but a combination of Flex 2 and ColdFusion releases on the first day (with a large number of attendees downloading hundreds of megabytes of files) as well as configuration problems with the wireless made connecting to the Internet frustrating at times.

Overall, I would say the conference was a success. It was fun to attend and educational. I was exposed to new technologies and frameworks that will benefit me in my job while giving me a chance to learn new techniques to improve my coding. 

About the Author

Ryan Hartwich is a mechanical engineer with 7 years of ColdFusion experience.

Faster Debugging with CFTIMER and CFTRACE

Hidden gems

By Shlomy Gantz

Two of the most common debugging tasks are displaying variables and identifying bottlenecks. For years I relied on CFDUMP and CFABORT to display variables and getTickCount() to calculate processing time and identify slow-running code.

Indeed these wonderful tags and functions have long been the quintessential toolkit of any ColdFusion developer.

```
<CFSET Email = "shlomy@bluebrick.com">
<CFDUMP var="#Email#" label="Email Address">
<CFABORT>
```

In the approach above I used <CFDUMP> and <CFABORT> tags to display the value of a variable. Want to display a few variables? The screen quickly becomes a rainbow of purple, green, and gray as you look for your variables. While the wonders of CFDUMP probably deserve their own article, Adobe recently introduced a powerful new tag called CFTRACE.

Introducing CFTrace

With CFTRACE, you can easily display changing variable values either inline or in the debugging section, categorize variables, and even abort execution, all in one tag. So the good ole CFDUMP becomes:

```
<CFSET Email = "shlomy@bluebrick.com">
<CFTRACE var="Email" text="Email" inline="yes" abort="yes">
```



*Note that with CFTRACE the var attribute accepts a variable name without pound signs.

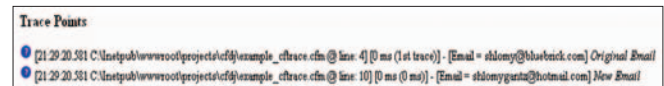
The Advantages of CFTrace

"So what" you say, "I've been using CFDUMP and CFABORT for years, why would I switch?"

Well, there are many reasons. Have you ever had to display a variable more than once? The task of identifying those values on the screen can become very cumbersome, especially if manipulated and changed multiple times.

Showing variable values at different points in a template is easy with CFTRACE. Simply add CFTRACE tags after every change and you'll see an organized list in the debugging info section at the bottom of the screen.

```
<CFSET Email = "shlomy@bluebrick.com">
<CFTRACE var="Email" text="Original Email">
.
.
.
<CFSET Email = "shlomygant@hotmail.com">
.
<CFTRACE var="Email" text="New Email">
```



Another clear advantage of CFTRACE is that it can easily be turned on and off via the ColdFusion administrator. I find it extremely useful not to constantly remove and add code during the development process just to display variables.

Simply browse the ColdFusion Administrator and go to Debugging & Logging -> Debugging Settings to turn CFTRACE on and off.

☒ Tracing Information

Select this check box to show trace event information in the debugging output. Tracing lets a developer track program flow and efficiency through the use of the CFTRACE tag.

Not convinced yet? Consider the fact that CFTRACE also lets you categorize and type information as well as display the time differences between traces. CFTRACE calls are also displayed in Dreamweaver's server debug tab results window and ColdFusion logs <cftrace> calls to the {CFUSIONMX}\logs\cftrace.log.

All of the above should be enough to convince any developer to give CFTRACE a test drive. It's a hidden gem that has long been overlooked.

—CONTINUED ON PAGE 50

BALANCE

Designer/developer, front-end/back-end, clients/sanity. . .web development is a balance and we can help you maintain it. Join now and experience a wealth of training resources tailored to the tools you use every day.

www.communitymx.com



Visit www.communitymx.com/trial/ for your free 10 day trial.



ColdFusion to the Rescue

A stopgap solution saves the day

By Adedeji Olowe

In the 21st century business environment, companies live and die by their fat and bogus enterprise applications. New mega-industry groups have been created not only to develop these applications, but deploy, support, and train.

Nigeria is not left out of these influences. The banking industry in Nigeria is a typical example. Every bank in the country runs massive core banking applications such as Finacle from Infosys, FlexCube from Iflex (an Oracle company), Bankmaster, Phoenix, Equations, Globus, etc. These apps are expensive, clunky, and terribly difficult to use. Of all these, the cost of support, training, and hardware requirements are most annoying. Deploying any of these applications would set any organization back no less than a million dollars and it could take between eight months to four years to install. Additional user licenses could run as high as \$2,000.

The cost would have been easier to account for if the ancillary cost didn't build up so fast. In terms of usability, organizations spend a lot training users how to use these apps. They usually build their processes around their idiosyncrasies instead of the other way round.

Access Bank is a typical example of a Nigerian enterprise that has deployed some of these applications. Access Bank was incorporated in 1989 as a commercial bank. It was a small fringe player until March 2002 when it was taken over by a new set of owners led by Aigboje Aig-Imoukhuede and Herbert Wigwe. At that time, the bank was running FlexCube 4.3. Either because the bank, before the new management came in, didn't implement it well, or for some other reason, FlexCube was anything but flexible. It was slow, buggy, and not user-friendly. However, in delivering its vision of transforming Access Bank into a world-class financial services provider, the new management knew it had to upgrade the banking application to the latest version. And like all enterprise applications, it would take time, cost a lot, and everyone had to be well trained.

Since customers couldn't wait for the FlexCube upgrade before getting superior customer service (why else would a customer stay with a bank when he could get better service from a com-

petitor down the road?), the IT department was saddled with the responsibility of finding temporary solutions while the upgrade project was handed over to a project implementation team.

Like other banking applications, FlexCube 4.3 was a three-tier application with an Oracle 8i database, an application server running the business logic, and client interfaces. Since bottlenecks were at the client and application server, it was apparent that any solution must bypass the trouble spots to deliver speedy response to end users.

Specifically, the proposed temporary solution should be able to deliver all customer, account, and transaction information seamlessly to account officers and back-office staff without needing to install client-side applications. The solution must be web-based. Moreover, it must be cheap and cost-effective since it would just be ad hoc. Ad hoc because at that time it was thought that the solution would be discontinued once the FlexCube upgrade was in.

Immediately, the IT group swung into action. A slew of technologies were considered. ASP was dropped because it was considered legacy. ASP.NET was almost chosen because it came from Microsoft, was mature, and would plug well into the existing infrastructure. However, the cost of add-ons to provide functionalities such as charting, PDF generation, etc. was way beyond budget.

ColdFusion was considered. Everyone was initially put off by the price, especially when competing technologies were basically free. However, by the time the capabilities that ColdFusion



could provide straight out of the box were considered dollar for dollar against ASP, JSP, ASP.NET, etc., it was obvious it was the clear winner.

ColdFusion was discovered to have the shallowest learning curve among these products. Its rapid application development paradigm was also way beyond the others.

The bank made a decision. A proprietary framework similar to Mach-II was developed. The first batch of modules was released to the staff in less than two weeks flat. This was achievable because there was no need to integrate expensive third-party middleware for PDF generation, charting, and LDAP. The most amazing thing was that only a short mail was sent out describing what the application could do. There was no need for expensive training because the interfaces and workflows were very user-friendly and extremely intuitive.

The benefits Access Bank derived from this adventure were manifold. The most important being that it was able to live up to its service delivery standard to the customers. The reality is that customers don't know and don't care what core application the bank uses. What matters is the quality of service they get.

Another benefit was that for the level of service delivered, the ColdFusion application was cheaper, more efficient, faster, and easier to use.

Overall, the application's return on investment (ROI) was the best the bank has ever seen. The total cost of deployment was less than 10 user licenses to the core banking application. Because all non-transactional activities were migrated to it, user

licenses were only bought for the back-office staff, reducing the total cost of using the core banking application itself.

After this, the bank systematically developed other applications around the first one. The initial application, called Infopool, has now become the platform on which the bank is running almost all its processes. These include the bank's Web site, employee/self-service portal, the stock broking platform, etc.

Unlike other large organizations, especially other banks in Nigeria, Access Bank has discovered that implementing smart applications can complement core banking applications and reduce the total cost of business. All these would not have been possible without Adobe ColdFusion.

Other Information

Access Bank runs ColdFusion MX 7 Enterprise on three ML350 servers on Windows 2000 Enterprise Servers. 

About the Author

Adedeji Olowe, a Web application developer at First City Monument Bank Plc in Lagos Nigeria, has been using ColdFusion for several years. He has experience developing and extending enterprise applications for companies in the financial industry as well as in Active Directory integration.

adedeji@olowe.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.



fast
easy
affordable

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

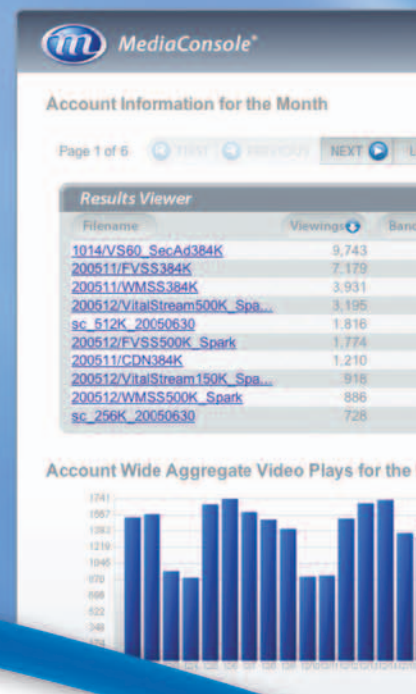
© Copyright 2005 PaperThin, Inc. All rights reserved.

Innovative Tools

To Manage, Deliver and Track Streaming Media On the Internet

MediaConsole

Real-time



Using the Right Tools For the Job Is Critical to The Success of Your Business

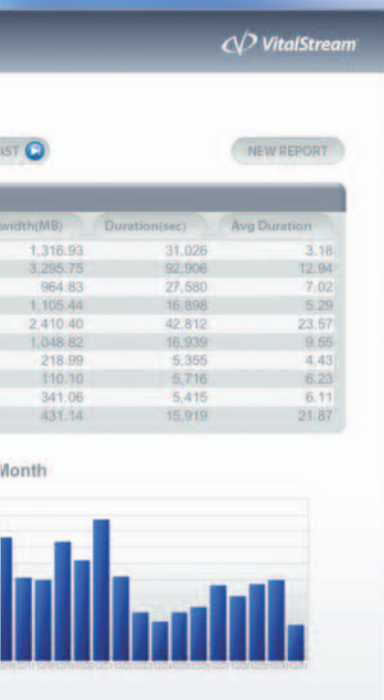
VitalStream reliably delivers streaming media to large global audiences using an award-winning content delivery network that powers all of our services. These services come with innovative tools that support and enable a variety of business models for online media distribution including advertising, subscriptions, pay-per-view and more.

Introducing the New VitalStream Media Player for Flash

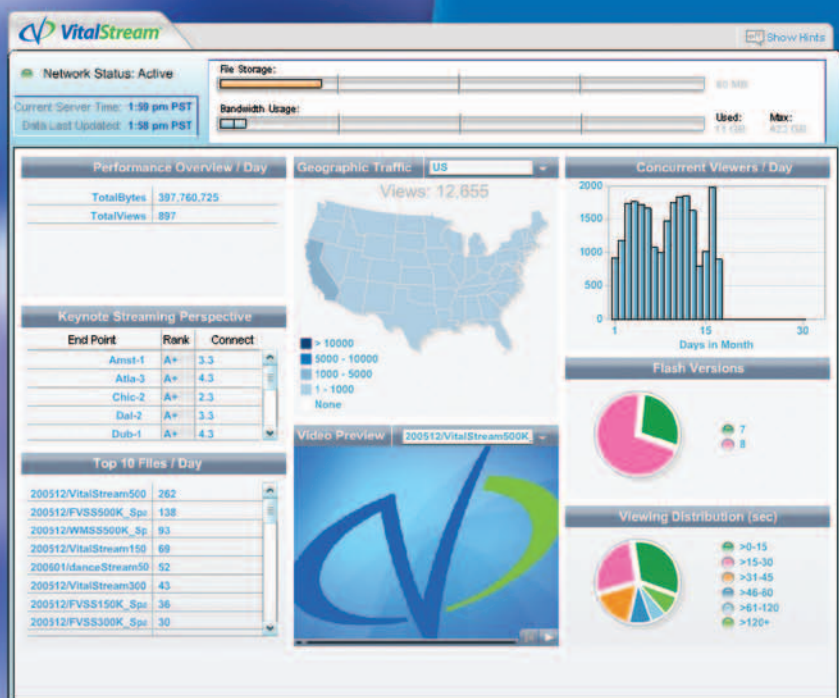
Continuing its commitment to innovation, VitalStream is proud to announce the new Media Player for Flash to help speed up your workflow by allowing you to test and play streaming Flash video locally on your computer before you upload the files to our servers.

Download this player for free at <http://www.vitalstream.com/tools/mxdj.asp>

Reporting



Reporting Dashboard



To learn more about VitalStream, call (800) 254-7554 or visit www.vitalstream.com



The World Leader In Audio and Video Streaming

Handling 404 Errors for a Migrated Blog

The technology behind the solution

By Joshua Curtiss

I just transitioned my blog in two huge ways: (a) I reassigned it to a different domain name, and (b) I changed the blogging engine I was using, which incidentally used a file organization structure that's incompatible with my new engine.

Furthermore, I decided to import all of my old posts into my new blog rather than leave the old and start anew.

I don't regret any of the changes I've made, because I've laid groundwork that I'll be much happier with in the future. But these changes can wreak havoc on my site's perceived status, since any links to my pages and any stale search engine referrals will now result in ugly 404 errors. Sure, we could set up a friendly 404 page that notifies the user of what's happened. However, why don't we eliminate the embarrassment of notifying the user at all and simply redirect them as best as possible?

An Example Scenario

In my case, I was moving my blog from the root of my site (<http://www.nazin.com>) to a sub-domain (<http://blog.nazin.com>). Furthermore, I was migrating from Blogger.com, which publishes an HTML page for each post, to WordPress, which loads everything from `index.php` and uses the post slug in the query string to determine what post to display. For instance, one highly trafficked page was <http://www.nazin.com/2004/08/airtunes-concerns-answered.htm>, and my new site housed this post under <http://blog.nazin.com/index.php/airtunes-concerns-answered/>. Once all the old HTML files were removed from the old site, those pages would come up as 404 Page Not Found errors.

The Technology Behind the Solution

To redirect the requests, assign a custom 404 page via the system administration for your site. This will naturally vary from one situation to the next depending on your server situation (Do you have direct server access? Do you have shared hosting?). Most Web servers will communicate the error and the address in the CGI query string. We'll use the original address the user

attempted to navigate to as a means to redirect them to the post on the newly migrated blog. As always, I'll do this with CF.

Building a ColdFusion 404 Handler

Through the CGI structure, we can take a look at the input we're getting from the Web server for the error and address. Let's create a preliminary 404 handler page. Let's name it `404.cfm`.

```
<cfoutput>#CGI.QUERY_STRING#</cfoutput>
```

If we were to implement this and attempt to browse to the scenario page above, we'd get output similar to this:

```
404;http://www.nazin.com:80/2004/08/airtunes-concerns-answered.htm
```

To the human eye, it's immediately apparent how to take this output and create a redirect. At this point, it's simply a matter of text manipulation. Perhaps we might write some code like:

```
<cfscript>
searchTerm=ListLast(CGI.QUERY_STRING,"/") ;
searchTerm=ListDeleteAt(searchTerm,ListLen(searchTerm,"."),".") ;
newURL="http://blog.nazin.com/index.php/" & searchTerm & "/" ;
</cfscript>
<cflocation addtoken="no" url="#newURL#">
```

Here, we're effectively telling ColdFusion:

- (1) Grab just the filename of the address (producing "airtunes-concerns-answered.htm").
- (2) Delete the filename's extension (producing "airtunes-concerns-answered"). There are UDFs that have been written to perform this function as well, such as `ripExt()` at cflib.org.
- (3) Build the new URL and
- (4) Redirect the browser to that location.

Try browsing to the deleted page, and the redirection works like a charm. However, there's a serious issue lingering in our solution: What if your post slugs don't always match from the old blog to the new blog, if you imported your old posts like I did?

Addressing Mismatched Slugs: A Different Approach

For my scenario page, the previous example worked per-

fectly. However, during the import process, WordPress didn't always create post slugs with exactly the same names as the HTML files generated by Blogger.com. Is it still possible to redirect the user?

Sure. Instead of attempting to redirect the user to the new post's address, try redirecting them to the more forgiving search page.

For the next example, let's use another high-traffic page that breaks the previous solution: <http://www.nazin.com/2005/06/how-to-install-itunes-on-windows-xp.htm>. On the new blog, this page is <http://blog.nazin.com/index.php/how-to-install-itunes-on-windows-xp-sp2/>; as you can see, the slug is slightly different because Blogger.com truncated it. But this isn't the only thing that could cause different slugs. Perhaps certain characters are treated differently in the different engines, or perhaps a customized slug was created. In any case, relying on the new blog's search engine will address the problem.

The code is actually simple:


```
<cfscript>
searchTerm=ListLast(CGI.QUERY_STRING,"/") ;
searchTerm=ListDeleteAt(searchTerm,ListLen(searchTerm,"."),".") ;
searchTerm=Replace(searchTerm,"-","+", "ALL") ;
newURL="http://blog.nazin.com/?s=" & searchTerm ;
</cfscript>
<cflocation addtoken="no" url="#newURL#">
```

The only difference with this approach is that it converts all the dashes (-) in the filename to pluses (+), which are used in the URL string to indicate spaces. Thus, the URL generated by this code (<http://blog.nazin.com/?s=how+to+install+itunes+on+windows+xp>) would be the equivalent of someone searching for "how to install itunes on windows xp" in the search box.

The user will thus be redirected to the new blog, with a link directly to the page desired.

Further Enhancements

The final step you could take to really polish the process would be to modify the search results page on your blog to redirect the user to the search results page in the event that only a single record is returned. The code to do this would vary from one engine to the next, but it would achieve the same effect as the previous solution, even when your post slugs differ.

Migrating your blog to a new domain name or to a new blogging engine can be disruptive, especially when you have other sites or search engines linking to your old, out-dated URLs. With very little time and effort, however, a simple 404 handler that utilizes the techniques we just considered can take the pain out of the migration. 

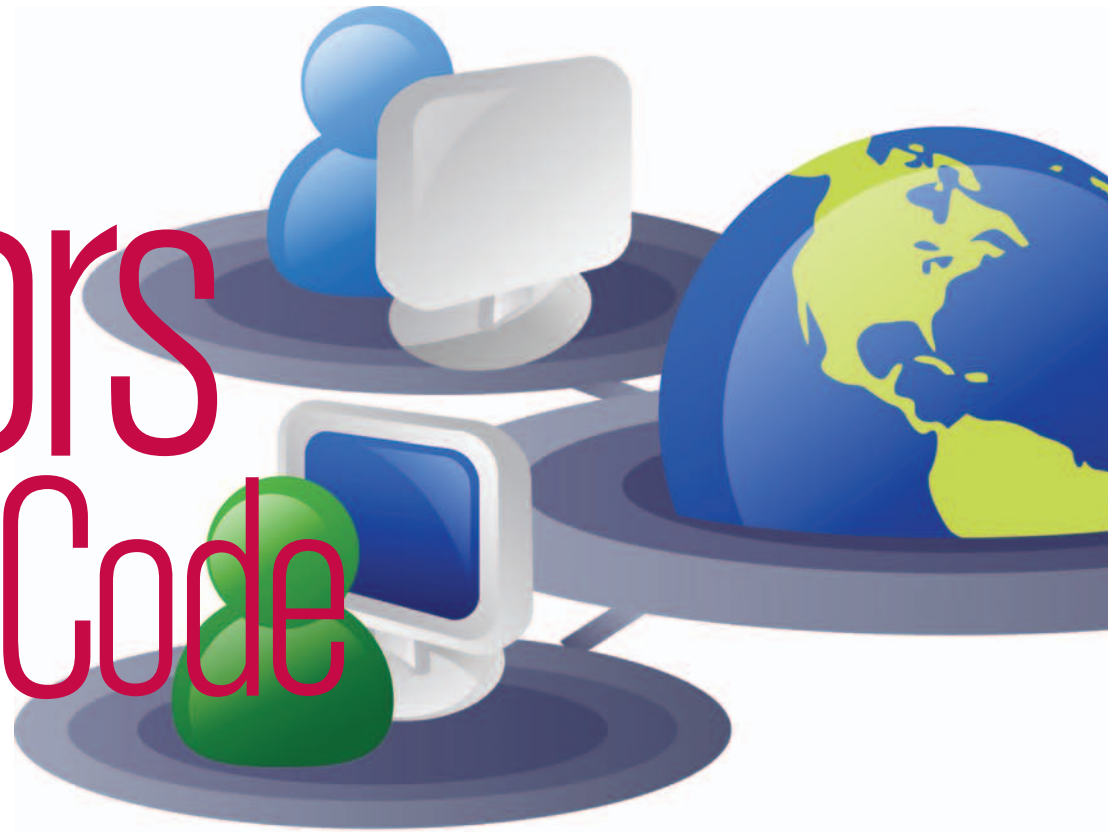
<http://blog.nazin.com>

Other companies in this magazine spent a lot of time on pretty ads. As you can see, we did not. We spent our time hiring the best people and training them to deliver outstanding support for your website. We spent our time building a state of the art datacenter and staffing it with people who care about your website like it's their own. Compassion, respect, credibility, ownership, reliability, "never say no," and exceed expectations are words that describe our service philosophy. From the first time you interact with us, you'll see what a difference it really makes. And you'll also forgive us for not having a pretty ad.



WEB HOSTING • MANAGED DEDICATED SERVERS • COLOCATION • VPS • ECOMMERCE • BLOGGING • EMAIL

Errors in Your Code



Handling, debugging, and testing for them



By Charlie Arehart

Errors and bugs: they happen in all code, mostly in development but in production too

and perhaps more easily in CFML than in compiled languages. There are several features to help better handle, debug, and

test for them, and this article will focus on those.

I started writing in the CFDJ in 1999, and I often point people back to my previous articles and those of others when challenges arise. The most common one I point to is my 4-part series on error handling from five years ago, and still other authors have addressed related but different aspects. It seems the issues of dealing with, resolving, and testing for errors remain important.

In this article I'll revisit the coding techniques and configuration settings available to improve error handling, as well as approaches to help debug and resolve errors when they occur. Some may surprise you. I'll also discuss testing techniques to help alleviate the likelihood of errors occurring in the first place.

I'll end with a few other error-related topics that might often miss attention, and along the way I'll point you to those past articles and other resources to get you started. (Unless otherwise indicated, all the features discussed here apply to BlueDragon as well as ColdFusion 5 and MX.)

Gracefully Handling Errors

We're all going to have errors as we're coding. It's one thing to deal with an error when it arises during development. You see the error and go back into the code to fix it then re-run it. Actually, you may need some help in determining the cause of the error, and I'll talk about that in the section on debugging. But it's quite a separate (and often ignored) problem of dealing with errors that occur when your code is running in production. Does the user get the traditional — admittedly Spartan — default error message that ColdFusion generates? And are you careful about what details they are shown (as can be controlled in the Admin console)? Or better still do you “handle” the error more gracefully? If you don't know how to do these, this section will help get you started.

As the CFML language has evolved, there have been various features offered to help you handle errors. This variety of alternatives can sometimes confuse newcomers, and even experienced developers. It's important to remember that CFML is now over 10-years-old, and some tags or features have been



superseded by other more powerful approaches. Each of the following topics is covered in more detail in the articles and documentation references that I'll offer throughout this article.

Error Handling Approaches

THE TAGS AND FEATURES

CFERROR type="request"

CFERROR type="exception"

Sitewide error handlers

Controlling exception information

CFTRY/CFCATCH

Try/catch

CFRETHROW

onError() in Application.cfc

The first tag introduced to help with error handling was the **CFERROR** tag, which names a template to be shown to the user when an error occurs. Typically defined in the application.cfm file (and so enabled for every page requested in that app), this error template would override the user seeing that basic CF error message.

When it first came out, it had only a *type="request"* option, and the template pointed to by this tag could not have any CFML, which made it quite limited. You could only use some HTML code to format an attractive error. There were tech-

niques that evolved to get around that, and you may still see code that uses this tag and the *type="request"* option, but you really want to consider changing to the newer *type="exception."*

I say "newer" but it was introduced in CF4.5 (in 1999!) Sadly, many still seem to miss this feature, which brought radically new functionality. At last you could do any CFML at all in the error page, and naturally many started by adding CFINCLUDEs to reuse code to match the look-and-feel of their app or, more important, some added CFMAIL tags to finally inform the developer that an error occurred. If you still have no notification that an error is occurring this is your first stop. In the section below on "Reporting the Occurrence of Errors," I'll show even more useful approaches.

While you can provide the CFERROR in every application if you're in a situation where you have many apps and they would all have the same error handler, you can take advantage of another long-standing feature of ColdFusion. In the Admin console, you can set a "**Site Wide Error Handler**" (also introduced in CF 4.5), which is just a place to name a template as discussed above which can do whatever form of error handling you'd like.

Speaking of the Admin console, I mentioned previously that if you don't handle an error at all, the end user sees the default error page that ColdFusion generates. Note that there is an Admin console setting for **controlling exception information**, which can limit the level of detail shared. This is important from a security standpoint and is often neglected. Even if you do handle your errors, there are times when that may not work as expected and the user will get the default error message. See the resources for more information.

Older still than the two "newer" features above is the **CFTRY/CFCATCH** tag pair introduced in CF 4. Where CFERROR is intended to catch any error that might occur, these tags are intended to be wrapped around some set of code that you think might possibly create an error such as a database action that might fail for lack of a database connection. Again, we don't have time to go into the particulars of using this approach, but it's quite easy and discussed in detail in the references below.

Related to CFTRY/CFCATCH, there may be times when — once you've caught an error and done something with it — you may want to pass the error handling control back to whatever other error handler (a CFTRY/CFCATCH in a calling template or the CFERROR handler or Sitewide Error handler) that might do more processing. In this case, after handling the error in CFCATCH you can then use **CFRETHROW** to pass the error "up the chain."

These topics have been covered in previous CFDJ articles that are available online. First, I'll point out that I did a four-part series covering the topics above:

- "Toward Better Error Handling – Part 1 (Admin Settings)" (October 2000) <http://coldfusion.sys-con.com/read/41977.htm>
- "Toward Better Error Handling – Part 2 (Error Handling Templates)" (November 2000) <http://coldfusion.sys-con.com/read/41844.htm>
- "Toward Better Error Handling – Part 3 (Application-Level Error Handling)" (January 2001) <http://coldfusion.sys-con.com/read/41870.htm>

- “Toward Better Error Handling – Part 4 (Page-Level Error Handling)” (June 2001) <http://coldfusion.sys-con.com/read/41772.htm>

Ben Forta also wrote on error handling both before and after that series:

- “Expect The Unexpected” (October, 1999) <http://coldfusion.sys-con.com/read/41494.htm>
- “To Err Is Human, to Gracefully Handle Errors Is Divine” (March 2002) <http://coldfusion.sys-con.com/read/41727.htm>

New Features Since CFMX

All those came out before CFMX and there have been a couple more changes since then. First, in CFMX (6.0), we were given the ability to do **try/catch** processing in CFSCRIPT tags. Sadly, there's no corresponding **rethrow** statement (or a throw to match the CFTHROW tag mentioned later), but you can find a user-defined (UDF) function for the latter written by Ray Camden at the cflib.org site: <http://www.cflib.org/udf.cfm/throw>. For now, there is no rethrow UDF.

Finally and more recently CFMX 7 added the new **application.cfc** file to replace the older application.cfm. This has also introduced another approach to error handling in its **onError()** method. Nik Molnar has discussed this in his article “Macromedia ColdFusion Debugging – Don't Forget Your Bug Spray!” (September 2005) at <http://coldfusion.sys-con.com/read/122162.htm>.

Because BlueDragon doesn't support application.cfc yet, this is the one feature discussed so far that it can't support.

Of course, all these error-handling mechanisms are covered in the ColdFusion documentation, including both the CFML Reference Manual and the Developers Guide, as well as the Administration manual, all available online at <http://livedocs.macromedia.com/coldfusion/>.

Reporting the Occurrence of Errors

The final topic on error handling regards the reporting of errors. Even if you do improve the quality of what the user is shown, will you even know that the error has occurred? This is a bigger problem that many applications never properly address. It's not appropriate if your user is complaining to your manager (or leaving your commercial Web site) because of an error occurring while you never knew it.

The only built-in mechanism in ColdFusion to report such errors is the **application.log** file available in the /logs directory where ColdFusion is installed (the /log under CF 5). Still, not many are going to monitor that closely. Additionally, it's just one file for the entire server. If you have multiple applications installed, it really won't help solve the problem of notification.

Instead, you can leverage the error-handling pages discussed above to create some form of logging on your own to track the errors. Recall the suggestion of using CFMAIL to send a notification.

Still another idea is to store the error in a log file or database record as discussed in the article “Tracking Errors: How Good Is Your Code?” (Joe Danziger, June 2003) <http://coldfusion.sys-con.com/read/41618.htm>.

Debugging: Resolving Errors

When it comes to resolving errors, there are a variety of tags and techniques to assist you. Some are features to help you while running a request to provide additional information to track the program's state, while another is the ability to do interactive step debugging, and the last are related to getting system diagnostics. We don't have room to elaborate on these, but do be aware of them.

Additional Info To Track Program State

The first set of debugging tools, some of which are widely used while others are still a surprise to many, are the debugging info enabled in the Admin console, CFDUMP, CFLOG, CFTRACE, CFTIMER, and the cfstat command-line utility (BlueDragon doesn't currently support the last two). Even CFOUT-PUT and CFABORT can be useful techniques when debugging. These topics have been covered in more depth in the following articles (and of course, the documentation):

- “Debugging: Tips and Tricks for Detecting Errors” (Kevin Schmidt, October 2001) <http://coldfusion.sys-con.com/read/41827.htm>
- “Defending Socrates” (Hal Helms, October 2001) <http://coldfusion.sys-con.com/read/41810.htm>
- “A More Thorough Debugging” (Eric Brancaccio, April 2002) <http://coldfusion.sys-con.com/read/41737.htm>
- “Macromedia ColdFusion Debugging – Don't Forget Your Bug Spray!” (Nik Molnar, September 2005) <http://coldfusion.sys-con.com/read/122162.htm>
- “How to Debug Your ColdFusion Applications” (Jeff Houser, October 2005) <http://coldfusion.sys-con.com/read/122135.htm>
- “Debugging ColdFusion MX7” (Jason Heaslet, October 2005) <http://coldfusion.sys-con.com/read/138977.htm>

There is also a new article this month related to the above: “Faster debugging with CFTIMER and CFTRACE”, by Shlomy Gantz.

I'd also like to point out that BlueDragon offers a useful tag called CFDEBUGGER to supplement those above. It traces all the lines of CFML code executed in a given request, something that CF developers have never had. I wrote about it in a past

"Errors will happen. Are you prepared to handle them gracefully and debug them skillfully? Are you testing to prevent them in the first place?"

</cf_bugs>



FusionDebug™

The new interactive CF debugger.



www.fusiondebug.com

Using FusionDebug™ you can:

- interactively step through CF code
- set breakpoints
- step directly into custom CF tags and CFC's
- set watch expressions
- examine and modify variables during execution
- view stack frames
- integrate with CFEclipse and the Eclipse editor

CFDJ article: CFDEBUGGER (November 2003) <http://coldfusion.sys-con.com/read/42101.htm>.

Interactive Step Debugging

Of course, when some think of debugging they immediately think of interactive “step” debugging, and those folks have long complained that CFML lacked such a capability (while some others don’t miss it at all). As we went to press, the folks at Fusion-Reactor.com (Integral GMBC) released their FusionDebug tool, which is an Eclipse plug-in that finally provides step debugging for ColdFusion MX. Learn more at <http://www.fusion-reactor.com/fusiondebug>. It’s something I’m excited about. While many never knew it, we did indeed have debugging available in ColdFusion 5 and before by way of ColdFusion Studio (or today’s HomeSite+). I’ve mentioned it in, “Browsing within CF Studio/HomeSite+” (August 2003) <http://coldfusion.sys-con.com/read/42061.htm>. I even walked folks through using it in a 1999 presentation, available at <http://www.systemanage.com/presentations/debugger-setup.pdf>. One reason I never wrote about it here is that it didn’t always work. Still, I’ll point out that the official documentation, which many never knew existed, was in Chapter 9 of *Using ColdFusion 5 Studio* at http://livedocs.macromedia.com/coldfusion/5.0/Using_ColdFusion_Studio/debug.htm. It should be possible to get it working with CF Studio and/or HomeSite+ with ColdFusion 5, if you still have them both.

System Diagnostics

Of course, sometimes to understand error situations you need more information than just what’s reflected in running the current page. There could be other activities or problems in the environment that are impacting your code. When it comes to gathering diagnostics to understand and resolve problems, you should consider taking advantage of the built-in logs, available in the aforementioned ColdFusion logs directory. There’s also an interface to view them in the Admin console.

Beyond that, though, there’s even more powerful analysis of what’s going on in your environment available by way of two commercial (yet reasonably priced) tools that have come out in the past year or so: SeeFusion at www.seefusion.com and FusionReactor at www.fusion-reactor.com. Both are available on a free trial basis so check them out.

There may be problems that are outside of ColdFusion that are affecting you. Be sure to consider analyzing your Web server, database, network, and the operating environment of the machine running ColdFusion. There are logs and analysis tools for each that are beyond the scope of this article.

Finally, another vital tool when solving problems in CFML is simply knowing what you should and shouldn’t be doing in CFML. Don’t forget that you have documentation available at livedocs.macromedia.com (no, the domain name hasn’t changed yet). And more than just the CFML reference, there’s a 900+ page Developer Guide and more.

Testing: Preventing Errors in the First Place

While it’s admirable to do a better job catching, handling, and debugging errors when they occur, shouldn’t we try to prevent them in the first place? There are several ways you may

be able to test your code to prevent errors occurring at all, as we will discuss, ranging from unit testing to syntax checking to validating your page input and output.

Unit Testing

To prevent bad code from getting into production many would recommend doing better **unit testing**, a subject within the broader topic of test-driven development. CFML offers various implementations of such unit-testing tools ranging from **test harnesses** to **cfunit** and **cfcunit**, as discussed in:

- “Tipping Points: Little Things That Make a Big Difference” (Hal Helms, May 2000) <http://coldfusion.sys-con.com/read/41938.htm>
- “Testing for Smarties” (Hal Helms, June 2003) <http://coldfusion.sys-con.com/read/41899.htm>
- “ColdFusion Unit Testing Framework” (Harry Klein, September 2004) <http://coldfusion.sys-con.com/read/46361.htm>
- “Extreme Programming” (Hal Helms, October 2004) <http://coldfusion.sys-con.com/read/46792.htm>
- <http://cfcunit.sourceforge.net/>
- <http://www.cfcunit.org/cfcunit/>

Note that cfunit used to be part of the Developer Resource Kit (DRK) but it’s now available at the site above. There are still other forms of testing to keep in mind.

CFML Syntax Checking

Since CFML isn’t a compiled language, many errors slip by because we don’t need to “run a compile step” before releasing our code into the wild. We may have syntax errors that would cause a page to fail as soon as it’s run. How can we catch them without actually running every page? Again, good unit tests would help, but until you implement them, there’s still another approach.

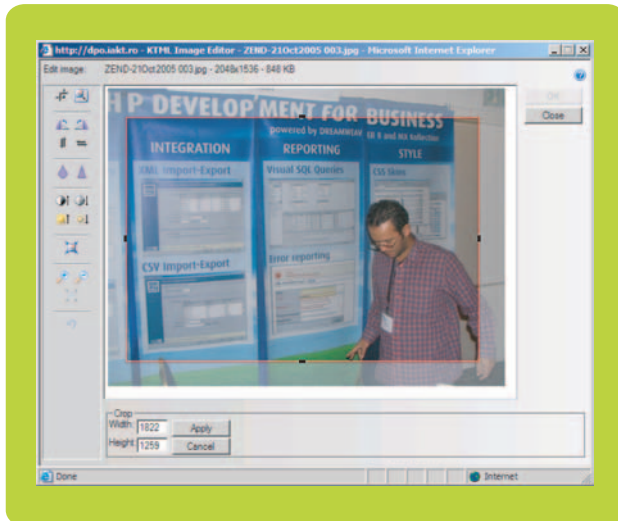
Did you know that there are various means to check your CFML code’s syntax before releasing it? Since release 4, the Admin console has had a mechanism that lets you point to a directory of code. It will analyze your CFML and report if there are problems. While in CFMX it’s formally called a **Code Compatibility Analyzer**, with a focus on checking code from previous releases, it also does CFML syntax checking (see the “Advanced Options”). There has also been one in CF 4 and 5. Note that you can point it at an entire directory (and optionally recurse into subdirectories), so it’s definitely something to consider as a regular part of testing. See the online help in the Admin console for more information.

Perhaps you’re perturbed that that tool is a Web interface requiring you to enter data in a form. You want a command-line tool. I’m not aware of a means to call that compatibility analyzer from the command line, but did you know that CFMX 6.1 added a command-line **precompile** tool? It’s in the /bin directory as **cfcompile.bat**. While the primary purpose is performance improvement (saving the auto-compile that takes place on first load of the page into server memory), it will report if there are any errors during evaluation of the CFML, so it can act very much like the missing compilation step that other developers use to catch errors early. Again, it is a command-line tool, and it

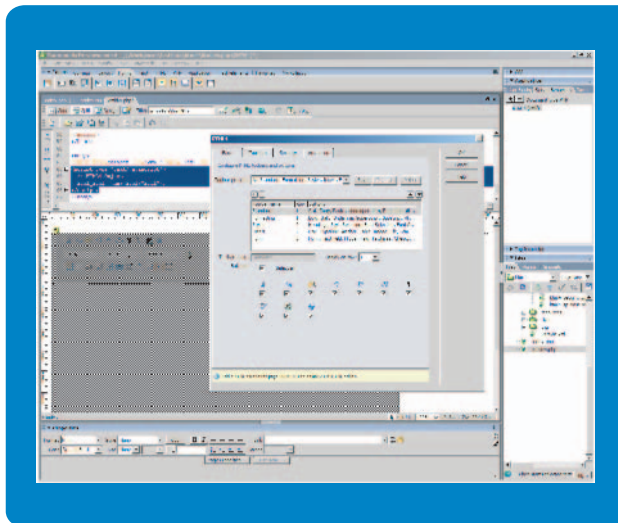
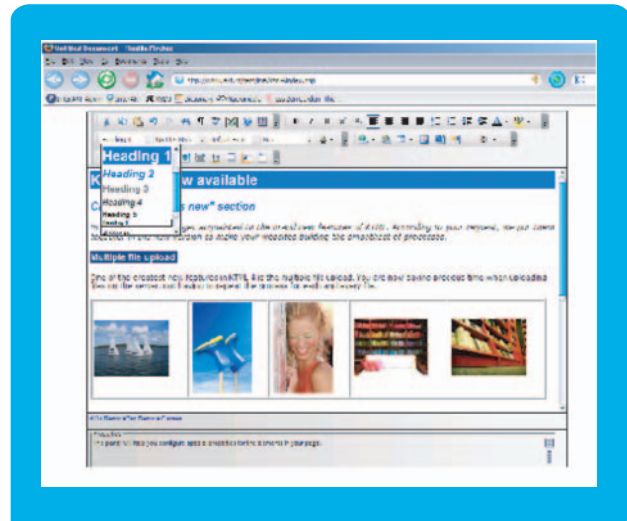
KTML4 Word editing in browser

Along with the most affordable **UNLIMITED** licence

Revolutionary Image Editor



Word-like visual CSS styles



Fast Dreamweaver integration

- Instant paste from Word
- Incredible speed
- Easy to use Word-like toolbars
- Improved CSS authoring
- Remote File Explorer
- XHTML 1.1 compliant

Wide browser compatibility

- Multiple file upload at once
- HTML Table Editor
- Support for multimedia (Flash, Avi)
- Documents management (.doc, .pdf)
- Page templates
- WAI compliant

Please visit www.interaktonline.com/ktml4/ for details

work smart

Interakt

does by default compile all templates in a given directory.

See Chapter 5 of the CFMX 7 Administration manual (but not in the CFMX 6.1 manual) for more information. It's also discussed in the following article. Though the focus is on J2EE Web application deployment, the discussions of cfcompile are still broadly useful: "Deploying Applications with ColdFusion MX 7" (Dave Carabetta, March 2005) <http://coldfusion.sys-con.com/read/48654.htm>.

I'd like to point out also that I had discussed the benefits of such a precompile tool in a two-part series. Though it was written before the tool was released officially in 6.1, some of the info may still benefit:

- "Compilation and Precompilation in CFMX Templates--Part 1" (Oct 2002) <http://coldfusion.sys-con.com/read/41672.htm>
- "Precompiling CFML Templates in CFMX--Part 2" (Nov 2002) <http://coldfusion.sys-con.com/read/41684.htm>

Input and Security Validation

Still another form of testing to consider before you turn your application out for production implementation is a range of tests that focus on ensuring that the application avoids errors by preventing inappropriate input or cross-site scripting as discussed in:

- "Safe Scripting" (James A. Brannan, June 2000) <http://coldfusion.sys-con.com/read/41943.htm>
- "Untrusted Data Sources" (Jackson Moore, October 2001) <http://coldfusion.sys-con.com/read/41818.htm>
- "ColdFusion Security Best Practices" (Bryan Murphy, September 2004) <http://coldfusion.sys-con.com/read/46358.htm>
- "Top 10 Web Security Tips" (Michael Smith, September 2004) <http://coldfusion.sys-con.com/read/46366.htm>
- "Strip Tease" (Isaac Dealey, April 2006) <http://coldfusion.sys-con.com/read/206288.htm>

Output Validation

Other forms of testing focus on the output of your pages such as validating your HTML code (accessibility checking, CSS validation, link checking, and spell checking) as well as regression, load, and concurrency testing. I covered all of these in "E-Testing: Debugging Your Projects" (October 2001) <http://coldfusion.sys-con.com/read/41816.htm>.

Other Error-Related Topics

There are still other topics that are tangentially (or perhaps directly) related to error handling, including handling errors from CFMAIL, using the CFTHROW tag, doing site monitoring, load testing, and leveraging version control to manage your code better and recover from erroneous code releases. Again, there's no room to elaborate, but I'll point out one more article that covers a range of topics including error handling, testing, and source control: "7 Habits of Highly Effective ColdFusion Developers" (Robi Sen, February 2004) <http://coldfusion.sys-con.com/read/43534.htm>.

And while I'm mentioning error-handling articles, some will

get value out of a past article on handling errors in JavaScript: "Error Handling in JavaScript" (Steve Bryan, October 2002) <http://coldfusion.sys-con.com/read/41671.htm>.

Finally, I'll also point out that in this month's issue there is an article from Joshua Curtiss on yet another error-handling concept, "Handling 404 Errors for a Migrated Blog".

I hope this trip through the archives of the CFDJ focusing on error handling, debugging, and testing will help improve the quality of your CFML development. 

About the Author

A veteran CFML developer since 1997, Charlie Arehart is a longtime contributor to the community and was recently selected to the Adobe Community Expert program. Many know he served as tech editor of the CFDJ until 2003 and was co-author of the CFMX Bible. A certified Advanced CF Developer and Instructor for CF 4/5/MX, he's frequently invited to speak to developer conferences and user groups worldwide. Formerly CTO of New Atlanta (BlueDragon), he is now an independent contractor and still lives in Alpharetta GA where he is president of the Atlanta CFUG.

charlie@carehart.org

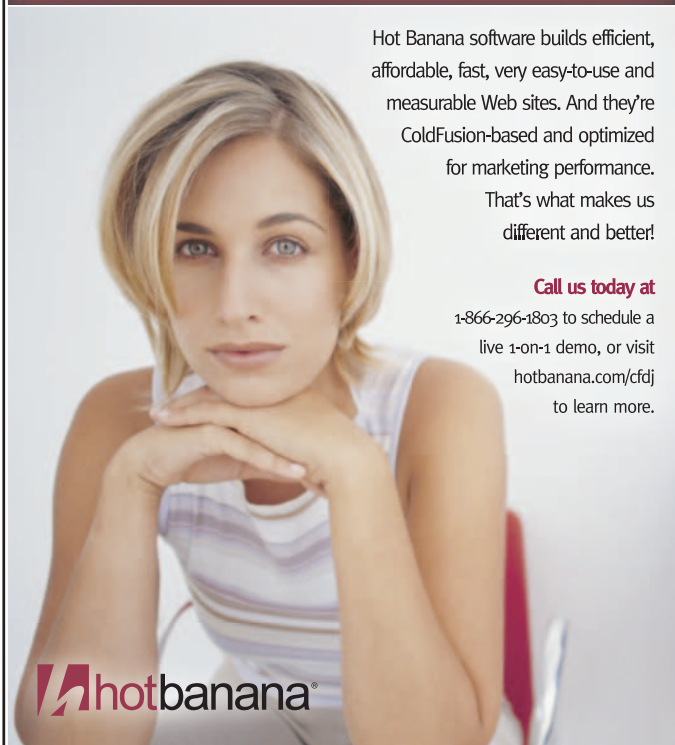
CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
Adobe	www.adobe.com		3,51
AjaxWorld	www.ajaxseminar.com	201-802-3022	30-31
CFDynamics	www.cfdynamics.com	866-233-9626	2
CFDJ	http://coldfusion.sys-con.com/	800-303-5282	35
CF-Underground	www.cf-underground.com		6
CommunityMX	www.communitymx.com/trial		11
EdgeWebHosting	edgewebhosting.net	1-866-334-3932	4
Hal Helms	halhelms.com		25
HostMySite	www.hostmysite.com	877-215-4678	17, 52
Hotbanana	hotbana.com/cfdj	866-296-1803	25
Integral	www.fusiondebug.com		21
InterAKT	www.interaktonline.com/		22
IT Solutions Guide		201-802-3021	
iTVcon.com	www.itvcon.com	201-802-3023	39
Paperthin	www.paperthin.com	1-800-940-3087	12
VitalStream	www.vitalstream.com/tools/mxdj.asp	800-254-7554	13-14

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 discretion 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.

Get more with Hot Banana!

Web Content Management + Active Marketing



Hot Banana software builds efficient, affordable, fast, very easy-to-use and measurable Web sites. And they're ColdFusion-based and optimized for marketing performance.

That's what makes us different and better!

Call us today at

1-866-296-1803 to schedule a live 1-on-1 demo, or visit hotbanana.com/cfdj to learn more.

"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.

REPRINT IT!

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 Megan Mussa
201 802-3024
megan@sys-con.com

Reprints

SYS-CON
MEDIA

Multiple-File Uploads with ColdFusion

Letting users pick whatever files they want to upload and then pushing them all at once

By Dave Shuck

One task that arises again and again in ColdFusion application development is the need for users to be able to upload multiple files to the Web Server. Back in the stone ages (and by this I mean more than a couple of years ago) we had a limited set of options to be able to do this. Usually a problem such as this was approached using one of three means.

Perhaps most crudely, the user would be able to load a file at a time. And if they needed to upload 10 files, that meant 10 round trips to the server and a heck of a lot of clicks. To paraphrase Hal Helms who, I believe, was paraphrasing someone else, this can create a disorder known as Post Back Traumatic Stress Syndrome.

To add a level of elegance, some of us began prompting the user ahead of time: "How many files do you want to upload?" This would be followed by an ugly page with rows and rows of file-type input boxes. And what if the user changed his mind about how many files? Well, he can always start over.

And sometimes we'd just take the approach of "knowing" that the user wouldn't need to upload more than x number of files, so we would just put x number of input boxes on a page, and they could use as many as they needed...unless of course they needed more, in which case they could just repeat the process.

What is the common thread here? While each of these are indeed functional approaches, none of them provides the level of user experience that our users are beginning to expect in Web applications thanks to the growing implementation of rich JavaScript libraries, AJAX, and various RIA technologies.

Wouldn't it be nice if there were some free tool that lets you give your users the ability to choose whatever files they want to upload and then push them all at once? Well, yes, it would and thankfully there is such a tool written by a developer who simply goes by the name Stickman as in www.the-stickman.com.

This tool lets you quite easily allow users to upload multiple files using a single-file input element. Once users have added files to a queue they can then push them all at once. If you're familiar with adding attachments in Gmail you know this concept well. We're going to provide a very simplified example and within a few minutes you can give your user an interface as shown in Figure 1.

Let's look at a step-by-step approach for adding this to your site. First, we need to get the JavaScript file that makes this process so easy. To do this, we need to download the zip file at <http://the-stickman.com/files/multiple-file-element.zip>. You'll find three files inside this zip file:

- example.html – This HTML file is the of the ColdFusion example we're about to build.
- multifile.js – The JavaScript library used by the HTML.
- multifile_compressed.js – This file is functionally the same as the previous file, the difference being the overall size of the compressed file by removing white space.

For our example we'll place the last of the three, multifile_compressed.js, in our application's webroot.

The next step is completely superficial and is simply included to stylize the form just a bit, but we'll create a simple CSS file called uploader.css and put that in our webroot. If you're in a big hurry, you can skip this step without any consequences in running the uploader. However, if you want your output to look like the example here, your CSS file should contain the following shown in Figure 2.

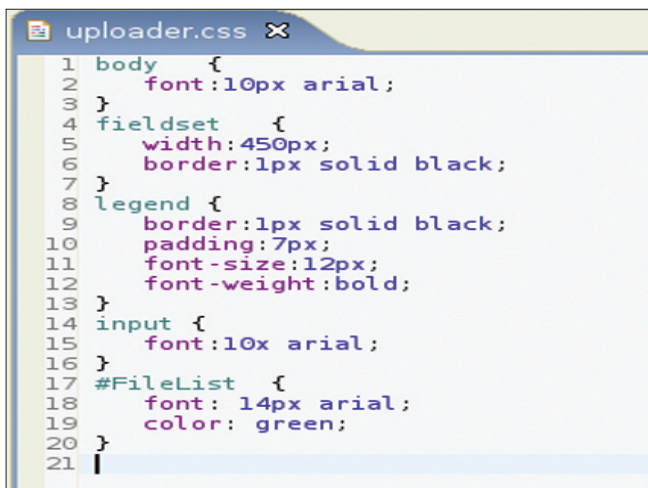
Now it's time to make the form our users are going to use to upload files. The upload form template looks like this Figure 3. As you can see in that code, we call the multifile_com-

Figure 1

pressed.js file in the head of our document in addition to linking the CSS file we created. In the body, you find that our form has a single-file input element. This is used to add files to the upload queue. As files are added, the script actually adds new file-input elements and hides them by absolutely positioning them 1000px to the left of the top-left corner of the screen. The author took this approach as opposed to setting the style equal to “display: none” to allow compatibility with Safari browsers.

Once the new input element is added, the addListRow() JavaScript function is called that writes a new row in the “FileList” div element. By default this row displays the name of the new file and a button to remove it before form submission, but this function can be easily modified to make this display appear however you wish.

Now that we’ve created our form page, it’s time to manage the action once the form is submitted. First, we should start by creating a directory to receive the uploaded files. In our example, we’ll create a sub-directory in the webroot called “upload.”

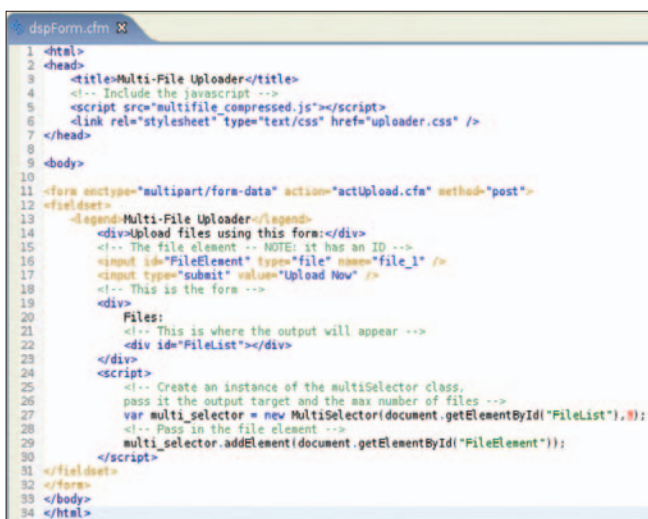


```

1 body {
2     font:10px arial;
3 }
4 fieldset {
5     width:450px;
6     border:1px solid black;
7 }
8 legend {
9     border:1px solid black;
10    padding:7px;
11    font-size:12px;
12    font-weight:bold;
13 }
14 input {
15     font:10px arial;
16 }
17 #FileList {
18     font: 14px arial;
19     color: green;
20 }
21

```

Figure 2



```

1 <html>
2 <head>
3     <title>Multi-File Uploader</title>
4     <!-- Include the javascript -->
5     <script src="multifile.compressed.js"></script>
6     <link rel="stylesheet" type="text/css" href="uploader.css" />
7 </head>
8
9 <body>
10
11 <form enctype="multipart/form-data" action="actUpload.cfm" method="post">
12 <fieldset>
13     <legend>Multi-File Uploader</legend>
14     <div>Upload files using this form:</div>
15     <!-- The file element -- NOTE: it has an ID -->
16     <input id="FileElement" type="file" name="file_1" />
17     <input type="submit" value="Upload Now" />
18     <!-- This is the form -->
19     <div>
20         Files:
21         <!-- This is where the output will appear -->
22         <div id="FileList"></div>
23     </div>
24     <script>
25         <!-- Create an instance of the multiSelector class.
26              pass it the output target and the max number of files -->
27         var multi_selector = new MultiSelector(document.getElementById("FileList"), 1);
28         <!-- Pass in the file element -->
29         multi_selector.addElement(document.getElementById("FileElement"));
30     </script>
31 </fieldset>
32 </form>
33 </body>
34 </html>

```

Figure 3



```

1 <cfif StructKeyExists(form,"fieldnames")>
2     <cfset variables.UploadedFiles = ArrayNew(1) />
3     <cfloop list=#form.fieldnames# index="i">
4         <cfif FindNoCase("file_",i) AND Len(form[i])>
5             <cffile action="upload"
6                 filefield="#i#"
7                 destination="#ExpandPath("/upload")#"
8                 nameconflict="makeunique"
9                 result="ThisFile" />
10             <cfset ArrayAppend(variables.UploadedFiles,ThisFile) />
11         </cfif>
12     </cfloop>
13
14     <cfdump var=#variables.UploadedFiles# />
15 </cfif>
16

```

Figure 4

Next we have to write our action page. Create a new template in the webroot called “actUpload.cfm” and include the following code shown figure 4.

So what exactly are we doing here? Let’s walk through it.

First, and hopefully quite obviously, we are ensuring that a form submission has actually occurred to get us to this page. On any form post in ColdFusion when the method type is set to “post,” there will be a variable “fieldnames” present in the form scope that is a list of the names of all the form elements that were passed to the action page. This multi-file uploader script creates fields that begin with the string “file_” and end with an incrementing value beginning with 0. There will always be one more element than the number of files that are being uploaded, and that element will be an empty string. For instance, when we upload three files we’ll have: file_0, file_1, file_2, and then file_3, which is an empty value. So in the case of three submitted files, and assuming no other fields were present in the form, #form.fieldnames# would output as: “FILE_0,FILE_1,FILE_2,FILE_3.”

On line 2, we’re creating a new array, which will be used later to house the results of each file upload.

Then we begin looping through our #form.fieldname# list. If the current element name begins with “file_” and the form variable isn’t an empty string, we’re going to attempt to upload this file to the server.

You can see in the <cffile/> tag that we’re storing the results of the upload process in a variable called “ThisFile.” We then append that result structure to our UploadedFiles array. Once we’re done, we simply dump the array to the screen. In an actual production scenario, you’d obviously want to trap more errors and probably do something more intelligent with the results of the uploads before redirecting the user to a confirmation page.

Providing your user with tools such as this can affect their overall perception of your site, and subtle things such as this can really boost their “feel good” factor. While there are other ways to achieve the same results, such as the commercially licensed CF_ProFlashUpload, or various other homegrown concoctions, Sickman’s multiple-file element uploader is a quick, easy, and flexible choice.

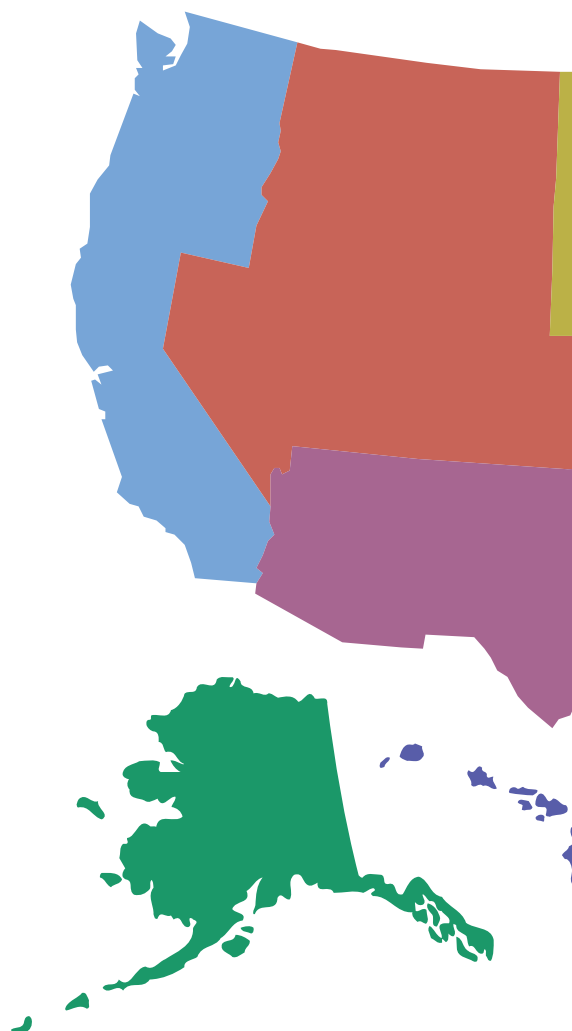
dshuck@gmail.com

ColdFusion

For more information go to...

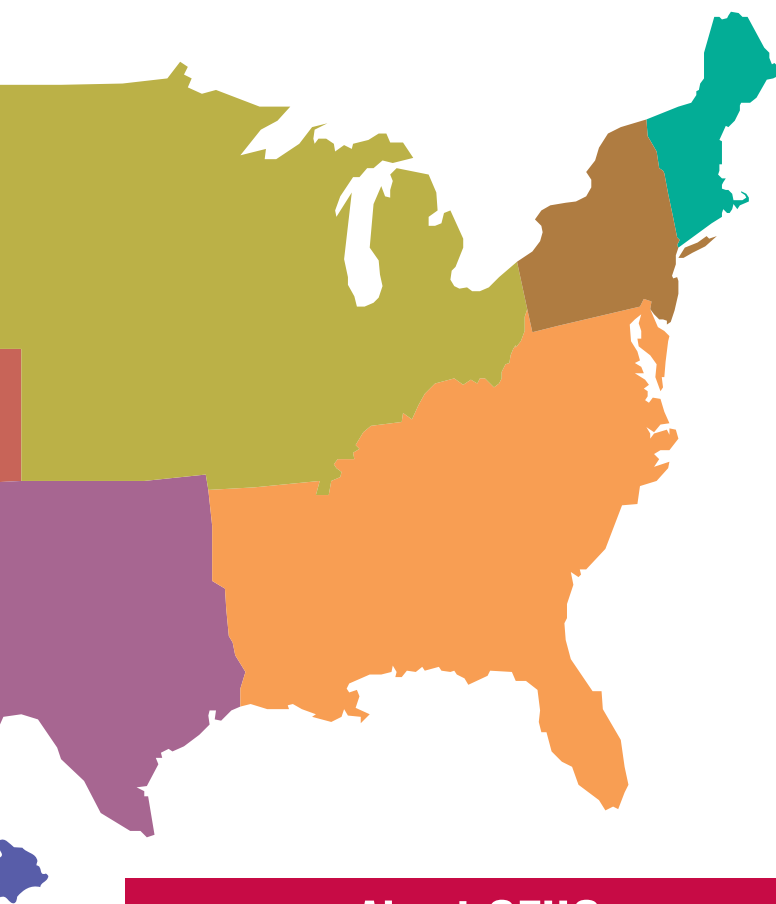
U.S.

Alabama Huntsville, AL CFUG www.nacfulg.com	Louisiana Lafayette, LA MMUG http://www.acadianammug.org/	New York Albany, NY CFUG www.anycfug.org
Arizona Phoenix, AZ CFUG www.azcfug.com	Maryland California, MD CFUG http://www.smdcfug.org	New York New York, NY CFUG www.nycfug.org
California Bay Area CFUG www.bacfulg.net	Maryland Maryland CFUG www.cfug-md.org	New York Syracuse, NY CFUG www.cfugcny.org
California Sacramento, CA CFUG http://www.sacfulg.com/	Massachusetts Boston CFUG http://bostoncfug.org/	North Carolina Raleigh, NC CFUG http://tacfulg.org/
California San Diego, CA CFUG www.sdcfulg.org/	Massachusetts Online CFUG http://coldfusion.meetup.com/17/	Ohio Cleveland CFUG http://www.clevelandcfug.org
Colorado Denver CFUG http://www.denvercfug.org/	Michigan Detroit CFUG http://www.detcfulg.org/	Oregon Portland, OR CFUG www.pdxcfug.org
Connecticut SW CT CFUG http://www.cfugitives.com/	Michigan Mid Michigan CFUG www.coldfusion.org/pages/index.cfm	Pennsylvania Central Penn CFUG www.centralpenncfug.org
Connecticut Hartford CFUG http://www.ctmug.com/	Minnesota Southeastern MN CFUG http://www.bittercoldfusion.com	Pennsylvania Philadelphia, PA CFUG http://www.phillycfug.org/
Delaware Wilmington CFUG http://www.bvccfulg.org/	Minnesota Twin Cities CFUG www.colderfusion.com	Pennsylvania State College, PA CFUG www.mmug-sc.org/
Florida Jacksonville CFUG http://www.jaxcfusion.org/	Missouri Kansas City, MO CFUG www.kcfusion.org	Tennessee Nashville, TN CFUG http://www.ncfulg.com
Florida South Florida CFUG www.cfug-sfl.org	Nebraska Omaha, NE CFUG www.necfulg.com	Tennessee Memphis, TN CFUG http://mmug.mind-over-data.com
Georgia Atlanta, GA CFUG www.acfulg.org	New Jersey Central New Jersey CFUG http://www.cjcfug.us	Texas Austin, TX CFUG http://cftexas.net/
Illinois Chicago CFUG http://www.cccfulg.org	New Hampshire UNH CFUG http://unhce.unh.edu/blogs/mmug/	Texas Dallas, TX CFUG www.dfwcfug.org/
Indiana Indianapolis, IN CFUG www.hoosierfusion.com	New York Rochester, NY CFUG http://rcfulg.org/	Texas Houston Area CFUG http://www.houcfug.org



User Groups

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



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
<http://www.actcfug.com>

Australia
Queensland CFUG
<http://qld.cfug.org.au/>

Australia
Victoria CFUG
<http://www.cfcentral.com.au>

Australia
Western Australia CFUG
<http://www.cfugwa.com/>

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfugtoronto.org

Germany
Central Europe CFUG
www.cfug.de

Italy
Italy CFUG
<http://www.cfmentor.com>

New Zealand
Auckland CFUG
<http://www.cfug.co.nz/>

Poland
Polish CFUG
<http://www.cfml.pl>

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

Spain
Spanish CFUG
<http://www.cfugspain.org>

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

Switzerland
Swiss CFUG
<http://www.swisscfug.org>

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org



Rich Internet Applications: AJAX,

www.AjaxWorldExpo.com

AJAXWORLDTM

CONFERENCE & EXPO

SANTA CLARA SILICON VALLEY

**SYS-CON Events is proud to announce the first-ever
AjaxWorld Conference & Expo 2006!**

**The world-beating Conference program will provide developers and IT managers alike
with comprehensive information and insight into the biggest paradigm shift in website design,
development, and deployment since the invention of the World Wide Web itself a decade ago.**

The terms on everyone's lips this year include "AJAX," "Web 2.0" and "Rich Internet Applications." All of these themes play an integral role at AjaxWorld. So, anyone involved with business-critical web applications that recognize the importance of the user experience needs to attend this uniquely timely conference – especially the web designers and developers building those experiences, and those who manage them.

CALL FOR PAPERS NOW OPEN!

We are interested in receiving original speaking proposals for this event from i-Technology professionals. Speakers will be chosen from the co-existing worlds of both commercial software and open source. Delegates will be interested learn about a wide range of RIA topics that can help them achieve business value.

Flash, Web 2.0 and Beyond...

REGISTER TODAY AND SAVE!

→ **October 3-4, 2006**

→ **Santa Clara Convention Center**
Hyatt Regency Silicon Valley
Santa Clara, CA

→ **To Register**

Call 201-802-3020 or
Visit www.AjaxWorldExpo.com

→ **May 7-8, 2007**

First International AjaxWorld Europe
Amsterdam, Netherlands

“Over the two information-packed days, delegates will receive four days’ worth of education!”

Early Bird*

(Register Before August 31, 2006)
..... **\$1,495****
See website or call for group discounts

Special Discounts*

(Register a Second Person)
..... **\$1,395****
See website or call for group discounts

(5 Delegates from same Company)
..... **\$1,295/ea.****
See website or call for group discounts

On-Demand Online Access

(Any Event)
..... **\$695**

*Golden Pass access includes Breakfast, Lunch and Coffee Breaks, Conference T-Shirt, Collectible Lap-Top Bag and Complete On-Demand Archives of sessions in 7 DVDs!

**OFFER SUBJECT TO CHANGE WITHOUT NOTICE,
PLEASE SEE WEBSITE FOR UP-TO-DATE PRICING

“It Was The Best AJAX Education Opportunity Anywhere in the World!” —John Hamilton

Topics Include...

Themes:

- > Improving Web-based Customer
- > Interaction
- > AJAX for the Enterprise
- > RIA Best Practices
- > Web 2.0 – Why Does It Matter?
- > Emerging Standards
- > Open Source RIA Libraries
- > Leveraging Streaming Video

Technologies:

- > AJAX
- > The Flash Platform
- > The Flex 2 Framework & Flex Builder 2
- > Microsoft's approaches:
ASP.NET, Atlas, XAML with Avalon
- > JackBe, openLaszlo
- > JavaServer Faces and AJAX
- > Nexaweb
- > TIBCO General Interface

Verticals:

- > Education
- > Transport
- > Retail
- > Entertainment
- > Financial Sector
- > Homeland Security

GROUP DISCOUNTS AVAILABLE:
— 5 Delegates from Same Company —
for only \$995 (each)
— Register a Second Person —
for only \$1195

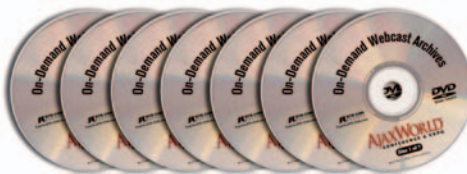
**Hurry! Limited Seating
This Conference Will Sell-Out!**



LIVE SIMULCAST!
AROUND THE WORLD ON SYS-CON.TV

Receive **FREE**
WebCast Archives
of Entire Conference!

The best news for this year's conference delegates is that your "Golden Pass" registration now gives you full access to all conference sessions. We will mail you the complete content from all the conference sessions in seven convenient DVDs after the live event takes place.



► This on-demand archives set
is sold separately for \$995

HYATT
HYATT REGENCY SILICON VALLEY



SYS-CON
EVENTS

For more great events visit www.EVENTS.SYS-CON.com

VISIT WWW.AJAXWORLDEXPO.COM FOR THE MOST COMPLETE UP-TO-DATE INFORMATION

Developing Flex 2 Applications with ColdFusion and XML Without Needing FDS or Mystic

Solutions for the low-budget developer using HTTP services



By Faisal Abid

Flex is one of the greatest technologies

around. Combine it with ColdFusion and it

just gets better. Adobe has made it possible

to use Flex 2 efficiently with ColdFusion

and easily get data across the server to a Flex front-end.

It's also released Flex data services that don't need ColdFusion to pass data in and out. However, these technologies might be outside the price range of a small developer or hard to deploy. Take FDS, for example. Of course Adobe has an Express edition, but deploying a good application for all the world to see hosted on your own computer is a hassle, and if you were to go out and look for a host that supports FDS it would cost you about \$360 a year. Not a lot for some, but for an individual developer who's already paying for a solid ColdFusion host, it may be too much.

Another technology that Adobe introduced was an update to Flash remoting in ColdFusion 7.0.2, yet most hosting providers won't support Flex connectivity on their servers, and again you'd have to buy a VPS, which would cost you \$360 a year. Nevertheless, in the world of dynamic technology there are other alternatives. And ColdFusion-generated XML or what some might consider HTTP services is one of them. This is an easy way to get data from your MySQL, Access, and Oracle database and bring it into Flex, read it, update it, and if you want even delete records based on the user level. This article will discuss the basics of using this method, teach you how to build ColdFusion-based XML files,



and bring them in and explain how you can use them in your Flex application.

The Background

Using XML as a data delivery source has been around since the dawn of Web 2.0 or before. Almost all Web 2.0 Web sites use XML as a method of delivering news using RSS to their users. Although this is a good way to use XML there are better ways. Instead of transporting news in XML what if you can transport the user data and authentication. Many sites such as Flickr and Blogger have an API that lets you to do this and, in fact, this is a good solution. Rather than getting FDS to retrieve data, we just encapsulate the data in an XML file that in turn is read by Flex and displayed. No Mystic, No FDS, just plain ColdFusion, XML, and Flex.

Why Use This Method over WSDLs, FDS, or Mystic?

There are different ways of getting data from a database and bringing it into your front-end like Web Services (WSDL). However, not a lot of developers who are starting off know how to create a WSDL or they don't have the time to learn how. Using ColdFusion-generated XML basically means creating a query and outputting it through some XML and cfloops. Any ColdFusion developer who is starting off can quickly harness this power and create applications in record time. After all, developing applications that are in demand is a race. Using this method is not restricted to Flex applications, you can create an application in virtually any modern programming language without modifying a single line of back-end ColdFusion code. How? Well, because XML is read, as I said, by any modern programming language, all you have to do is "hook up" the XML to the front-end via an HTTP service call. The application we're going to build in this article makes use of the HTTP service call rather than a Web Service call, and gets all the data back from the server and then the user can manipulate the data.

Another great reason to use this type of programming method is because you can “open” the back-end files or give the links to developers who are masters in their own field who can then develop third-party front-ends using your back-end. This can extend the reach of your site and deliver your data to more people, and will entice developers into creating mashups, where they take your data and using another API (as this is called) mash them up to create one heck of an application. An example would be if your application was a user-based contact manager, and after opening the API to developers, they could take the data and mash it into Google maps, where when you click on the contact it will show where the house is based on the ZIP code.

Time to Build Your Own Application

This isn't your boring old computer science class where all you did was learn theory and learn and learn. As a result instead of telling you more about this type of programming, why not just dive in and learn on the way. The Flex 2 application we're going to build will be an RSS manager, which will have user authentication and a place to keep and read all your favorite RSS feeds. (See Figure 1 for the final application.)

What we're going to create is a bunch of objects that will interact with the Flex application via an HTTP service call and then interact with the database via a cfquery. It will be a learning experience. You'll learn the basics of a Flex application and HTTP services. Also since we're not building a sophisticated application, we're not going to have Flex do the authentication. We'll leave that to ColdFusion simply because it's too hard and will take a lot of time otherwise. What this means is that even if the username/password combination is wrong, Flex will let the user pass through to the next step but there will be no data in the data grid.

It's possible to let Flex do the authentication, but this is just a proof-of-concept application, not a “real-world” one. What I hope is that you learn a little more about this method and then you can build on it however you want.

Getting the Tools

Since this is for the individual developer who wants to make the most out of a small budget, we'll be using tools that are completely free. The first thing you should do is go and download ColdFusion, which you should already have anyway. Next you'll want to download the Eclipse platform along with CFECLIPSE (all the links are at the end of this article), which is a good ColdFusion IDE for coding, not designing. Seeing as how we're going to use ColdFusion just for coding CFECLIPSE is the way to go.

Next you'll want to download the Flex Builder and read up on the compiling instructions since this will be your tool to compile Flex applications into Swfs that can be run on virtually any server. Yes, there is the free version of Flex 2 called the Flex 2 SDK However, Flex Builder 2 is more enchancing and will make coding a whole lot easier. I recommend that you download Flex Builder 2. In addition you'll also want to download a database technology

such as MySQL or Oracle and create a DSN in the ColdFusion Administrator.

Architecting the Database

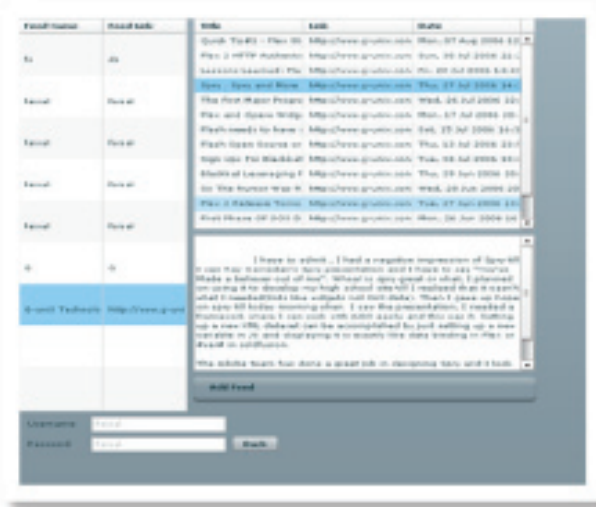
After that it's good practice to lay out your database first and create all the required tables. This way you can build your Flex 2 application and the ColdFusion code relying on the database rather than making guesses on the tables you'll create. So create a new DSN in the CF Admin named global since all the example code uses this DSN. There will be two tables in this simple application that we're going to create. The first will be a user table that will hold all our user data. It will also add the values Faisal, Faisal (user, pass) and the values Abid, Abid (user, pass). If you want, you can add more, but this application is just a concept application. If you're using MySQL as a database then here's the code:

```
CREATE TABLE `CFDJ_users` (
  `member_ID` int(11) NOT NULL auto_increment,
  `Username` varchar(255) NOT NULL,
  `Password` varchar(255) NOT NULL,
  PRIMARY KEY (`member_ID`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

INSERT INTO `CFDJ_users` VALUES (1, 'Faisal', 'Faisal');
INSERT INTO `CFDJ_users` VALUES (2, 'Abid', 'Abid');
```

Then you'll want to create your RSS table. In this table the user will post a link for an RSS feed and a name to go with it, so the user can come back days later and click on the link and get the feed delivered instantly. You're going to need just four fields for this: an RSS ID field that will be your primary, a username field, an RSS link field, and last an RSS name field. If you did this in MySQL here is the code:

```
CREATE TABLE `CFDJ_RSS` (
  `Rss_ID` INT NOT NULL AUTO_INCREMENT ,
  `Username` VARCHAR( 255 ) NOT NULL ,
  `Rss_Name` VARCHAR( 255 ) NOT NULL ,
```



```
'Rss_Link' VARCHAR( 255 ) NOT NULL ,
PRIMARY KEY ( 'Rss_ID' )
);
```

Writing ColdFusion to XML Translation

The next step is to write the middle tier, the ColdFusion code that will generate valid XML depending on the data in the database. We're going to want to create five CFM files that will be called in our HTTP service tag. Why five you ask, and not one WSDL. Well, because we're exploring HTTP services. In Flex you won't be able to call on a Web Service inside an HTTP service tag. If you've ever used the Flickr API, you may have noticed that there's one root URL with different sub-URLs that do different things. This is just an example; don't take this URL seriously. With Flickr.com/get.cfm?tag=32, you can clearly see that you can't get anything more from that URL than just the tags. However, Flickr has a whole range of other methods like SOAP and XML-RPC that will let you to do much more. Nevertheless, since we're doing it the HTTP call way, why not take a step-by-step process to write up these CFM files. You can surely apply these techniques in creating a WSDL. We're not going to have registration capabilities in our application, since it's a simple proof-of-concept application, but if you go to Listing 2 you'll see a register.cfm file you can use if you want registration capabilities.

Login.cfm

The next step is to actually let users use the login.cfm file. Since we've already covered the basics of the code in the Register.CFM section, we'll briefly explain what's happening here and you can see the completed code in Listing 2. First, ColdFusion is checking whether the variables defined in the URL exist in the database or not, and if they don't ColdFusion generates an XML message. However, if the username and password exist then ColdFusion generates a successful message and gets all the feeds in the database attached to the username.

RSS Manager Vitals

Since we're building an RSS manager, it would be useless if we just had an RSS reader. So we're going to quickly write two files, which add and read the RSS feed. We have to build a reader in ColdFusion first. Because of Flash Player sandbox security issues, it won't go directly to the site and read it, instead ColdFusion will be the delivery boy that will go and get the feed and pass it locally to the Flash player.

Building the Flex Front-End (For the final code see Listing 5)

The foundation of this Flex application relies on the CFM files you created in the previous steps; you'll call these files via HTTP services, and hook up the results to the data grids. Since this explores some topics that were in the intermediate group you should already know how to hook up files and create data grids

and so on. I won't walk through the application step-by-step. But I will outline of what's happening in the application.

First, when you load the application in the browser, you'll be prompted with a username and password field. When you insert the right username and password it will make an HTTP call to the login.cfm file with the user and password, and change the state to "feed." If the username and password were successful, you'll be able to add a feed and read it; however, if it wasn't, you'll still be able to go the feed state but won't be able to add or read any feeds.

When you click "add feed," you'll be taken to a new 'add' state that will have a feed name, feed URL field, and save button. Then you'll make a new HTTP call to the file WriteRSS.cfm and add a click handler for the button that will call the WriteRSS.cfm and the login.cfm files so that the data grid refreshes.

Last you'll create your last HTTP call to the ViewRSS.cfm file and you'll add it to the feed URL scope so that it will read the selected item (link) of the data grid that gets all your feeds in the database and then hooks that service to a new data grid. What's going on is that when you click the first data grid, it populates the second data grid with the news from the URL. You'll also create a new text area that will get the description of the selected item in the second data grid, so that when you click on the second data grid it will get the news in the text area.

Well, that's about it for the Flex application. It was simple to build, but with a lot of functionality. As I said before, you can add the register.cfm and make it so that you have to register before you have to use it. You can also add transitions, effects, and cool skins to make your Flex application look pretty. Flex is a technology that is just being discovered, it will soon become the ultimate way to build rich Internet applications and with a growing community it will be as easy to learn as your ABCs.

Links

- G-uniX Technologies: <http://www.G-uniX.com>
- G-uniX Blog: <http://www.G-uniX.com/blog>
- Eclipse Platform: <http://www.Eclipse.org>
- CfEclipse: <http://www.cfeclipse.org/>
- Adobe Flex Builder: <http://www.Adobe.com/flex> 

About the Author

Faisal Abid is a third-year student at West Hill CI and loves Flex, ColdFusion, and rich Internet applications. He is also the founder and owner of G-uniX Technologies (www.G-uniX.com), an RIA software development consultancy based in Toronto that specializes in Flex 2 and ColdFusion applications. He has a blog at blog.G-uniX.com that he updates bi-weekly with G-uniX news, Flex tutorials, ColdFusion tutorials, and Spry tutorials.

faisal@g-unix.com

Listing 1

Register.CFM

```
<cfcontent type="application/xml">
<cfparam name="URL.Username" default="">
<cfparam name="URL.Pword" default="">
<cfquery datasource="global" name="CheckUsername">
```

```
Select Username
FROM CFDJ_users
Where Username = '#URL.Username#'
</cfquery>
<cfif CheckUsername.RecordCount GTE 1>
<application>
```


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 \$29.99 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



```

<dashboard>
<broadcast>
<cfoutput>The Username Is Already Taken, Please Choose A Different
One</cfoutput>
</broadcast>
<value> The Username Is Already Taken, Please Choose A Different One
</value>
<username>#URL.Uname#</username>
</dashboard>
</application>
<cfelse>
<cfquery datasource="global" name="Register">
Insert INTO CFDJ_users (Username,Password)
Values ('#URL.Uname#', '#URL.Pword#')
</cfquery>
<application>
<dashboard>
<broadcast>
<cfoutput>Welcome To G-unix Rss Manager</cfoutput>
</broadcast>
<value>Welcome To G-unix Rss Manager </value>
<username><cfoutput>#URL.Uname#</cfoutput></username>
</dashboard>
</application>
</cfif>

```

Listing 2

```

Login.CFM
<cfcontent type="application/xml">
<cfparam name="URL.Uname" default="">
<cfparam name="URL.Pword" default="">
<cfquery datasource="global" name="Login">
Select *
From CFDJ_users
Where Username='#URL.Uname#'
AND Password ='#URL.Pword#'
</cfquery>
<cfquery datasource="global" name="Feed">
Select *
From CFDJ_RSS
Where Username='#URL.Uname#'
</cfquery>
<cfif Login.Recordcount GTE 1>
<application>
<dashboard>
<broadcast><cfoutput>Welcome To G-unix Rss Manager</cfoutput>
</broadcast>
<username><cfoutput>#URL.Uname#</cfoutput></username>
<feed>
<cfloop query="Feed">
<name>
<cfoutput>#Rss_Name#</cfoutput>
</name>

```

```

<mlink>
<cfoutput>#Rss_Link#</cfoutput>
</mlink>
</cfloop>
</feed>
</dashboard>
</application>
<cfelse>
<application>
<dashboard>
<broadcast>
<cfoutput>The Username and Or Password Is Wrong Please Try Again</
cfoutput>
</broadcast>
</dashboard>
</application>
</cfif>

```

Listing 3

```

ViewRSS.cfm
<cfcontent type="application/xml">
<cfparam name="URL.RSS" default="">
<cfhttp url="#URL.RSS#" method="get">
</cfhttp>
<cfset feed = xmlParse(cfhttp.filecontent) />
<cfoutput>#feed#</cfoutput>

```

Listing 4

```

WriterSS.cfm
<cfcontent type="application/xml">
<cfparam name="URL.Uname" default="">
<cfparam name="URL.Rlink" default="">
<cfquery datasource="global" name="DeleteRSS">
Delete
FROM CFDJ_RSS
Where Username='#URL.Uname#'
AND Rss_Link='#URL.Rlink#'
</cfquery>

```

Listing 5

```

Main.mxml
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute">
<mx:HTTPService id="Login" url="http://localhost:8500/httperv/Login.
cfm?Name={Uname.text}&Pword={Pword.text}" useProxy="false">

</mx:HTTPService>
<mx:HTTPService id="Read" url="http://localhost:8500/httperv/ViewRSS.
cfm?feed={dg.selectedItem.rlink}" useProxy="false">

</mx:HTTPService>

```




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:

IT Solutions Guide

MX Developer's Journal

Information Storage+Security Journal

ColdFusion Developer's Journal

JDJ

XML Journal

Web Services Journal

Wireless Business & Technology

.NET Developer's Journal

Symbian Developer's Journal

LinuxWorld Magazine

WebSphere Journal

Linux Business News

WLDJ

Eclipse Developer's Journal

PowerBuilder Developer's Journal



The World's Leading i-Technology Publisher


```

<mx:HTTPService id="Save" url="http://localhost:8500/httperv/WriteRSS.
cfm?Uname={Uname.text}&name={Fname.text}&rlink={Furl.text}"
useProxy="false">

</mx:HTTPService>
<mx:states>
    <mx:State name="feed">
        <mx:SetProperty target="{Pword}" name="x" value="88"/>
        <mx:SetProperty target="{Pword}" name="y" value="623"/>
        <mx:SetProperty target="{Uname}" name="x" value="88"/>
        <mx:SetProperty target="{Uname}" name="y" value="593"/>
        <mx:SetProperty target="{label1}" name="x" value="10"/>
        <mx:SetProperty target="{label1}" name="y" value="595"/>
        <mx:SetProperty target="{label2}" name="x" value="10"/>
        <mx:SetProperty target="{label2}" name="y" value="625"/>
        <mx:AddChild position="lastChild">
            <mx:DataGrid x="0" y="0" id="dg" dataProvider="{Login.las-
tResult.application.dashboard.feed}" click="Read.send()" height="585">
                <mx:columns>
                    <mx:DataGridColumn headerText="Feed Name"
dataField="name"/>
                    <mx:DataGridColumn headerText="Feed Link"
dataField="rlink"/>
                </mx:columns>
            </mx:DataGrid>
        </mx:AddChild>
        <mx:AddChild position="lastChild">
            <mx:DataGrid x="210" y="0" width="406" height="314"
dataProvider="{Read.lastResult.rss.channel.item}" id="dg2">
                <mx:columns>
                    <mx:DataGridColumn headerText="Column 1"
dataField="title"/>
                    <mx:DataGridColumn headerText="Column 2"
dataField="link"/>
                    <mx:DataGridColumn headerText="Column 3"
dataField="pubDate"/>
                </mx:columns>
            </mx:DataGrid>
        </mx:AddChild>
        <mx:AddChild position="lastChild">
            <mx:TextArea x="210" y="322" width="406" height="203"
text="{dg2.selectedItem.description}"/>
        </mx:AddChild>
        <mx:AddChild position="lastChild">
            <mx:ApplicationControlBar x="210" y="533" width="406">
                <mx:LinkButton label="Add Feed" click="currentState='a
dd'"/>
            </mx:ApplicationControlBar>
        </mx:AddChild>
        <mx:SetProperty target="{button1}" name="x" value="256"/>
        <mx:SetProperty target="{button1}" name="y" value="623"/>
        <mx:SetProperty target="{button1}" name="label"
value="Back"/>
    </mx:State>
    <mx:State name="add">
        <mx:SetProperty target="{Pword}" name="x" value="78"/>
        <mx:SetProperty target="{Pword}" name="y" value="593"/>
        <mx:SetProperty target="{label2}" name="x" value="0"/>
        <mx:SetProperty target="{label2}" name="y" value="595"/>
        <mx:SetProperty target="{button1}" name="x" value="182"/>
        <mx:SetProperty target="{button1}" name="y" value="623"/>
        <mx:SetProperty target="{Uname}" name="x" value="78"/>
        <mx:SetProperty target="{Uname}" name="y" value="563"/>
        <mx:SetProperty target="{label1}" name="x" value="0"/>
        <mx:SetProperty target="{label1}" name="y" value="565"/>
        <mx:SetProperty target="{button1}" name="label"
value="Back"/>
        <mx:AddChild position="lastChild">
            <mx:Panel x="142.5" y="75" width="341" height="246"
layout="absolute" title="Add A Feed">
                <mx:TextInput x="115.5" y="40" id="Fname"/>
                <mx:TextInput x="115.5" y="91" id="Furl"/>
                <mx:Button x="210.5" y="132" label="save" click="Save.
send();Login.send();currentState='feed'"/>
                <mx:Label x="39.5" y="42" text="Feed Name"/>
                <mx:Label x="45.5" y="93" text="Feed URL"/>
            </mx:Panel>
        </mx:AddChild>
    </mx:State>
</mx:states>
<mx:TextInput x="233" y="214" id="Uname"/>
<mx:TextInput x="233" y="244" id="Pword"/>
<mx:Label x="155" y="216" text="Username" id="label1"/>
<mx:Label x="155" y="246" text="Password" id="label2"/>
<mx:Button x="328" y="274" label="Login" click="Login.send();current
State='feed'" id="button1"/>
</mx:Application>

```

```

        <mx:SetProperty target="{Uname}" name="editable"
value="false"/>
        <mx:SetProperty target="{Pword}" name="editable"
value="false"/>
        <mx:SetProperty target="{Uname}" name="enabled"
value="false"/>
        <mx:SetProperty target="{Pword}" name="enabled"
value="false"/>
        <mx:SetEventHandler target="{button1}" name="click" handler="
currentState=' '/>
    </mx:State>
    <mx:State name="add">
        <mx:SetProperty target="{Pword}" name="x" value="78"/>
        <mx:SetProperty target="{Pword}" name="y" value="593"/>
        <mx:SetProperty target="{label2}" name="x" value="0"/>
        <mx:SetProperty target="{label2}" name="y" value="595"/>
        <mx:SetProperty target="{button1}" name="x" value="182"/>
        <mx:SetProperty target="{button1}" name="y" value="623"/>
        <mx:SetProperty target="{Uname}" name="x" value="78"/>
        <mx:SetProperty target="{Uname}" name="y" value="563"/>
        <mx:SetProperty target="{label1}" name="x" value="0"/>
        <mx:SetProperty target="{label1}" name="y" value="565"/>
        <mx:SetProperty target="{button1}" name="label"
value="Back"/>
        <mx:AddChild position="lastChild">
            <mx:Panel x="142.5" y="75" width="341" height="246"
layout="absolute" title="Add A Feed">
                <mx:TextInput x="115.5" y="40" id="Fname"/>
                <mx:TextInput x="115.5" y="91" id="Furl"/>
                <mx:Button x="210.5" y="132" label="save" click="Save.
send();Login.send();currentState='feed'"/>
                <mx:Label x="39.5" y="42" text="Feed Name"/>
                <mx:Label x="45.5" y="93" text="Feed URL"/>
            </mx:Panel>
        </mx:AddChild>
    </mx:State>
</mx:states>
<mx:TextInput x="233" y="214" id="Uname"/>
<mx:TextInput x="233" y="244" id="Pword"/>
<mx:Label x="155" y="216" text="Username" id="label1"/>
<mx:Label x="155" y="246" text="Password" id="label2"/>
<mx:Button x="328" y="274" label="Login" click="Login.send();current
State='feed'" id="button1"/>
</mx:Application>

```

Download the Code...
Go to <http://coldfusion.sys-con.com>

Welcome to the Future of Video on the Web!

REGISTER NOW!
www.iTVcon.com

CALL FOR PAPERS NOW OPEN!

 **LIVE SIMULCAST!**
AROUND THE WORLD ON SYS-CON.TV

iTVCON.COM
INTERNET TV CONFERENCE & EXPO 2006

Coming in 2006 to New York City!

“Internet TV is wide open, it is global, and in true ‘Web 2.0’ spirit it is a direct-to-consumer opportunity!”



For More Information, Call 201-802-3023
or Email itvcon@sys-con.com

Welcome to the Future!

Did you already purchase your “.tv” domain name?

You can't afford not to add Internet TV to your Website in 2006!

2005 was the year of streaming video and the birth of **Internet TV**, the long-awaited convergence of television and the Internet. Now that broadband is available to more than 100 million households worldwide, every corporate Website and every media company must now provide video content to remain competitive, not to mention live and interactive video Webinars and on-demand Webcasts.

20 years ago the advent of desktop publishing tools opened the doors for the creation of some of today's well-known traditional print media companies as well as revolutionized corporate print communications. Today, with maturing digital video production, the advent of fully featured PVRs, and significant advances in streaming video technologies, **Internet TV** is here to stay and grow and will be a critical part of every Website and every business in the years to come.

It will also very rapidly become a huge challenge to network and cable television stations: **Internet TV** is about to change forever the \$300BN television industry, too.

The Internet killed most of print media (even though many publishers don't realize it yet), Google killed traditional advertising models, and **Internet TV** will revolutionize television the way we watch it today. You need to be part of this change!

Jeremy Geelan
Conference Chair, iTVCon.com
jeremy@sys-con.com

PRODUCED BY
SYS-CON
EVENTS

List of Topics:

- > Advertising Models for Video-on-demand (VOD)
- > Internet TV Commercials
- > Mastering Adobe Flash Video
- > How to Harness Open Media Formats (DVB, etc)
- > Multicasting
- > Extending Internet TV to Windows CE-based Devices
- > Live Polling During Webcasts
- > Video Press Releases
- > Pay-Per-View
- > Screencasting
- > Video Search & Search Optimization
- > Syndication of Video Assets
- > V-Blogs & Videoblogging
- > Choosing Your PVR
- > Product Placement in Video Content
- > UK Perspective: BBC's "Dirac Project"
- > Case Study: SuperSun, Hong Kong

- | | |
|----------------|--|
| Track 1 | Corporate marketing, advertising, product and brand managers |
| Track 2 | Software programmers, developers, Website owners and operators |
| Track 3 | Advertising agencies, advertisers and video content producers |
| Track 4 | Print and online content providers, representatives from traditional media companies, print and online magazine and newspaper publishers, network and cable television business managers |



Playing with Arrays

A powerful tool in the hands of a skilled coder



By Jeff Peters

When I found out this month's issue would be a "back to basics" issue, I was torn between several topics that I hope are of interest to every CFML developer. I settled on the array, which

is a powerful tool in the hands of a skilled coder. Just to make sure everyone's on board we'll start with the assumption that we need to explore the nature of an array first.

Every programming language has simple datatypes and complex datatypes. Simple datatypes are things like strings and numbers – datatypes that store a single value. Complex datatypes are things like arrays and structures (which are also called associative arrays) – datatypes that can store more than one simple value. An array is a complex datatype that stores values in a series of elements. Each element is addressed by a number known as its index.

That's the programming definition of an array. To get a handle on the concept, I find it easiest to think of an array as a container with several pockets like an egg carton. To put something in the carton, we have to say something like "Put an egg in pocket one of the egg carton." In CFML that statement looks like this:

```
<cfset eggCarton[1] = "egg">
```

or if we're working inside a <cfscript> block:


```
eggCarton[1] = "egg";
```

Similarly, if we want to find out what's in the fifth pocket of the egg carton, we could output it to the browser like this:

```
<cfoutput>#eggCarton[5]#</cfoutput>
```

The egg carton example is an array of one dimension. That means it only has one index number – the number of the pocket we need. But we can also have more complex situations. For example, we could have a set of egg cartons. In that case we have to specify not only which pocket we need, but which carton as well. To do this, we use a second index number. So the sixth pocket of the second egg carton could be referenced as eggCartons[2][6] (the plural eggCartons is arbitrary; it just helps to remember the purpose of this particular array). This new array has two dimensions indicated by the two index numbers used. We can think of the first dimension as “carton” and the second dimension as “pocket.”

This brings us around to the syntax for creating an array in ColdFusion. Before we can put anything in the array, we have to create a variable with the array datatype. The ArrayNew() function handles the job, and it takes one argument: the number of dimensions we want in the array. CFML will accept a value of 1, 2, or 3 for the number of dimensions. Here are the statements we would use to create the arrays used in the two egg carton examples we've seen so far:

```
<cfset eggCarton = ArrayNew(1)>
<cfset eggCartons = ArrayNew(2)>
```

The ArrayNew() function creates an array, but it doesn't store any values in the array's elements. That job remains for us to do.

Even though ColdFusion only allows us to create up to three dimensions with the ArrayNew() function, the number of dimensions isn't really restricted to three. Each extra dimension is really another layer of arrays, so it's possible to manually construct arrays of more than three dimensions. That's a discussion for another article, but here's a bit of code to give you a hint; we could create the eggCartons array this way (assuming we want four cartons):

```
<cfset eggCartons = ArrayNew(1)>
<cfset ArraySet(eggCartons,1,4,ArrayNew(1))>
```

The second line in this example is equivalent to the following loop:

```
<cfloop from="1" to="4" index="i">
    <cfset eggCartons[i] = ArrayNew(1)>
</cfloop>
```

Based on this example, we can extrapolate the same technique to build an array of as many dimensions as desired. In practical use, though, arrays of more than three dimensions are generally reserved for esoteric uses.

In some programming languages, an array's elements must all be the same datatype such as integer or alphanumeric. This is not the case where ColdFusion is concerned. We can store any

datatype in any element of an array, even mixing datatypes in the array. We can have arrays of structures or arrays of arrays (as seen in the code above), or arrays of CFCs. Again, that's a little more than we have room to cover in this article.

However, we do want to do something useful with arrays and take advantage of a few of the array functions in CFML, so let's get to some more practical code. Card games seem to be increasingly popular these days, especially poker, so let's say we want to cook up a way to store a deck of cards, shuffle them, and deal out hands.

The first thing we want to do is create, in code, a model of a deck of cards. Listing 1 shows one way to approach the job with CardsArray.cfm. A quick note on the listings in this article: they all use a Fusedoc at the top of the file to describe the expectations for the code. Even though this code isn't being written for a Fusebox application, the Fusedocs help communicate our intentions in designing the code.

So the responsibilities for CardsArray.cfm are to create an array then populate it with 52 members representing the cards of a standard deck. Line 20 sets up the array then we start the code to get it populated.

Line 21 sets a variable with a list of the suits we want in the deck. Lines 24-34 contain code, including a nested loop, to store card values in the deck array. The outer loop uses the list of suits and the variable aSuit to hold the value of each element of the list as it loops through. For each value in the list of suits, the inner loop stores the number cards (2-10).

The inner loop uses the variable i to store the value of its index. This is common practice when using counting loops – i is an abbreviation for index. When counting loops are nested, it's common practice to use sequential letters starting with i for the indices. So two nested counting loops are often seen using the indices i and j.

deckArray - array			
1	2S	14	2D
2	3S	15	3D
3	4S	16	4D
4	5S	17	5D
5	6S	18	6D
6	7S	19	7D
7	8S	20	8D
8	9S	21	9D
9	10S	22	10D
10	JS	23	JD
11	QS	24	QD
12	KS	25	KD
13	AS	26	AD
27	2C	39	AC
28	3C	40	2H
29	4C	41	3H
30	5C	42	4H
31	6C	43	5H
32	7C	44	6H
33	8C	45	7H
34	9C	46	8H
35	10C	47	9H
36	JC	48	10H
37	QC	49	JH
38	KC	50	QH
		51	KH
		52	AH

Figure 1: The deck

The `ArrayAppend()` function is used to add a new element to the `deckArray`. `ArrayAppend()` is very useful in that we don't need to know the size of the array to add an element – we just specify the array's name and the value to append.

Once the inner loop has run, we use four more `ArrayAppend()` statements to add the face cards and aces for the suit. That's it – the `deckArray` array has been created and populated. However, there's no code in `CardsArray.cfm` to display the array, so how do we know it worked properly? Since we might want to reuse this routine, we don't want to display its output here. So we'll use a test harness template.

The test harness is named `testCardsArray.cfm`, and is shown in Listing 2. All it does is use `<cfinclude>` for the `CardsArray.cfm` template then `<cfdump>` to show the `deckArray` variable. The output is shown in Figure 1 (the `cfdump` output has been broken up to save print space).

Now that we have a deck of cards, we're ready to start playing. So we sit down at the card table and I pick up the deck and start

shuffledDeckArray - array			
1	7S	14	4H
2	KC	15	3S
3	KS	16	5H
4	8H	17	JD
5	3H	18	10H
6	9C	19	6D
7	8S	20	3D
8	5S	21	6C
9	QD	22	AH
10	2H	23	6H
11	QS	24	10C
12	9S	25	JC
13	9D	26	QC
		27	5D
		28	JS
		29	AS
		30	2C
		31	KD
		32	KH
		33	AD
		34	QH
		35	8C
		36	4D
		37	5C
		38	7D
		39	AC
		40	8D
		41	JH
		42	2S
		43	6S
		44	4C
		45	7H
		46	3C
		47	2D
		48	4S
		49	9H
		50	10S
		51	7C
		52	10D

Figure 2: shuffled deck

handsArray - array			
1	7H	4	5D
2	AH	5	AS
3	9H	6	8S
4	9S	7	AD
5	KS	8	5S
		9	8H
		10	5H
		11	6S
		12	2H
		13	10C
		14	4H
		15	3S
		16	3C
		17	9D
		18	QH
		19	8C
		20	10S
		21	7C
		22	7S
		23	3D
		24	2D
		25	AC
		26	9C
		27	JC
		28	6C
		29	5C
		30	KH
		31	KD
		32	JD
		33	4S
		34	10H
		35	6H
		36	8D
		37	3H
		38	QC

Figure 3: Dealt hands and remaining deck

dealing. What's that you say? You want me to shuffle the cards? Oh, all right, if you insist. Some people are such sticklers for details.

Listing 3 shows `ShuffleDeck.cfm`. Its job is to take `deckArray` as created by `CardsArray.cfm` and produce a shuffled copy of it named `shuffledDeckArray`. Line 24 creates the copy then lines 27-31 take care of the shuffle.

There have been many articles written over the years on the subject of sorting and shuffling. There are probably as many shuffle algorithms available in the programming world as there are opinions on any given topic. Fortunately for us, there's a function in CFML that makes shuffling easy. That function is `ArraySwap()`. As its name implies, it swaps two specified elements of an array. So to do our shuffle, we just pick two random indexes between 1 and 52 using the `RandRange()` function and then swap those two elements in the deck using the `ArraySwap()` function.

As with the `CardsArray.cfm` file, we don't produce any output with `ShuffleDeck.cfm`. We test it with another test harness called `testShuffleDeck.cfm`. It's shown in Listing 4 and just includes `CardsArray.cfm` and `ShuffleDeck.cfm`, and then uses `<cfdump>` to show the `shuffledDeckArray` variable after the shuffling process. Its output is shown in Figure 2.

Okay, now that the deck is shuffled, may I deal the cards? Thank you; I'm glad this is such a friendly table. Listing 5 shows `DealCards.cfm`. It creates a two-dimensioned array named `handsArray`. The two dimensions represent hands and cards in each hand.

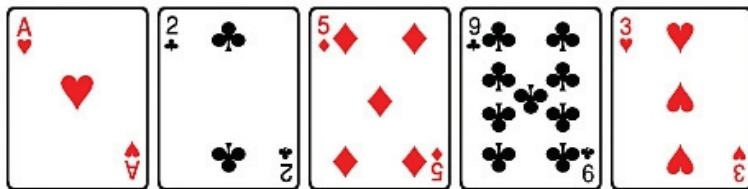
To deal, we use another nested loop (lines 34-40). For each hand, we create a new element in `handsArray` and initialize it as an empty array. Then for the number of cards per hand we just append the first element from the `shuffledDeckArray` to the hand and delete that first element from the `shuffledDeckArray`. The result of this process is the `handsArray` with the specified number of hands and cards per hand, and the `shuffledDeckArray`, which now holds the remainder of the deck. We use another test harness to set the parameters for the hands and cards per hand, run `DealCards.cfm`, and display the resulting arrays (Listing 6). Figure 3 shows those arrays.

So the deck has been shuffled and the cards have been dealt. But so far all we have to show for it are these lousy `<cfdump>` outputs. Not a very attractive game, is it? So now we want to build a way to make the dealt hands look like dealt hands. For that, we can use a set of images – one for each card, with the names matching the notation we used for cards in the arrays. Then it's just a matter of writing a nested loop to output the `handsArray`. That code from `ShowHands.cfm` is shown in Listing 7. It uses two nested counting loops, and their indices are `i` and `j` like the common practice we looked at earlier.

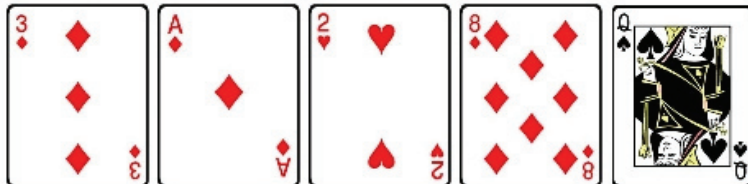
Since we need all the previous non-test code to make `ShowHands.cfm` work, we once again use a test harness as shown in Listing 8. It just includes the necessary files and doesn't set values for the number of hands or the number of cards per hand. We're just allowing `ShowHands.cfm` to use its default values for those variables. The final output is shown in Figure 4.

Arrays provide a powerful way to handle data that has a common purpose, like the cards in a deck; and whose elements aren't identified by name or idea. But until we've used them a few times, they aren't really a natural concept. So how do we decide an array would be a good idea? If you find yourself tempted to create variables with names like `item1`, `item2`, `item3`,

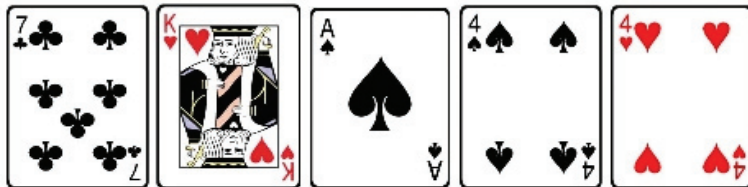
Hand 1:



Hand 2:



Hand 3:




Hand 4:



Figure 4: Dealt hands graphic

etc., it's probably a good case for an array. On the other hand, if there seems to be related sets of variables – like `firstName1`, `firstName2` and `lastName1`, `lastName2` – it's probably a good case for a structure. Fortunately, you can find information on structures in this issue as well.

CFML also offers several array functions we haven't seen in this article, including `ArrayIsEmpty`, `ArrayPrepend`, `ArrayAvg`, `ArrayResize`, `ArrayToList`, `ArrayClear`, `ArrayMax`, `isArray`, `ArrayMin`, `ArraySort`, `ListToArray`, `ArrayInsertAt`, and `ArraySum`. The names of these functions provide a pretty good indication of their purposes, so feel free to experiment with them on your own. I highly recommend the use of a test directory on your development box. You can create little CFML templates to test your ideas and play with code before you try it in a "real" environment.

If you're interested in a more complete discussion of complex datatypes in ColdFusion, you might be interested in my book *ColdFusion Lists, Arrays & Structures*. It can be ordered online at www.ProtonArts.com. 

About the Author

Jeff Peters is a program manager and application architect for Operational Technologies Services in Vienna, Virginia. His ColdFusion-related books are available at www.protonarts.com. Listen to Jeff with Hal Helms on the Helms and Peters Out Loud podcast at www.helmsandpeters.com.

jeff@grokfusebox.com

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 *beta*

— This site will go beta February 15, 2003!

Listing 1 - CardsArray.cfm

```

<!---
<fusedoc fuse="CardsArray.cfm" language="CFML" version="2.0">
  <responsibilities>
    I create an array with an element for each card in a standard
    playing card deck.
  </responsibilities>
  <properties>
    <history date="2006-08-05" author="jeff@grokfusebox.com"
    role="Architect" type="Create" />
  </properties>
  <io>
    <in>
    </in>
    <out>
      <array name="deckArray">
        <string comments="a card value and suit" />
      </array>
    </out>
  </io>
</fusedoc>
--->
<cfset deckArray = ArrayNew(1)>
<cfset suitList = "S,D,C,H">

<!--- Populate the deck array --->
<cfloop list="#suitList#" index="aSuit">
  <!--- Now add the numbered cards for this suit --->
  <cfloop from="1" to="9" index="i">
    <cfset ArrayAppend(deckArray,"#i1##aSuit#")>
  </cfloop>
  <!--- Now add the 'special' cards for this suit --->
  <cfset ArrayAppend(deckArray,"J#aSuit#")>
  <cfset ArrayAppend(deckArray,"Q#aSuit#")>
  <cfset ArrayAppend(deckArray,"K#aSuit#")>
  <cfset ArrayAppend(deckArray,"A#aSuit#")>
</cfloop>

```

Listing 2 - testCardsArray.cfm

```

<cfinclude template="CardsArray.cfm">

<cfdump var="#deckArray#" label="deckArray">

```

Listing 3 - ShuffleDeck.cfm

```

<!---
<fusedoc fuse="ShuffleDeck.cfm" language="CFML" version="2.0">
  <responsibilities>
    I use a deck array to create a shuffled deck array.
  </responsibilities>
  <properties>
    <history date="2006-08-05" author="jeff@grokfusebox.com"
    role="Architect" type="Create" />
  </properties>
  <io>
    <in>
      <array name="deckArray">
        <string comments="a card value and suit" />
      </array>
    </in>
    <out>
      <array name="shuffledDeckArray">
        <string comments="a card value and suit" />
      </array>
    </out>
  </io>
</fusedoc>
--->

<cfset shuffledDeckArray = deckArray>

<!--- Shuffle the deck by swapping random pairs of cards; higher 'to'
value means more shuffling --->
<cfloop from="1" to="100" index="i">
  <cfset Card1 = RandRange(1,52)>
  <cfset Card2 = RandRange(1,52)>
  <cfset ArraySwap(shuffledDeckArray,Card1,Card2)>
</cfloop>

```

Listing 4 - testShuffleDeck.cfm

```

<cfinclude template="CardsArray.cfm">
<cfinclude template="ShuffleDeck.cfm">

<cfdump var="#shuffledDeckArray#" label="shuffledDeckArray">

```

Listing 5 - DealCards.cfm

```

<!---
<fusedoc fuse="DealCards.cfm" language="CFML" version="2.0">
  <responsibilities>

```

```

    I deal a specified number of hands, each hand consisting of a
    specified number of cards.
</responsibilities>
<properties>
    <history date="2006-08-05" author="jeff@grokfusebox.com"
    role="Architect" type="Create" />
</properties>
<io>
    <in>
        <array name="shuffledDeckArray">
            <string comments="a card value and suit" />
        </array>
        <number name="numberOfHands" precision="integer" optional="yes"
        default="4"/>
        <number name="cardsPerHand" precision="integer" optional="yes"
        default="5" />
    </in>
    <out>
        <array name="handsArray">
            <array comments="an array for each hand">
                <string comments="a card value and suit" />
            </array>
        </array>
    </out>
</io>
</fusedoc>
--->
<cfparam name="cardsPerHand" default="5">
<cfparam name="numberOfHands" default="4">

<!-- Make an array to hold the dealt hands -->
<cfset handsArray = ArrayNew(1)>

<!-- Deal the deck into hands of the specified number of cards -->
<cfloop from="1" to="#numberOfHands#" index="i">
    <cfset handsArray[i] = ArrayNew(1)>
    <cfloop from="1" to="#cardsPerHand#" index="j">
        <cfset handsArray[i][j] = shuffledDeckArray[1]>
        <cfset ArrayDeleteAt(shuffledDeckArray,1)>
    </cfloop>
</cfloop>

```

Listing 6 - testDealCards.cfm

```

<cfinclude template="CardsArray.cfm">
<cfinclude template="ShuffleDeck.cfm">
<cfset numberOfHands = 6>
<cfset cardsPerHand = 5>

```

```

<cfinclude template="DealCards.cfm">

<cfdump var="#handsArray#" label="handsArray">
<cfdump var="#shuffledDeckArray#" label="shuffledDeckArray">

```

Listing 7 - ShowHands.cfm

```

<!--
<fusedoc fuse="ShowHands.cfm" language="CFML" version="2.0">
    <responsibilities>
        I use image files to display an array of dealt hands.
    </responsibilities>
    <properties>
        <history author="jeff@grokfusebox.com" role="Architect"
        type="Create" />
    </properties>
    <io>
        <in>
            <array name="handsArray">
                <array comments="an array for each hand">
                    <string comments="a card value and suit" />
                </array>
            </array>
        </in>
        <out>
            <out>
        </out>
    </io>
</fusedoc>
--->

<cfloop from="1" to="#ArrayLen(handsArray)#" index="i">
    <cfoutput><h3>Hand #i:</h3></cfoutput>
    <cfloop from="1" to="#ArrayLen(handsArray[i])#" index="j">
        <cfoutput></cfoutput>
    </cfloop>
    <br />
</cfloop>

```

Listing 8 - testShowHands.cfm

```

<cfinclude template="CardsArray.cfm">
<cfinclude template="ShuffleDeck.cfm">
<cfinclude template="DealCards.cfm">
<cfinclude template="ShowHands.cfm">

```

Download the Code...
Go to <http://coldfusion.sys-con.com>

Power Your Productivity

Leveraging cfcPowerTools

By Tom Schreck,
Byron Bignell &
Nolan Dubeau

One of the biggest problems developers face everyday is moving a project along and producing solid code without cutting corners. This typically means a lot of copying, pasting, searching and replacing, and a whole lot of manual editing. At every point the potential for error increases. Confidence in your code diminishes and your day gets longer.

Now, to add to the joy, consider this: change.

Change is inevitable and becomes another item for developers to manage throughout the development lifecycle. Change comes in the form of scope creep, modified business requirements, database schema changes, functionality enhancements, etc. Now change can be good, however at the end of the day these changes can have a big impact on the cost, timelines, and ultimately the success of your project.

A project, like a house, requires a strong foundation; the stronger the foundation, the more confidence you have that the structure will withstand the pressures put on it. The same applies to a project. The stronger the foundation code, the more reliable the rest of the application built on it will be.

So, how do we as developers deal with change efficiently and effectively? It's not uncommon for code to be scrapped as changes are incorporated. You spend a great deal of time developing component models to abstract yourself and your project from change, but at some point, when the database schema changes you find yourself editing that code again and you'll ask yourself, "How many times have I had to go back through this code and make modifications to accommodate database changes?"

Introducing cfcPowerTools

The idea of creating data access layers (DAL) is pretty common for a number of other programming languages such as ASP.NET and Java. Newer is the idea of creating code via an object-relation mapping tool and while tools like Hibernate and LLBLGen, which are quickly becoming more common for Java

and .NET. But ORM tools aren't quite there yet on the ColdFusion scene.

Until now cfcPowerTools has been a productivity tool for generating consistent, predictable, and repeatable (CPR) code. It lets developers quickly and accurately generate ColdFusion components (CFCs) and supporting templates (html/xml/flash forms, lists) in a fraction of the time it would take to hand-code.

CFC Types

ColdFusion provides developers with a powerful set of tools to create components and Web Services and implement these objects in a wide range of frameworks and methodologies. While CF provides all of this flexibility and power the one thing it doesn't provide is the capability to quickly and easily generate the code you need for the approach or methodology you choose to work in. cfcPowerTools is methodology-agnostic and generates ColdFusion components (CFCs) using generation templates (called CFC Types). Better still you can create your own CFC Types, or modify the existing ones to meet your needs.

CFC Types include (see Figure 1).

- **Model Controller** is composed of two CFCs: a primary cfc that houses all property tags and a model cfc that manages interactions with the database.
- **MachII** generates CFCs supporting the MachII framework. The primary cfc for the MachII cfc type is the DAO cfc. The DAO is responsible for persisting data. The DAO CFC will retrieve data from a database and return a bean cfc. The bean contains all cfproperty tags. The bean has no knowledge of the DAO. The DAO contains all of the CRUD functionality.
- **Concrete** is designed to extend another CFC file. The con-

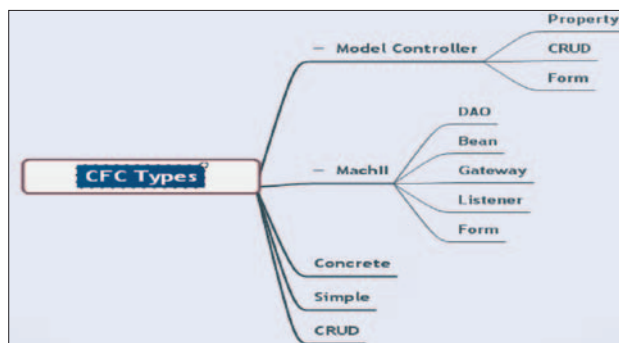


Figure 1

crete CFC Type works really well with the model controller CFC type. A good technique is to use a model controller CFC Type to generate your “abstract” CFC. This abstract CFC contains all of your property definitions plus data persistence code generated by cfcPowerTools. Use a concrete CFC Type to produce a CFC that extends the abstract. You put all of your business logic in your concrete CFC. You can now

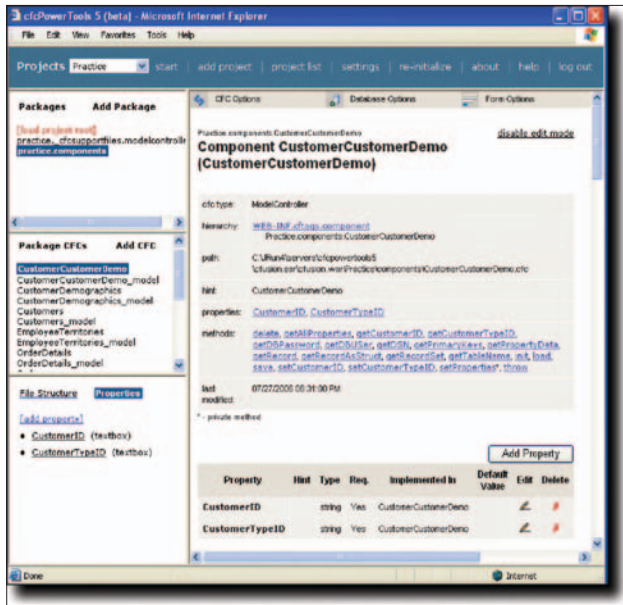


Figure 2

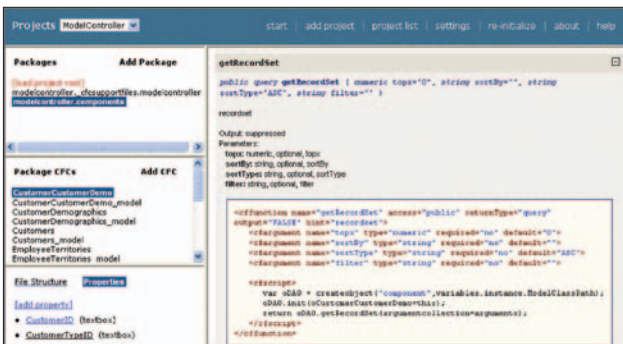


Figure 3

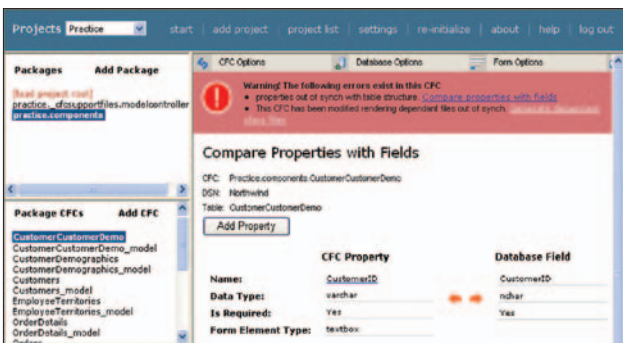


Figure 4

manage change by regenerating your abstract CFCs, keeping your business logic separate.

- **Simple** is used as a teaching tool for learning how to create a CFC Type.
- **CRUD** generates a single CFC file that contains all properties and CRUD (Create, Read, Update, and Delete) methods.

The Model Controller CFC Type generates Property and CRUD CFCs as well as data entry forms (HTML, XML, FLASH)

The MachII CFC Type generates DAO, Bean, Gateway, and Listener CFCs as well as data entry forms (HTML, XML, FLASH)

User Interface

When was the last time you took a look at the Component Doc tool in CFAdmin? Probably never, right?

The Component Doc view lets a user see the metadata about a CFC as well as the information about the properties and methods available in a component; it's a read-only tool.

cfcPowerTools utilizes Component Doc view, but makes it really useful: you can add, edit, and delete properties, add methods, as well as toggle between a method's metadata and the actual <cffunction> code (see Figure 2). By leveraging a known user interface and by adding real functionality to it that allows a developer to get the most out of their time, cfcPowerTools brings a whole new take on working with components in ColdFusion.

Code Behind

Code behind is the process of reading the actual <cffunction> code into a UI. You can toggle between the metadata view of Component Doc and the actual <cffunction> code (see Figure 3). It's real-time, so as you make changes, those changes will get pulled into the UI the next time you toggle the function.

Productivity

We all need to be productive. When was the last time you got paid for producing nothing? cfcPowerTools is focused on making you more productive. Its purpose in life is to get a lot of the grunt work code generated, to get you past writing all of the CRUD code, getters and setters, and properties. You can focus on business requirements faster. An added benefit is you start off with a solid code foundation.

Here is an example of generated Getter/Setter code:

```
<cffunction access="public" name="getCustomerID" returnType="string"
output="FALSE">
    <cfreturn variables.instance.CustomerID>
</cffunction>

<cffunction access="public" name="setCustomerID" output="FALSE">
    <cfargument name="CustomerID" type="string" required="1"/>
    <cfset variables.instance.CustomerID = arguments.CustomerID>
</cffunction>
```

Initial test results generated around 14,000 lines of code supporting 22 database tables in about five minutes. Because code is generated from templates, you can control the quality of the code produced and can rest assured that the code will be consistent, predictable, and repeatable.

```
<cffunction name="TestMethod" access="public" returntype="void" DisplayName="Test Method" Hint="test
method hint" output="false">

    <!--
    this is pseudo code. This is meant for you to enter ideas and concepts for this method so you will
    remember what it is supposed to do.
    -->

    <cfabort showerror="TestMethod under construction">

</cffunction>
```

Figure 5

Manage Change

Change happens. It can be viewed as job security or as a hassle that we have to get through. Either way, it's inevitable. So, how does cfcPowerTools mitigate change?

Wouldn't it be nice if we were notified when our code was out of synch with our database structure? cfcPowerTools analyzes and compares your CFC with its underlying table and warns you when the two are out of synch (see Figure 4). With a click of a button you can synchronize the properties with the fields or vice versa.

Having properties and fields out of synch means your CRUD statements are out of synch as well. Bad things can happen. Not to worry, since all CRUD statements are within managed code blocks, you can regenerate your CFCs and the CRUD statements will be updated to account for the synchronization.

Managed Code and Round Trip Editing

So, what are managed code blocks? Managed code blocks are "tags" that are placed inside your CFC that wrap a block of code. I use the term "tags" loosely. It's really special comments that cfcPowerTools searches for when regenerating a CFC. If the code is outside a managed code block, then it's ignored. The special comments say where to get the content that goes between the managed code comments. A pretty simple concept, but very powerful.

Managed code blocks support round-trip editing. Round-trip editing means you can interact with your CFC manually or via a tool and not lose code. The only time you lose code is when you add code within a managed code block section because everything between the tags gets regenerated. Managed code is the process of designating "sections" of your code that cfcPowerTools will reproduce. Managed code blocks can be put in a <cfscript> block or anywhere else throughout your CFC.

So what happens if something goes wrong? Auto-backups of CFCs are generated prior to applying any changes during regeneration. A history is generated of the modifications made to your CFC files.

This is a managed code pattern:

```
<!-- START MANAGEDCODE BLOCK: [CFC
PACKAGE].[CFCNAME]:[METHOD] --->
... GENERATED CODE GOES HERE...
<!-- END MANAGEDCODE BLOCK --->
```

cfcPowerTools searches your CFC



code for "START MANAGEDCODE BLOCK:" then instantiates the CFC (using the package and cfc name) listed in the pattern and calls the method that will regenerate the block of code.

Below is an example of a managed code block:

```
<!-- START MANAGEDCODE BLOCK: cfcPowerTools.library.codeSource:getManaged-
agedCodeProperties --->
... GENERATED CODE GOES HERE...
<!-- END MANAGEDCODE BLOCK --->
```

cfcPowerTools.library.codeSource CFC is instantiated and the getManagedCodeProperties method is consumed. This method regenerates the code and returns it to be included in the CFC when it gets rewritten. This method is responsible for regenerating everything inside the managed code block.

Here's an example of using managed code inside <cfscript>:

```
<cfscript>
...
//START MANAGEDCODE BLOCK: cfcPowerTools.library.codeSource:getManaged-
CodeInstanceData
... GENERATED CODE GOES HERE...
//END MANAGEDCODE BLOCK
...
</cfscript>
```

Sections of your code that deal with properties, <cfproperty> tags, and SQL statements are ideal candidates for managed code blocks. If a CFC doesn't have any managed code blocks then cfcPowerTools will only create the CFC the very first time. CFCs without managed code blocks can't be modified.

So as you can see managed code plays an important role for solving the synchronization challenge when change happens.

The Guts of cfcPowerTools

cfcPowerTools approaches CFCs and database tables by focusing on the property. Typically, properties have been used for documentation purposes. cfcPowerTools uses them as a direct correlation to fields in a database table. Properties are generated for each field in a table. In doing so, cfcPowerTools gathers a bunch of data that describes the property in terms of how the property is used in the CFC, database, and data entry form. All of this data is

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



OFFER SUBJECT TO CHANGE WITHOUT NOTICE

stored in a generated XML metadata file. The XML File is governed by a XML schema.

Data Entry Forms

Another productivity feature is form generation. cfcPowerTools use the metadata XML document that describes how properties are “seen” in a data entry form to generate the form. Raw data entry forms are generated. Remember, the purpose is to get the grunt work out of the way, not to produce production-ready code. You’re still the developer. HTML, XML, and FLASH forms are supported. Typically, you take the generated form and modify it to your needs.

Example Code

The links below are examples of code generated by cfcPowerTools. Each zip file contains CFCs and data entry forms for several tables in the Northwind database. These examples demonstrate the code generated for the Model Controller and Mach II CFC Types.

<http://www.cfcpowertools.com/downloads/ModelController.zip>

<http://www.cfcpowertools.com/downloads/MachII.zip>

Pseudo-Code

The improved Component Doc view lets you interact with your CFC in ways you couldn’t before. You can create and append CFC functions to your CFC via the new Component Doc view. Powering your productivity goes a step further by letting you enter pseudo-code when creating new functions. You use pseudo-code as a reminder of what the function is supposed to do. Pseudo-code increases your productivity by describing the function so you can fill in the actual code at a later date.

Figure 5 provides an example of a <cfunction> generated from a UI with pseudo-code notes.

Wrapping Up

cfcPowerTools is not an end-all be-all and it’s not trying to be all things to everyone. It won’t do your dishes or tie your shoes. It doesn’t produce your final code base. You still have to use your skills as a developer. It doesn’t and won’t replace your skills. It’s simply a tool that will make you more productive, generate a consistent base code, and let you focus on the work of engineering the business rules, behaviors, and presentation and not the plumbing of the application.

For more information, please visit www.cfcpowertools.com.

About the Author

Tom Schreck is a Macromedia certified ColdFusion MX 6.1 developer. He has been working with ColdFusion since 1997. Check out www.cfcPowerTools.com for more information on cfcPowerTools.

tschreck@sbcglobal.net

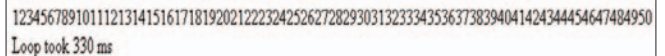
—CONTINUED FROM PAGE 10

Faster Debugging with CFTIMER and CFTRACE

Measuring Processing Time with CFTIMER

Occasionally, I’ll have to debug a very slow running template. In the past I had to rely on `getTickCount()` to find out how long it took a section of code to run.

```
<cfset start = GetTickCount()>
<cfloop from="1" to="100" index="i"><cfoutput>#i#</cfoutput></cfloop>
<cfset end = GetTickCount()>
<cfoutput>Loop took #end-start# ms</cfoutput>
```



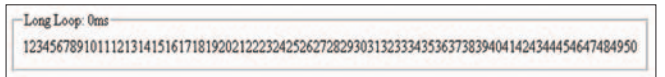
But recently I’ve been using a new tag called CFTIMER. Simply wrap a section of your code with CFTIMER and, voilà, you instantly know how long it took to run that section.

```
<CFTIMER label="Long Loop">
<cfloop from="1" to="1000" index="i"><cfoutput>#i#</cfoutput></cfloop>
</CFTIMER>
```



CFTIMER lets you put the information inline (as seen in the example above) in an HTML comment, in the debugging info, or even in a nice outline as seen in the example below.

```
<CFTIMER label="Long Loop" type="outline">
<cfloop from="1" to="100" index="i"><cfoutput>#i#</cfoutput></cfloop>
</CFTIMER>
```



Using “outline” can become very useful when you have several nested sections you want to test.

Summary

CFTRACE and CFTIMER introduce new powerful debugging capabilities. These tags offer significant advantages over the previous methods and make it both easier and faster to debug code.

About the Author

Shlomy Gantz is president of BlueBrick, Inc., a certified Adobe instructor and a member of the Adobe Community Expert program. shlomy@bluebrick.com



Beyond boundaries.

In the right environment, imagination has no limits.

MAX 2006, the annual Adobe user conference, offers a unique opportunity to learn about Adobe software, interact with industry experts, connect with the Adobe community, and explore ideas yet to be imagined.

Choose from over 100 different workshops and hands-on sessions presented by Adobe experts and industry leaders. Exchange ideas with other community members at a variety of networking events. Experience current and emerging Adobe technology.

Join us for MAX 2006 this October 23-26

in Las Vegas to learn, network, and move beyond the boundaries of what you believe is possible. Register by September 25 to qualify for a special Early Bird discount of \$200 off the standard registration fee.

www.adobe.com/go/cfdj



Beyond boundaries. The 2006 Adobe conference.

Adobe and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Other companies in this magazine spent a lot of time on pretty ads. As you can see, we did not. We spent our time hiring the best people and training them to deliver outstanding support for your website. We spent our time building a state of the art datacenter and staffing it with people who care about your website like it's their own. Compassion, respect, credibility, ownership, reliability, "never say no," and exceed expectations are words that describe our service philosophy. From the first time you interact with us, you'll see what a difference it really makes. And you'll also forgive us for not having a pretty ad.

