ColdFusion Developer's Journal

ColdFusionJournal.com
June 2004 Volume: 6 Issue: 6

RETAILERS PLEASE DISPLAY UNTIL AUGUST 31, 2004
$9.99US $9.99CAN
07
0 09281 02689 1
SYS-CON MEDIA

Flash Remoting: An Alternative Approach

By Ian Bale page 12

edgemodern.com

"Macromedia® Dreamweaver® MX 2004 has helped us design and build a fully functional site which helps us immediately respond to customer needs–all without a Ph.D. in web development."

Drew Sanocki, EDGE*MODERN, Co-founder.

gulfstream.com

"With Dreamweaver, we'll be able to take even greater advantage of CSS, which will add huge efficiencies to how we develop and maintain our website."

Will Dent, Gulfstream Aerospace, Interactive Marketing.

Dreamweaver. The highest common denominator.

Now updated and
up to 70% faster.

See more at: macromedia.com/go/dwupdated

It’s everybody’s PDF™

Finally, a software company that offers affordabe yet flexible PDF solutions to meet every customer's needs. Using activePDF™ to automate the PDF creation process eliminates the need for end-user intervention so your employees can concentrate on what they do best.

Licensed per server, activePDF solutions include a COM interface for easy integration with ColdFusion. Dynamically populate PDF forms with information from a database, convert ColdFusion web pages to PDF on the fly, dynamically print reports to PDF using CF and much more. Users can also merge, stitch, stamp, secure and linearize PDF, all at a fraction of the cost of comparable solutions. Download your free trial version today!

activePDF®
Leading the iPaper Revolution

www.activePDF.com

ColdFusion Developer's Journal

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
Robert Diamond robert@sys-con.com
technical editors
Raymond Camden raymond@sys-con.com
Simon Horwith simon@sys-con.com
executive editor
Jamie Matusow jamie@sys-con.com
editor
Nancy Valentine nancy@sys-con.com
associate editors
Gail Schultz gail@sys-con.com
Jean Cassidy jean@sys-con.com
Jennifer Van Winckel jennifer@sys-con.com
research editor
Bahadir Karuv, PhD bahadir@sys-con.com

production

production consultant
Jim Morgan jim@sys-con.com
lead designer
Abraham Addo abraham@sys-con.com
art director
Alex Botero alex@sys-con.com
associate art directors
Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Tami Beatty tami@sys-con.com

contributors to this issue

Charlie Arehart, Ian Bale, Steve Drucker, Ben Forta, Hal Helms, Simon Horwith, Jeffry Houser, Craig Rosenblum

editorial offices

SYS-CON MEDIA
135 CHESTNUT RIDGE RD., MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 **FAX:** 201 782-9638
COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
is published monthly (12 times a year)
by **SYS-CON Publications, Inc.**
postmaster: send address changes to:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645

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

ColdFusion in 30 Minutes or Less

By Robert Diamond

Tim Buntel, the ColdFusion Product Manager for Macromedia, started a new blog about Blackstone (the next release of ColdFusion) that's well worth reading and bookmarking over at www.buntel.com/blog/.

In it, he makes reference to something that we've talked about here and elsewhere before, but puts it so brilliantly, talking about the goals for Blackstone, that it's worth mentioning again: "If we can get someone to install the product and, within 30 minutes at their desk, create a basic dynamic Web page, we will win them over to CF."

That's certainly what attracted me initially to the world of ColdFusion a few years back, as well as the majority of developers that I've spoken to. Now, granted that those first few applications that I, and many others, made were absolutely awful, the point was that it was done fast, and easily, and without a wealth of knowledge. The basics of ColdFusion represent one of the least steep learning curves out there in the world of Web development, and I'd naturally put it at the top of the list.

That's a point that I hope Macromedia will push, and push heavily because it's the sticking point that should be helping to drive new developers over to the world of CF in droves. With today's economic and corporate realities, cutting down project time is essential, and there are few faster ways to get up and running than with our dear CFML.

Back to Tim for a moment. His blog is already interesting, and I'm sure it will continue to grow as time goes on.

We'll be discussing Blackstone and lots more at the *CFDJ* author's panel at CFUN 2004 this June, and we'll be providing full coverage in an upcoming issue. I hear the numbers for this year are stronger than ever, which is another great sign for our community.

2004 Readers' Choice Awards

Speaking of communities, I am proud to announce that nominations are now open for our 2004 Readers' Choice Awards program at www.sys-con.com/mx/readerschoice2004. As we wanted to include the rest of the Macromedia product line, and didn't want to leave ColdFusion out of the ***MXDJ*** Readers' Choice Awards, we've started a brand-new awards program combining categories from both magazines into one mega-awards program.

Nominations are now open, and voting will start shortly in what I'm predicting will be our biggest Readers' Choice Awards program yet with the combined support of both leading publications. Categories include all the usual ones, such as: Best Book, Best CF Web Service, Best Consulting Service, Best Content Management Tool, Best Custom Tag or Component, Best E-Business Software, Best Education and Training, Best Web Application, Best Web Development Tool, Best Web Hosting, Best Web Site or Community, and Most Innovative Application. There'll be some new ones too, so check it out and add your favorite products and services if you haven't already done so.

About the Author

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

SYS-CON MEDIA

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

vp, business development
Grisha Davida grisha@sys-con.com
group publisher
Jeremy Geelan jeremy@sys-con.com

advertising
senior vp, sales & marketing
Carmen Gonzalez carmen@sys-con.com
vp, sales & marketing
Miles Silverman miles@sys-con.com
advertising director
Robyn Forma robyn@sys-con.com
director of sales & marketing
Megan Mussa megan@sys-con.com
associate sales managers
Kristin Kuhnle kristin@sys-con.com
Beth Jones beth@sys-con.com
Dorothy Gil dorothy@sys-con.com

sys-con events
president, events
Grisha Davida grisha@sys-con.com
conference manager
Lin Goetz lin@sys-con.com

customer relations
circulation service coordinators
Shelia Dickerson shelia@sys-con.com
Edna Earle Russell edna@sys-con.com
Linda Lipton linda@sys-con.com

manager, jdj store
Brunilda Staropoli bruni@sys-con.com

sys-con.com
vp, information systems
Robert Diamond robert@sys-con.com

web designers
Stephen Kilmurray stephen@sys-con.com
Matthew Pollotta matthew@sys-con.com

online editor
Lin Goetz lin@sys-con.com

accounting
financial analyst
Joan LaRose joan@sys-con.com

accounts payable
Betty White betty@sys-con.com

subscriptions
Subscribe@sys-con.com
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

Tales from the List

Protecting your resources

This month's *Tales from the List* is all about protecting resources. Specifically, it is about protecting image files.

By Simon Horwith

Evik James, a *CFDJ-List* regular, asked the List for advice on how to protect from "hot-linking." Apparently, one of his employer's competitors has been linking to images on his server from pages on their own. They are not "stealing" the images since the images still reside solely on Evik's server(s), but they are using them for their own gain in their own Web applications.

As much as I love the idea of sharing resources (I believe that open source is the best thing that's ever happened to the software industry), I also recognize that it is not unreasonable for a company to not want their competitors to benefit from the images, bandwidth, and other resources that they've spent good money on. The thread prompted dozens of responses and a variety of suggestions.

Before examining the responses, I should clarify that in his original post, Evik stated that he was under the impression that he could bury the images in a directory that isn't publicly Web accessible and then deliver the images using CFCONTENT. He also noted that he's running ColdFusion 5 with an IIS 5 Web server.

Che Vilnonis immediately responded with an alternative – why not simply watermark all of the images so that they give a "plug" to Evik's site? This is a creative idea in that it would turn the negative (a competitor stealing resources) into a positive (now the competitor would be giving a plug to Evik with every image they linked to). Of course, if Evik doesn't want his images watermarked and/or doesn't have the time or resources to watermark all of his images, then this is not an option.

Mosh Teitelbaum noted that using CFCONTENT to deliver every image would add considerable strain to the application server and added a third option: moving the files. The idea here is to move the images to another location, store that location path in a variable that all of the pages use to pull images from, and then change the location of the images and the value of the variable once per week or however often Evik chooses to. The whole process could even be performed with a little ColdFusion code scheduled to run in the ColdFusion Administrator.

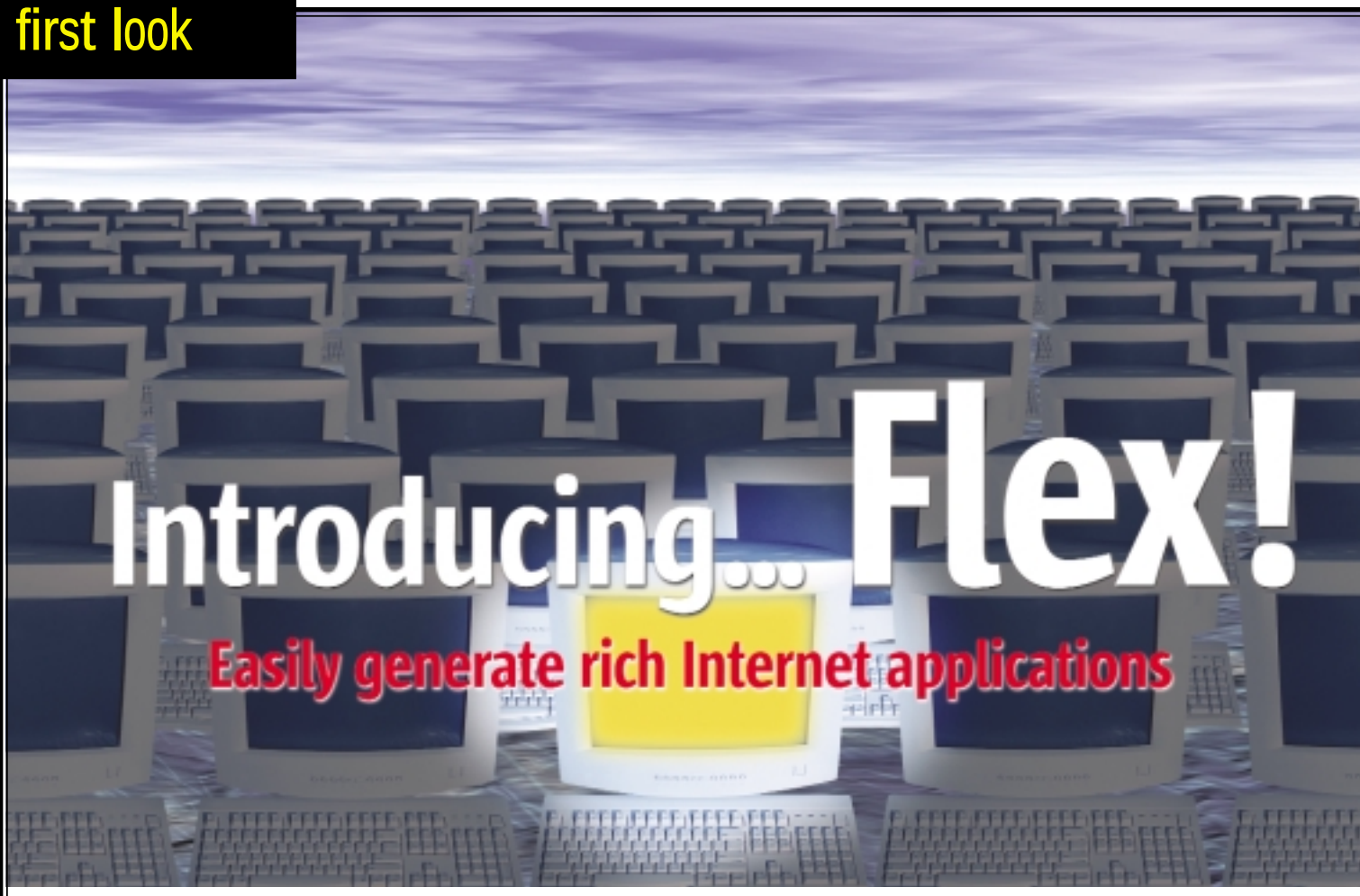
Again, not a bad idea, though the process could put a lot of strain on the server when it's run and there'd also be no guarantee that the competitor site wouldn't keep up. It wouldn't be very difficult to write a script that uses CFHTTP to parse a page on Evik's site and parse it to find out the value of today's path to the images directory.

Another option presented by Che was to use Flash to deliver the images. It's simple enough to do in Flash and would prevent the images from being stolen. The downside to this is that the page download size would increase and that all site visitors would now be required to have the appropriate Flash plug-in.

— continued on page 38

About the Author

Simon Horwith is co-technical editor of CFDJ*, and chief technology officer of eTRILOGY Ltd., a software development company based in London, England. Simon has been using ColdFusion since version 1.5 and is a member of Team Macromedia. He is a Macromedia Certified Advanced ColdFusion and Flash developer and is a Macromedia Certified Master Instructor. In addition to administering the* CFDJ *List mail list and presenting at CFUGs and conferences around the world, he has also been a contributing author of several books and technical papers.*
simon@horwith.com



Introducing... Flex!

Easily generate rich Internet applications

By Simon Horwith

By now you've probably heard about Flex, Macromedia's most recent offering in their line of server-side products. What is Flex? How does it work? Is Flex a replacement for Flash or ColdFusion? Is it something you should be evaluating as a solution? What do you need to begin development with Flex?

In this article I'll introduce you to Flex and answer these and other questions that ColdFusion developers might have about Macromedia Flex. I'll start at the beginning – what is Flex, anyway?

Flex is a J2EE application, just as ColdFusion is a J2EE application. Like ColdFusion, Flex can be installed as a stand-alone application (which installs an underlying JRun J2EE application server) or it can be deployed for use on an existing J2EE application server. Flex is not a replacement for ColdFusion nor is it a replacement for Flash. Flex is a J2EE application that reads in XML files and translates them into Flash. MXML is the XML flavor created by Macromedia for use by Flex. Let's look at what Flex can do and then revisit where it sits in relation to ColdFusion and Flash.

Flex is a presentation-layer server. That is, Flex is responsible for generating a user interface for Web applications. While this is true, Flex also allows developers to develop object-oriented applications – I'll discuss how shortly. In order to understand what Flex does, you must have a little bit of knowledge about its markup language, MXML.

MXML has far too many tags to be discussed in this article, but the majority of these tags can be categorized as being responsible for either a user interface element (such as a text input or dropdown list), a user interface layout element (such as a panel, an accordion pane, or a section of a page), or defining data (such as format and value validators, Web service feed(s), or external objects such as Flash classes or CSS files).

Obviously, if MXML contains tags that connect with Web services and other external data feeds and business logic, then it must be capable of doing more than just generating user interfaces. This is quite an understatement; you can build entire applications in Flex using nothing but MXML. The truth is, though, these applications typically won't do much without either connecting with code written in another programming language that is exposed as a Web service or calling on functionality written in ActionScript. Let's revisit the topic of where Flex sits in relation to ColdFusion and Flash.

Do We Still Need Flash and ColdFusion?

If Flex generates Flash, do I need Flash anymore? Well, yes and no. If you need to take advantage of the robust timeline features in Flash, Flex is not the solution for you. In fact, if you need

to do anything other than generate business-application front ends in Flash, then there's a good chance that Flex is not suited to your needs. That's not to say that you couldn't make it do what you want, but the Flex server isn't designed for that and it'd probably be overkill (and an over-expenditure) to use Flex in such scenarios. In addition to the features already mentioned, Flex does have some limited graphical/animation ability in the form of some simple effects, such as fade, movement, and zoom.

If you plan on using Flex rather than Flash to build complex business applications, Flash knowledge is still an essential skill. The reason for this is that Flex natively supports ActionScript, the programming language used in Flash. It can import ActionScript classes and/or components, expose its components and data to these same entities, and even has an MXML tag that allows you to write inline ActionScript within MXML files. Pretty much every MXML tag generates an (ActionScript) object that is accessible from within ActionScript classes and/or components. So what about ColdFusion?

Obviously, Flex is not a replacement for ColdFusion. It does not have the same robust server-side programming capabilities that CFML has. The two products do complement each other very well, though. ColdFusion is ideally suited to writing all of the complex business logic and database/file system integration in an application. It also has the ability to easily expose its functionality as a SOAP Web service.

Flex is ideally suited to creating front-end interfaces, but it also has the ability to consume Web services. Because Flex files are written in XML, ColdFusion developers do not have to learn all of the nuances of Flash and the Flash authoring environment. In fact, with a little XML knowledge and ColdFusion on the back end, developers can accomplish quite a lot. A SOAP Web service is not the only way that ColdFusion and Flex can be used together to build rich Internet applications (RIAs); you can also mix MXML and CFML in the same file (read on)!

Integrating Flex with ColdFusion

ColdFusion MX is a J2EE application. There is no rule that says ColdFusion has to be the only J2EE application running on a J2EE server instance – and there is no reason why other applications cannot be deployed as part of the ColdFusion EAR. In fact, it'd be very prohibitive if this were the case. Flex is also a J2EE application and with a little bit of tweaking, can be deployed as part of the ColdFusion EAR. At the time of this writing, the steps required to integrate Flex with ColdFusion aren't trivial, but the process isn't really all that difficult, especially if you are familiar with the basics of J2EE application architecture (but don't worry if you're not).

In order to integrate the two, choose the J2EE installation option when installing Flex. This will put all of the Flex files in their own folder (off of the "Program Files/Macromedia/Flex" directory structure). In this directory structure is a .war file that you unzip to a temporary directory. After modifying an XML file to disable Flex logging, copy the flex directory from Flex's web-inf directory structure to ColdFusion's web-inf directory structure. Also copy the flex-bootstrap.jar file from Flex's lib directory to ColdFusion's lib directory. The last thing you need to do is a series of copy-and-paste jobs in order to modify the doctype declaration (to support servlet filters) and copy all of the servlet, filter, and tag library mapping and setting information from the Flex web.xml file into the appropriate places in ColdFusion's web.xml file. Easy, right?

These are very general instructions and I don't recommend trying to merge Flex with ColdFusion based on what I've just outlined – use the Macromedia Tech Note found at www.macromedia.com/support/documentation/en/flex/1/flexforcf.html.

Once Flex and ColdFusion have been integrated and you've cycled the server, you can embed MXML within your CFM pages, along with any CFML – and the MXML will be processed and rendered the same way it would in an MXML page.

How Flex Works

Getting back to our original set of questions, how does Flex work? When a request for an MXML file is made (or an MXML block is encountered in a file being processed by a J2EE application configured to process MXML, such as a Flex/ColdFusion integration) the MXML is handed to the Flex application for processing. Flex converts all of the MXML tags into their ActionScript equivalents, imports a shared library, and then compiles it all into a SWF (Flash File Format). The SWF is returned to the client for viewing. There is a Flex XML configuration file node that can be modified to tell Flex to save the ActionScript it creates before compilation; otherwise the ".as" files are never saved.

Speaking of saving files, Flex performs this compilation process only the first time an MXML file is requested; after that the compiled SWF is returned without any compilation overhead. Similar to the way ColdFusion compiles its CFM files, if you modify the code in an MXML file, Flex will recompile it on the first subsequent request. Speaking of modifying XML files, it's worth noting that after installing Flex there are a few XML "whitelist" entries that must be modified if you plan to interact with any remote Java objects, Web services, or HTTP sites from Flex. See the Macromedia TechNote at www.macromedia.com/support/flex/ts/documents/whitelist.htm for more information on using the Flex whitelist(s).

So far I have focused on reviewing Flex as a possible solution and discussed the various ways in which Flex can be installed and/or configured. By now I'm sure you're wondering how easy Flex is to use (and whether or not I'm going to show any code)?

Once you get started, Flex is extremely easy to use. I'm not going to show very much MXML code in this article simply because there isn't enough space for me to do the language any justice. I will show you a few variations of "Hello World," and you can rest assured that in the very near future ***ColdFusion Developer's Journal*** will be running articles that go into MXML syntax in more depth. I should also mention that Macromedia has a new class – "Fast Track to Macromedia Flex," which I highly recommend for any developers interested in learning everything they need to know to get started building RIAs with Flex. I will be reviewing that course curriculum in next month's issue of ***CFDJ***. Now, on with some (albeit simple) code.

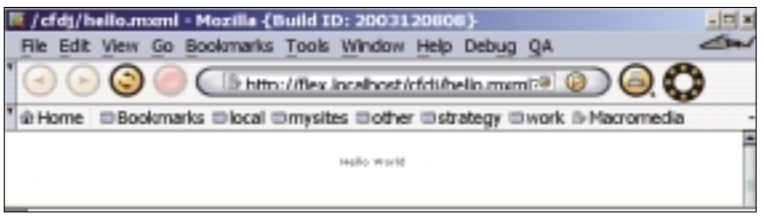


Figure 1: hello.mxml output

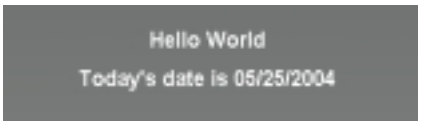


Figure 2: hello.cfm output

Taking Flex for a Spin

The first code sample I want to show is a simple "Hello World" application. I'll create a file named "hello.mxml", which contains the code in the following snippet, the results of which are shown in Figure 1. Note that I added a "backgroundColor" attribute to the "mx:Application" tag to make it clearer for print. The default background is a "halo" dark gray.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.macrome-
dia.com/2003/mxml" backgroundColor="#FFFFFF">
    <mx:Label text="Hello World"/>
</mx:Application>
```

One more simple example: I'll create a "hello.cfm" file that says hello and shows the current date using ColdFusion and MXML – and I'll add a little style to the MXML output to make it a little bit prettier, using the Flex support for CSS. (It also supports ActionScript style properties.) Note the use of <CFIMPORT> to import the Flex tag library and the use of an external stylesheet. The code is shown in Listings 1 and 2, and the output is shown in Figure 2.

Conclusion

As you can see, Flex is relatively simple to use for creating Flash front ends. Though we didn't have an opportunity to look into using MXML to do anything too complex in this article, I hope you now have a good understanding of how Flex works and where it sits in the Macromedia suite of products – and that you are confident that MXML offers developers a way to generate good-looking, robust RIA interfaces to back-end business logic.

Flex is ideal for use as a UI platform in the enterprise and offers the benefits of runtime compilation of XML and a consistent (halo) look and feel that aren't easily achieved using the "traditional" Flash authoring environment. If you have these requirements, I strongly recommend evaluating Flex by attending the Fast Track to Macromedia Flex class (read next month's issue for more on that) and/or visiting Macromedia's Web site. It is worth mentioning that at the time of this writing Macromedia has recently released a Flex Updater. You can read more about the Updater and download the code at www.macromedia.com/support/flex/downloads_updaters.html. The Flex white paper found at www.macromedia.com/software/flex/whitepapers/pdf/flex_tech_wp.pdf is a good continuation of the topics in this article.

Supported Application Servers

- Macromedia JRun 4 Updater 2
- IBM WebSphere Application Server 5
- BEA WebLogic Server 7 or 8.1
- Tomcat 4.1.29 or 5.0.18

Java Virtual Machine (JVM) Support

- Sun 1.3.1, Sun 1.4.x
- IBM 1.3 or 1.4 for use with WebSphere
- BEA JRockit 1.4 for use with WebLogic

Client Requirements

- Flash Player 7 or higher is required to view applications created with Flex.

Pricing

Flex presentation server pricing starts at $12,000 for two CPUs and includes annual maintenance. Special pricing is available for ISVs and discounts are available to government and educational organizations in certain regions. E-mail oemsales@macromedia.com for more information.

About the Author

*Simon Horwith is co-technical editor of **CFDJ**, and chief technology officer of eTRILOGY Ltd., a software development company based in London, England. Simon has been using ColdFusion since version 1.5 and is a member of Team Macromedia. He is a Macromedia Certified Advanced ColdFusion and Flash developer and is a Macromedia Certified Master Instructor. In addition to administering the CFDJ List mail list and presenting at CFUGs and conferences around the world, he has also been a contributing author of several books and technical papers.*

simon@horwith.com

VITALS
Macromedia, Incorporated
600 Townsend Street
San Francisco, CA 94103

Phone: 888 649.2990
Fax: 415 626.0554
E-mail: oemsales@macromedia.com
Web: www.macromedia.com

Listing 1: CFML/MXML code in hello.cfm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Hello World</title>
</head>

<body>
<cfimport prefix="mm" taglib="/WEB-INF/lib/flex-bootstrap.jar" />
<mm:mxml>
<mx:Application xmlns:mx="http://www.macromedia.com/2003/mxml"
themeColor="haloSilver" height="1200" width="2000">
  <mx:Style source="cfdj.css" />

  <mx:Label styleName="cfdjTitle" text="Hello World" />
  <mx:Label styleName="cfdjTitle" text="Today's date is
<cfoutput>#dateFormat(now(),"mm/dd/yyyy")#</cfoutput>" />
</mx:Application>
</mm:mxml>
</body>
</html>
```

Listing 2A - CFML/MXML code in "hello.cfm"

Listing 2: cfdj.css MXML CSS code

```
.cfdjTitle
{
  fontFamily:Arial, Helvetica, sans-serif;
  fontWeight:bold;
  color:#FFFFFF;
  fontSize:24pt;
  marginLeft:10;
}
```

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

CommonSpot™

Efficient Content Management

fast. easy. affordable.

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

With CommonSpot Content Server you get it all. CommonSpot's exceptional blend of rapid deployment, ease of use, customization and scalability make it the leading ColdFusion content management solution.

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

For larger implementations, CommonSpot scales efficiently, delivering enterprise-level capabilities like replication, static content generation, multi-site clustering, personalization and custom authentication, across a diverse set of platforms.

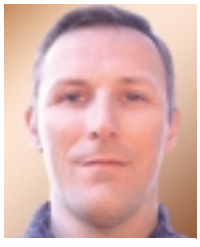
For the past six years, PaperThin has been a leader in the ColdFusion community, and CommonSpot has been the solution of choice for organizations of all sizes, including AFL-CIO, Boeing, Kaiser Family Foundation, Ohio University, PGA.com and hundreds of others. CommonSpot's sophisticated feature set and affordable pricing are an unbeatable combination.

Call us today at **800.940.3087** to schedule a live demonstration of your site running under CommonSpot, or visit **www.paperthin.com** to learn more.

PaperThin

Flash Remoting: An Alternative Approach

Bucking the conventional wisdom with CFM files



By Ian Bale

Although an "undeniably excellent" product, Flash Remoting presents a few problems. This article offers solutions in the areas of authentication and error-handling.

Most people who have experienced Flash Remoting seem to be impressed with it. I, for one, couldn't help but be impressed by the demonstration Macromedia gave at one of their seminars. (See http://reservations.broadmoor.com for an excellent example.)

Building on the component structure provided by ColdFusion MX, Flash Remoting promises the world: a platform-independent rich client that interacts seamlessly with the server. Using Flash Remoting developers can produce dynamic clients that encompass more functionality per page than was previously possible. Since the Flash client remains loaded on the client's machine, and all that is sent back and forth is the data, less bandwidth is required, and the load on the servers is lessened.

Sounds perfect! But every silver lining has its associated cloud, and the same is true of Flash Remoting. Whereas it is undeniably excellent, Flash Remoting does present its own problems. The thing is, can we get around these problems in a way that makes Flash Remoting still worth using?

How do we use this wonderful product? Well, conventional wisdom says that we use components (CFCs) on the server to collect together related functions, and call those functions directly from the Flash client. What's wrong with that? you ask. Well, there are two main reasons why I did not take this approach.

1. I wanted a central authentication system that just worked without my having to add code to every function to call the authenticator.
2. I wanted to handle errors such as message timeout, authentication errors, wrong parameters, or just about anything else in a sensible manner so that I could pass back useful error results to the client.

So why couldn't I achieve these results with CFCs? Originally I thought I could, and maybe sometime in the future (when suitable modifications have been made to ColdFusion) I will be able to, but right now I can't.

If I use an Application.cfm file, then I should be able to add my authentication there and halt the call to the CFC function if it is not authenticated. Macromedia provides a really useful function, getHTTPRequestData(), which should show the contents of the message sent to the server from the client. The problem is, it doesn't work. It supplies some information. However, some of the data that I need to access (which function was called, what parameter values were provided) is simply not available.

Without knowing which function was called, I cannot decide whether or not authentication is required. (I did not require it for every function, as some parts of the site are accessible without logging in.) Even if I did authenticate for every function, I cannot access the parameter values, so I can't access the username/password. It's possible that I could obtain the values that I need by directly accessing structures within the Java factory, but I am reluctant to use such undocumented features on critical projects, as they may not be there in future versions.

Therefore, I'm stuck with putting the authentication inside each and every CFC <function> tag. The thing I really hate about this is that if another developer comes along in the future and adds a function but forgets to add the authentication code, then we have a security problem. To my mind, this is simply unacceptable.

The other thing I wanted was to be able to trap errors and return sensible error messages to the client. <cferror> in Application.cfm would seem to do the trick; but, alas, no. When calling a CFC function, a <cferror> is ignored.

What about <cftry>/<cfcatch>? The problem is that we can only put these inside the <cffunction> tags, which means they won't trap errors such as missing or mistyped parameters. Also, as with putting the authentication inside each function, we have the problem that if a future developer forgets to add this code, then we lose our error trapping.

The Silver Lining

Okay, enough of the cloud! Where's that silver lining I mentioned earlier? Well, it comes in the form of CFM files. If you delve into the ColdFusion documentation you will find that not only can you call functions within CFCs via Flash Remoting, but you can also call code in a CFM file as if it were a function.

But who would want to put each of their functions into separate CFM files? Not me, for a start! When I said that the answer "comes in the form of CFM files" I actually meant "file," not "files." What I did was to create a "gateway" CFM file that accepts calls from Flash. This CFM file contains all the functionality needed to decide whether or not to authenticate, call the authenticator if required, invoke the required function (in a separate CFC file), and trap any errors by wrapping the whole thing in a <cftry> tag.

The main part of the server gateway code (see Listing 1) looks like this (the code for this article can be downloaded from www.sys-con.com/coldfusion/sourcec.cfm):

```
<cfif authenticated>
   <cfinvoke component="#flash.params[1].module#"
method="#flash.params[1].method#"
          returnvariable="flash.result"
argumentcollection="#flash.params[1].params#"/>
<cfelse>
   <cfscript>
          flash.result = structNew();
          flash.result.status  = false;
          flash.result.result  = "Not authenticated";
   </cfscript>
</cfif>
```

This is wrapped in a <cftry> tag to catch any errors, such as an invalid module or method name, or missing required parameters.

We also pass back the module and method name so that our Flash client knows which message is being replied to.

```
<cfset flash.result.component = flash.params[1].component>
<cfset flash.result.function  = flash.params[1].function>
```

First we call our authenticate function to do the usual username/password check and return a true/false value.

If we pass the authentication test, then we try to invoke the function in the component requested by the Flash client. If that function does not exist, or we pass it the wrong parameters, or if something goes wrong that is not trapped within that function, then the error is caught here courtesy of our <cftry>/<cfcatch> tags. If we did not trap it, then a <function>_Status message containing information about the exception will be returned to the Flash client. By trapping the error, we can instead send more relevant information back to the client.

There are many more things we can do in our CFM file to make our lives easier, and I'll cover some of them later, but first let's consider the Flash client.

The Flash Client

In order to communicate with the server from Flash, we need to define a connection to the gateway. For this example, my serverRequest.cfm is located in a directory called "client" within the Web root directory, and I'm using the Web server provided with ColdFusion. I'm sure you've all seen code like this:

```
#include "NetServices.as"

NetServices.setDefaultGatewayUrl("http://localhost:8500/flashservices/g
ateway")
gatewayConnection = NetServices.createGatewayConnection()
remotingService = gatewayConnnection.getService("client", this)
```

We need a function to prepare our remoting call. This will add the username and password to the parameters prepared for the function call.

```
function serverRequest(params)
{
   params.username = _global.username
   params.password = _global.password
   remotingService.serverRequest(params)
}
```

We need to be able to handle the results, which will be returned to our Flash function, serverRequest_Result(). This function will check the status of the result, calling a specific handler function if the call is successful, and producing some tracing if it is not.

```
function serverRequest_Result(result)

{
   switch (result.status)
   {
          case -1:
                 trace("E R R O R : Authentication error - need to
login (again)");
                 break;
          case "0":
                 trace("E R R O R : " + result.result);
                 break;

          case "1":
                 trace("R E M O T I N G : Flash remoting call
returned a success message");
```

```
                    targetFunction  = result.component + "_" +
result.function + "_Result";
                    this[targetFunction](result.result);
    }
}
```

The following code shows a typical invocation. We are calling a function, "getPageTemplate", from a component called "coursePlayer". We are passing two parameters to this function. *Note:* Because we are using argumentcollection="#flash.params[1].params#" in serverRequest.cfm, we must pass the params parameter, even if it's an empty structure.

```
flash = new Array();
flash.component = "coursePlayer";
flash.function = "getPageTemplate";

flash.params = new Array();
flash.params.param1 = "Hello";
flash.params.param2  = "World";
serverRequest(flash);
```

The following code shows our function, which is called if we have a successful reply.

```
function coursePlayer_getPageTemplate_Result(result)
{
    trace("coursePlayer_getPageTemplate_Result called")
}
```

We now have a ColdFusion CFM file that acts as a gateway to our main functions (located in various CFC files) and that centrally controls whether or not authentication is required, and if so, handles that authentication. The CFM file also manages error trapping and returns useful errors to our Flash client.

The Ramifications of #include "NetServices.as"

Now let's think a little more about our Flash client. We've all used this line of code (see Listing 2):

```
#include "NetServices.as"
```

But how many of us have stopped to think about what it does – besides the obvious of handling the remoting communication? Since the remoting is pretty much taken care of, perhaps the most important thing for us to consider is that it adds 4.5KB to our SWF file.

Who cares, you ask? Well, if you create your Flash client as a single huge movie clip, then not you. On the other hand, if, like me, you prefer to split your client into smaller parts, then read on.

Hey, 4.5KB is not that much….Well, let's put in some code to add some authentication to the message. This brings the communication code up to 6.5KB.

So what's the problem? Well, if you just add "#include NetServices.as" to each of those movie clips, then you add 6.5KB to each one. If your client consists of 50 movie clips, then you've added over 300KB. This is rather a lot, so what can we do about it?

The obvious thing is to add this code only to our root movie, using calls like:

```
_root.remotingService.myFunction()
```

That'll do it. Problem solved. End of article. Bye, see you next time. Hang on, maybe there's more than meets the eye here? Rather than doing that sort of call, I created a function in my root movie to handle it for me.

```
function serverRequest(params)
{
    params.username = _global.username
    params.password = _global.password
    remotingService.serverRequest(params)
}
```

Now I don't need to worry about adding the authentication information to my functions; it's done for me. So if we go this route, won't all the remoting call results be sent back to the root movie? Yep. So how does the root movie get the results back to the movie that originally made the call? We use:

```
Flash.path = targetPath(this)
```

This function returns a string that represents the calling movie clip's position in the hierarchy. Say we have a movie called "outer.swf". Into this we load a movie clip called "middle.swf", and into that we load "inner.swf". Inner.swf wants to make a remoting call, so it makes a call to targetPath(this), and gets back "middle.inner". The movie clip that wants to send the remoting call calls this first, and passes the result as a parameter (path), which gets sent on to ColdFusion, which passes it back in the return message. When the remoting result is received in the root Flash movie, it invokes a function:

```
function serverRequest_Result(result)
```

Using this path parameter, the result function can route the results back to the movie clip that made the call.

```
targetMovieClip = this
targetPath = result.path.split(".")

// Loop through our list of movie clips, building targetMovieClip up
into a reference to the target move clip

for (i=0;i<targetPath.length;i++)
    targetMovieClip = targetMovieClip[targetPath[i]]

targetFunction  = result.component + "_" + result.function + "_Result"
targetMovieClip[targetFunction](result)
```

This calls a function <component>_<function>_Result in the target movie clip. Component and Function are also returned from ColdFusion to enable this routing. We simply add the following line of code to serverRequest.cfm:

```
<cfset flash.result.path  = flash.params[1].path>
```

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, IIS and ASP problems in 60 seconds or less with no human interaction. Plus, our multi-user support system allows you to track support requests for each of your engineers individually, lookup server availability, receive a copy of all errors on your site in real time, and even monitor intrusion attempts on your site in real time. **For a new kind of easy listening, talk to EdgeWebHosting.net**

By the Numbers:

- 2 Rings or less, live support
- 100% Guarantee
- 99.998% Uptime
- 150 MBPS Fiber Connectivity
- 24 x 7 Emergency support
- 24 Hour free backup

EDGE WEB HOSTING

What are you WAITING for?

www.edgewebhosting.net

Win a year of free hosting*

*On Shared Hosting or the equivalent value
See http://edgewebhosting.net/cfdj for details

Shared Hosting ¥ Managed Dedicated Servers ¥ Semi-Private Servers
ColdFusion ¥ SQL Server ¥ .NET ¥ Self-Healing Servers ¥ Value Priced

What other useful features can we add to our remoting interface? A fairly common problem is clients losing connection to the server. Either the server fails, or more often, the client loses the Internet connection. We could let each part of our client deal with it, or better still, we can add a central timeout handler.

What we need to do is make note of the function calls that are sent to the server and check them off as they are responded to. If we don't get a response, then we can deal with it. In this example, we are going to pop up an information panel to tell the user that the server is not responding and ask if he or she wishes to retry or not. If the user chooses to retry, then all the messages that were not responded to will be re-sent. After we have done this, if we do eventually receive a response from the server to the original messages, we'll just discard them. Of course you can adapt and change this handling. Maybe you'd prefer to accept the original response and trash the response to the duplicate message.

First we need to make a copy of the message that we are sending. To do this, I've added the following function to RemotingSetup.as:

```
function RemotingMessage(params,messageID,secure)
{
    this.component = params.component
    this.function = params.function
    this.path = params.path
    this.params        = params.params
    this.timerID = setInterval(messageTimeout,timeoutPeriod,messageID)
     }
```

and the following declarations:

```
// Used to record messages sent to the server
var messageID = 1
var sentMessages    = new Array()

// Used to hold messages that have failed authentication, or have
timedout,
// and are awaiting resubmission.

var queuedMessages = new Array()
```

We now have a counter to provide a unique ID for each message, an array to hold a list of messages that we have sent, and a second structure to hold a list of messages waiting to be re-sent. I need to add two lines of code to the serverRequest function in order to pass the message ID through to the CF server, and to save this message in the sent messages array:

```
params.messageID = thisMessageID

_root["sentMessages.message_" + thisMessageID] = new
RemotingMessage(params,thisMessageID,secure)
```

Three more functions are required in RemotingSetup.as:

```
messageTimeout(messageID)
clearMessageQueue()
resendQueuedMessages()
```

The messageTimeout() function is invoked when the setInterval timer expires. It removes the message from the sentMessages array and adds it to the queuedMessages array. Then it opens a pop-up requester to alert the user. If the user elects to retry, then resendQueuedMessages() re-sends all the messages in the queue, and clearMessageQueue() is used to abort, resulting in failure messages being returned to the originator of each of the messages in the queue. That way no function is ever left hanging, waiting for a response.

The last thing we need to do is add a few lines of code to the serverRequest_Result function:

```
messageData = _root["sentMessages.message_" + result.messageID]
delete _root["sentMessages.message_" + messageID]
clearInterval(messageData.timerID)
```

This clears the timer and removes the message from the sentMessages array. If the message has already timed out, then messageData will be undefined and we just discard the message. If it is not, then we deal with the message as normal.

As we are now keeping a copy of the message in the Flash client, there is no need to pass information such as the path to the CF server since we can retrieve it from our new data stored in the sentMessages array.

We now have a central handler for Flash Remoting messages that will gracefully handle a loss of communication to the server, giving the user the option of resending any messages for which a response was not received. When responses are received, they are routed to whichever part of the Flash client initiated the communication.

What the Client Doesn't Know…

Before we finish, let's put our new message queue to one more use. As I mentioned earlier, some functions in my project require the user to be logged in, others do not. Obviously, the server knows which are which, so why worry about programming your client to know? Why not just let the client request a restricted function without having already forced the user to log in?

If the client requests a restricted function, the function call will fail, and the client will receive an authentication failure response. Rather than pass this failure back to the calling function, we can simply pop up a login box for the user, and ask him or her to log in. We can then automatically resend our failed request, but this time, with a (hopefully) valid username/password. We should then receive a successful response, which we can pass back to the calling function. It receives the information it requires without having been aware of the authentication failure or the re-sent message. See Listings 3 and 4 for the full code.

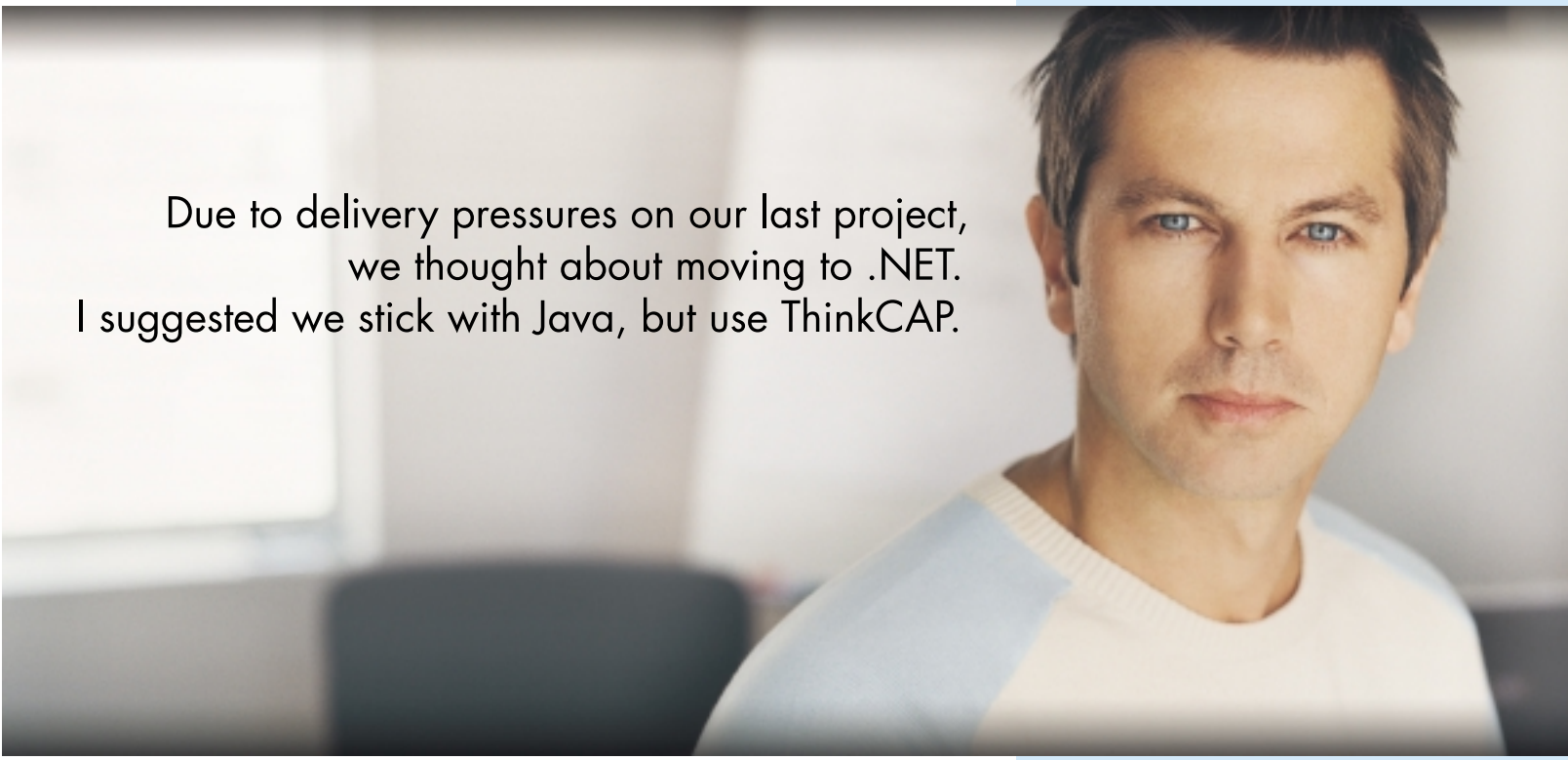
About the Author

Ian Bale is co-director of Celtic Internet ltd. (www.celticinternet.com), a UK-based software engineering and consultancy company. Ian has worked as a consultant since 1989 in both the telecom and Internet industries.

ian@celticinternet.com

Think .NET development is more productive than J2EE?

Think **again**.



Think **better**.

ThinkCAP™

ClearNova's ThinkCAP is a comprehensive application platform that simplifies and accelerates the development and maintenance of J2EE-based business applications by 50 to 70%.

ThinkCAP's visual & intuitive designers bring high productivity to business developers (those with VB or PowerBuilder-like skills), content owners, and administrators while allowing J2EE architects & programmers to leverage its component infrastructure and build business logic using the tools and approaches they prefer. ThinkCAP utilizes existing infrastructure, web services, legacy systems, and business applications.

ThinkCAP saves organizations time and money—and lowers project risks. Applications are written faster and require less maintenance. Project teams utilize in-house skills and require less training. Existing infrastructure and application servers are leveraged.

With ThinkCAP you can build quality applications faster.

Learn more about ThinkCAP at www.clearnova.com/thinkcap

CLEARNOVA

APPLICATION DEVELOPMENT • SIMPLIFIED • ACCELERATED

- Highly Visual Development Environment
- MVC Framework with Page Flow & Actions
- Advanced Data Aware Controls:
  Forms, DataViews, Queries, Navigations
  Workflows, Graphs, Treeviews, Grids, Tabs
- Smart Data Binding™ to data, objects, XML, sessions, or requests
- Browser & server-side validation
- Visual unit testing with RapidTest™
- Service Flow Designer aggregates Web Services, EJBs, XML, and POJOs
- Content Management engine & tools
- Supports .NET clients
- Integrated, seamless security
- Use any app server or 3rd party tool

CFML on J2EE: Easy as 1–2–3

Why reinvent the wheel when you can easily run your CFML applications on any J2EE server?

By Charlie Arehart

If your team is considering or being forced to move to J2EE (WebLogic, WebSphere, JBoss, Tomcat, etc.), it may be helpful to know that you don't have to throw away or even rewrite your CFML applications. Some people may want you to rewrite them, and as a long-term goal it may make sense in some cases, but did you know that CFML can run, as-is, on the J2EE server?

It really can be as easy as 1–2–3. If you've not seen it done, or you're skeptical – and especially if you've tried it with CFMX – I'll show you an alternative that you may not have seen before.

Why CFML on J2EE? Even for Skeptics

In my April 2004 ***ColdFusion Developer's Journal*** column, I wrote about the many benefits of deploying CFML on J2EE. Indeed, many folks expressed surprise to learn how much they could gain simply by running their CFML apps on a J2EE server – even if they'd never worked with J2EE before. It's still just CFML, supercharged!

If you're a J2EE developer or administrator who's been handed this article by a CFML fan hoping to keep his apps alive while following your architectural and software engineering direction, I, too, hope to persuade you. I appreciate that J2EE folks may have strong objections to considering the notion. You may not see a place for CFML in your organization.

But if the CFML is packaged as a standard J2EE Web app, do you really need to debate semantics? This solution allows you to work together to decide when (or if) to rewrite the CFML as a JSP/servlet application – while reaping all the benefits (for both J2EE and CFML folks) of running the CFML as a standard J2EE Web app until then.

In this article I want to demonstrate just how easy this can be. Whether you're an old hand at deploying J2EE Web apps, or have still not even installed a J2EE server, or perhaps don't even know what a J2EE server is, I'll show you how you can go from having a CFML application to having it run on a J2EE server in three quick, easy steps. It can literally be just a matter of drag, drop, and deploy.

BlueDragon/J2EE: A New Alternative

At this point some of you may balk; you wanted to see how to deploy CFML using Macromedia's ColdFusion MX Enterprise. That solution has been covered extensively both in previous *CFDJ* articles and in Macromedia's DevNet, documentation, and technotes. I pointed to many of these sources in my previous article (*CFDJ* Vol. 6, issue 4).

Here, instead, I'd like to show you an alternative, and one that may be new not only to J2EE folks who think ColdFusion is the only way to run CFML, but even to many CFML developers. More important to this article and to those deploying on J2EE, it's easier and more standard as a means to deploy CFML as a J2EE Web app.

BlueDragon, from New Atlanta, has become more and more familiar to many in the CF community. Just as J2EE developers can choose from many J2EE servers, now CFML developers can choose from more than just Macromedia ColdFusion MX as a solution to run their CFML. It's not a repackaging of ColdFusion (more a reverse-engineering effort), but the point is that your CFML simply runs on this alternative engine, which in the case of BlueDragon/J2EE means that your CFML runs as a J2EE Web app. No ColdFusion is required (and indeed no BlueDragon server is required, as it's just a J2EE Web app).

If you can hold your skepticism at bay, and wait a few moments for answers to your questions (price, performance, compatibility, legitimacy, etc.), I'd like to start by simply showing you the three-step deployment process. Folks who've tried to deploy on, say, WebLogic, WebSphere, or Tomcat using ColdFusion have always been very surprised at how easy this alternative is.

Download Free Trial/Developer Edition

To get started, simply visit www.newatlanta.com/bluedragon. It's a very rich site and you should eventually take the time to review the many available resources (all the documentation is available free online, and there are many self-help resources, including tag compatibility and feature comparison charts, configuration and requirements info, pricing info, FAQ, and more). There's also a very supportive mailing list staffed by New Atlanta engineers, with a searchable archive to review past discussions.

To check out the J2EE deployment solution, visit the downloads page (www.newatlanta.com/c/products/bluedragon/download/home). There are four editions of BlueDragon (two standalone server editions and the upcoming .NET edition, in addition to the J2EE edition). You'll want the J2EE edition for this article. (If you're interested in .NET instead, the .NET edition will allow you to deploy CFML on .NET. We make CFML the only language to deploy on both J2EE and .NET! More on that in an upcoming article.)

All editions are available free for trial and development. In the case of the J2EE edition, it comes first as a single-IP developer edition with no expiration date. Feel free to use it against any J2EE server you have and have your developers get started with seeing how easy deploying CFML on J2EE can be. (I say "any" because all of the major J2EE servers are supported, and BlueDragon has worked with every server tested so far. See the New Atlanta Web site and/or documentation for a current list of tested servers.) You can arrange to receive a 30-day trial license key with no IP limitations. And after the trial (or if you don't bother to get the trial license), you can continue to use the developer edition to test/create code for exploration or for deployment for production with a production license key.

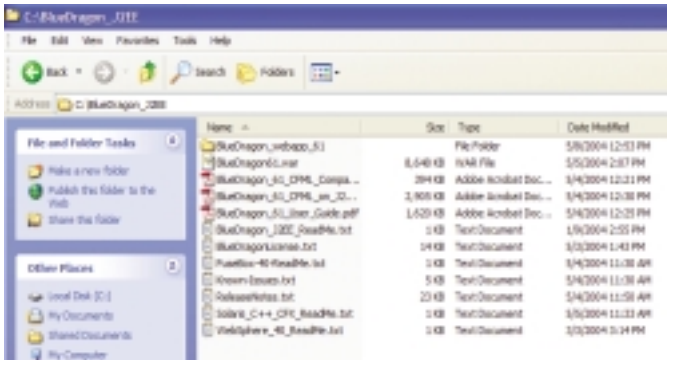


Figure 1: The downloaded BlueDragon J2EE Zip file

No Installer. No Wizard. No Server. Just Zips and WARs

When you select the J2EE edition to download, you'll see that it's just a Zip file available either as a self-extracting exe or .zip/.tar file, depending on your operating system. You can use the Zip file on Windows. Regardless of which approach you use, notice that there's no installer as with ColdFusion MX Enterprise's J2EE deployment. (Indeed, that's why it's a single-IP developer edition to start, since it's just a static Zip file that all downloaders use, so there's no license key embedded. You can easily add a licence key if you arrange to receive one.)

Just extract this Zip file to your workstation or server into the directory of your choice, as shown in Figure 1, where I've named it BlueDragon_J2EE. This isn't where the code will run. In fact, J2EE servers have a notion of deploying Web applications onto the server, so this directory will serve more as a starting point from which you'll begin the drag-and-drop process.

Notice that the directory contains mostly documentation in the form of PDFs and a couple of .txt files. Check those out after reading this article to learn more. The only two things other than docs are in fact the real meat of the BlueDragon/J2EE deployment process.

First is an available BlueDragon61.war file. Second is a BlueDragon_webapp_61 directory. You can use the first as a quick demo and the second as a model Web application with which you can deploy your own CFML application. Those who've tried to deploy on J2EE using CFMX will be surprised to see that the BlueDragon Web apps have a much smaller footprint than CFMX's version: about 10MB versus 70–110MB for CFMX.

For newcomers to J2EE, Web applications (typically JSPs and servlets) are packaged into a directory structure with certain key characteristics, notably a WEB-INF subdirectory and a web.xml file. Each of those exists in the BlueDragon_webapp_61 directory, and in the BlueDragon61.war file, which is really nothing more than a Zipped version of that directory structure, renamed as a WAR file.

Deploying BlueDragon/J2EE on Your J2EE Server

Here's where the drag-and-drop deployment process begins. You can take either the Web app directory or the WAR file and simply deploy it as-is on the J2EE server of your choice (or servlet engine, for those who know the difference). This is another difference from CFMX; not only was there no installer, but there are no steps to perform before deploying the Web application.

For a moment, let's not even worry about deploying your particular CFML application. The BlueDragon Web app as downloaded from New Atlanta has in it a sample index.cfm file that just does some basic things like calling a custom tag (as well as

Java and C++ CFX custom tags) and doing a CFDUMP. It can be used to determine if indeed the app, once deployed, is allowing CFML to run. Later I'll show you how to deploy your own CFML just as easily.

For now you can just take the WAR file (BlueDragon61.war) and either deploy it yourself (if you're running the J2EE server yourself) or give it to your J2EE administrator to deploy (e-mail it, FTP it, copy/paste it to a directory). Again, it's just a J2EE Web app. If you're not familiar with the concept of deployment, I'll explain that momentarily.

For those familiar with the concept, let me just say that the fact that the Web app's got a CFML page in it (and soon, your entire application), along with some files that run the CFML, should be immaterial and transparent to the J2EE folks. It could just as well contain the JSPs and servlets they'd normally expect; indeed, there's a sample JSP in there as well.

You may know that once the Web application is deployed, the J2EE server will know what to do with the JSP page. With BlueDragon, the entries in the web.xml file and other files in various subdirectories of the Web app will enable the CFML to run on the J2EE server.

The Concept of Deployment in J2EE

I keep using this word *deployment*. If you're new to J2EE, you may be confused. Those used to running CFML on a Web server on Windows or *nix will more generally be used to simply dropping a directory of CFML into the docroot of a Web server (like IIS or Apache), where the Web server has been configured to integrate with ColdFusion (or BlueDragon). At most you may create a new virtual directory or virtual host/Web site to point to the CFML, but there's no real concept of deployment.

With J2EE servers, things work differently. Actually, there are a few ways they can work, depending on whether you're using an external Web server like IIS or Apache or are just leveraging whatever built-in Web server exists in the J2EE server, and whether you're using EAR (enterprise archive) files or WAR (Web application archive) files.

The concept of deployment of a Web application can be nearly as simple as dropping a Web application directory into a particular place, but it generally involves extra steps taken by the app server under the covers. In the end, though, you still end up with your Web application ready to be called from a browser, perhaps using a particular port or *context path*, which is similar to a Web server mapping.

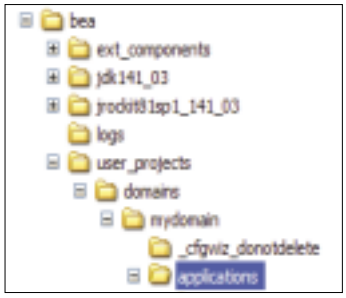


Figure 2: Auto-deployment directory in WebLogic

But what, then, does it mean to deploy a Web app on a J2EE server? In some cases, it's really no different than dropping the application into a particular directory on the J2EE server, which automatically deploys the code for you. In other cases, you may need to use an administrative console to deploy the app.

The actual steps vary. Some J2EE servers permit you to drop a Web app into a particular directory to auto-deploy it, or they may allow it only if the J2EE server has been set up in development mode. Servers permitting such drag-and-drop deployment include WebLogic, JRun, Tomcat, and ServletExec. Some servers (like JBoss) will let you drop the Web app into a particular location to auto-deploy, but it must be a WAR file (or, curiously, a Web application directory named with a .war extension).

Each of these has an admin console that can be used as well. WebSphere, prior to 5.1, on the other hand, can deploy a Web application only by use of its admin console. Fortunately, you can find a discussion of the vagaries of most common J2EE servers in the appendix of the BlueDragon document, "Deploying CFML on J2EE Application Servers."

Deploying the Sample WAR File

I'll explain the process of deploying the sample WAR file on three common servers: WebLogic, JRun, and Tomcat.

For WebLogic, let's assume that you have it installed on a Windows machine in the BEA directory. If you've left it configured in its most basic way, there will be a directory such as C:\bea\user_projects\domains\mydomain\applications\ where Web apps can be deployed. This is depicted in Figure 2.

You can simply drop the BlueDragon61.war example Web application into this applications directory. After doing so, and waiting a few moments (depending on the speed of your system and other activity in the server), you should be able to browse its sample index.cfm file using whatever port you use for WebLogic, using the name of the WAR file as the context path. So if you browse this WebLogic server as port 7001, and you dropped the file as named above, you could browse the index file as:

http://localhost:7001/BlueDragon61/index.cfm

That's it. The sample CFM file is now running on WebLogic. Of course, this is just the sample WAR file. In the next section I will show you how to deploy your own CFML. Also, you may wonder if you can have a context path other than BlueDragon61. You certainly can. Just rename the WAR file before dropping it into the applications directory.

The same approach applies to JRun and Tomcat (among other servers). You just need to know where to drop the WAR file or Web app. In JRun, assuming you've installed it in C:\JRun4, and have configured the default server, you drop it into C:\JRun4\servers\default\. In Tomcat, assuming you've installed it in C:\Program Files\Apache Group\Tomcat 4.1\, you'd drop it into C:\Program Files\Apache Group\Tomcat 4.1\webapps\.

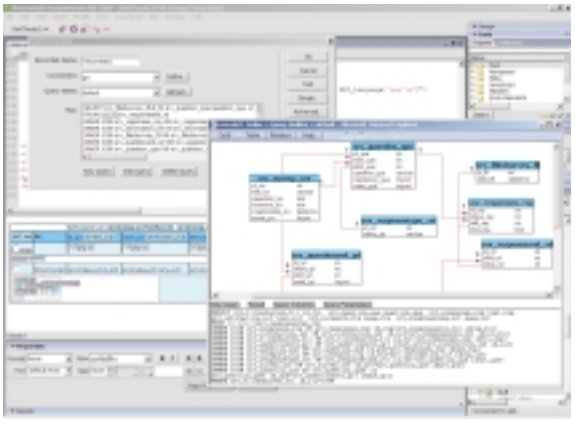
It's really that easy! Try it on your J2EE server of choice. Heck, try it on more than one. The same Web app can be deployed unchanged on each of these servers. This is another powerful benefit of the BlueDragon approach. Note, again, that in WebSphere prior to 5.1(or optionally in these other servers), you can use the admin console to deploy the Web application, as demonstrated in that appendix of the BlueDragon manual, "Deploying CFML on J2EE Application Servers."

Dreamweaver Website Development

MANY needs - ONE solution

MX Kollection

for ColdFusion and PHP



Create powerful dynamic lists

- Sort, filter and page result sets
- Perform multiple deletes
- Easily create Master/Detail lists

Build unified add/modify forms

- Client side and server side validation
- Insert into two tables
- Create form actions such as send mail or delete related record
- File/Image upload and resize

Create recordsets visually

- Build complex queries across multiple tables quickly

... and many more

The MX Kollection is a **suite of extensions** designed to **change the way you create dynamic web applications** with Dreamweaver MX.

ColdFusion and **PHP** developers will find our product enabling them to visually develop **e-Commerce, CMS, CRM, Backend** and other web solutions with increased efficiency, quality and capability.

Our customers think of it as **the next level in Dreamweaver MX visual software development.**

Download the demo and see the features and benefits of our extensions:
http://www.interaktonline.com/

MX Kollection - Professional web tools

Interakt
macromedia ALLIANCE PARTNER

CHANGING THE CONTEXT PATH

You may be wondering how you can deploy the Web application so that it uses something other than the name of the WAR file or Web app directory to be the context path for the deployed Web app. For instance, perhaps you want it deployed as "/" so that it appears to be the root of the server. You can do this, but the steps are different for different J2EE servers .

In the case of WebLogic, for instance, you'd need to create (or edit) a weblogic.xml file in the WEB-INF directory of the Web application, placing in it the following lines:

```
<weblogic-web-app>
<context-root>/</context-root>
</weblogic-web-app>
```

Notice that this is setting the context path to "/". Just be sure not to choose a path that would conflict with any other Web application you've deployed.

In JRun, you would instead edit jrun-web.xml in the WEB-INF directory of the Web application, using a similar entry:

```
<jrun-web-app>
  <context-root>/</context-root>
</jrun-web-app>
```

In the case of Tomcat, you would make a change to the single conf/server.xml file for the entire server (under the Tomcat install directory), and create a <context> entry naming the desired path and pointing to the deployed Web app. So assuming we deployed the BlueDragon61.war file, but we want to access it as "/" instead of BlueDragon61, we would use this entry:

```
<Context path="" docBase="bd61test"/>
```

For information on setting the context path in another J2EE server, see the appropriate documentation for that server.

Deploying Your CFML Application

While the previous example demonstrated use of the sample WAR file offered with BlueDragon, you'll naturally want to see it work with your own Web application. Here's where it really is just more drag-and-drop simplicity. You'll use the other of the two sample Web apps, the BlueDragon_webapp_61 subdirectory, which is the same as the WAR file, except that it is open (uncompressed) so you can see what's inside and add files to it.

Indeed, one of the simplest ways to demonstrate deployment onto CFML is to just copy your CFML from where you have it deployed now (perhaps the docroot of your development Web server) and paste it into the root of that BlueDragon_webapp_61 Web application directory.

Since you may want to do this often, you should duplicate the BlueDragon_webapp_61 directory in place (rename it to something preferable as a context path once deployed) and then simply copy your CFML into it.

Deploying the CF Example Apps

As an example, you can copy the example application code that comes with CFMX. It's found in cfdocs/exampleapps (and may be currently installed in your Web server docroot). We don't need the other directories under cfdocs, but the code in the exampleapps directory does expect to be found in a subdirectory of cfdocs.

So let's duplicate the BlueDragon_webapp_61 directory (as discussed above) and then rename it to bddemo. You can call it anything you like. Its contents would look like Figure 3

Notice that it has a WEB-INF directory, which contains the elements needed to run your CFML, as well as a sample index.cfm and index.jsp. The BlueDragon directory is there in support of the BlueDragon admin console (offered in the Web app by default, but removable if you don't want it there).

Let's create a cfdocs directory there (since the code expects to be found in one), and then copy the exampleapps directory (since we don't need all the other directories under the original cfdocs) into that directory. I found my copy installed at C:\Inetpub\wwwroot\cfdocs\exampleapps\. Again, even if you don't have this particular code, use whatever code you have and copy it into the duplicated, renamed BlueDragon webapp directory (see Figure 4).

That's really all there is to it. We can now deploy this Web application (the parent directory, now called bddemo) by simply copying and pasting it into the appropriate directory for the J2EE server, just as we did the WAR file. For instance, if I copy this "bddemo" directory into my C:\bea\user_projects\domains\mydomain\-applications\ directory, then after a moment I could browse the example applications on my machine at:

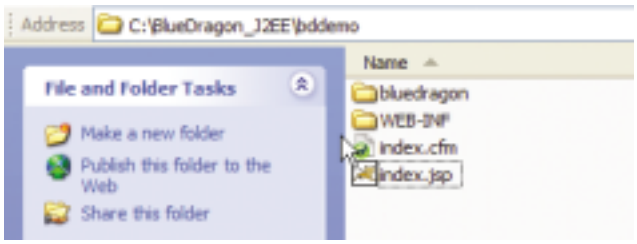


Figure 3: Initial contents of BlueDragon Web app directory renamed bddemo

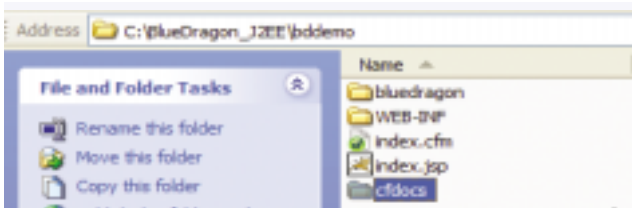


Figure 4: Contents of Web app after copy of CFML example code directory

http://localhost:7001/bddemo/cfdocs/exampleapps/index.cfm

See the sidebar "Changing the Content Path" for information on deploying the Web app to run at the root of the app server

Dealing with CFML Incompatibilities

If you try this, you'll notice that it doesn't work at first. You'll get a BlueDragon error, but this isn't a problem with deploying the CFML on J2EE. Instead, it's simply that the Application.cfm in this particular Macromedia written example application calls a template (global_vars.cfm) containing CFML that tries to use the coldfusion.server.ServiceFactory object, which is not available to BlueDragon.

The good news is that you can easily change the code in that file to work on BlueDragon by removing the reference to that particular object. It's only trying to decide whether or not the sample database, ExampleApps, is a Microsoft Access database. You can replace it with the following code snippet (save a copy of the original if you'd like):

```
<cfscript>
   request.app = structNew();
"#getPageContext().getRequest().getContextPath()#/cfdocs/exampleapps";
   request.app.dsn = "ExampleApps";

   request.app.ucase = "ucase";
   request.app.isAccess = true;
</cfscript>
```

You may encounter other CFML incompatibilities when you deploy your own code on BlueDragon. It's often a 100% plug-compatible replacement, though some minor adjustments may be needed. If you need the benefits it offers (including easier, more standard deployment on J2EE), then dealing with modest compatibility challenges can be a small price to pay. See the blog entry, http://bluedragon.blog-city.com/read/601768.htm, for more information on compatibility concerns and the distinct benefits of deploying CFML with BlueDragon.

There are just a few tags and functions that BlueDragon doesn't support. Then too, if you have a CF4 or 5 app, you would experience challenges in moving to CFMX as well. Many of the same tags that became obsolete or were deprecated in CFMX are also obsolete or deprecated in BlueDragon. See the BlueDragon manual *CFML Compatibility Guide* for more information.

One other concern in deploying CFML on J2EE is a possible need to change code due to the context path. If your code was designed to run at the root of the Web server, again, you may need to change the Web application context root to "/" (as discussed earlier), or to change the code to be more flexible about its location.

There's also the potential concern that some J2EE servers will copy and run the code from a different location than

where you place the code for deployment (depending on the app server and how you deploy). BlueDragon comes with some extended tags and functions to help with this challenge, as discussed in the BlueDragon manual *Deploying CFML on J2EE Application Servers*.

Changing CFML Once It's Deployed

You may wonder where to make the change to this CFML file in the example app, once the application is deployed on the J2EE server. You have a couple of choices. First, you may be able to simply change the file in the deployed Web app on the J2EE server. The location to do that will depend on the server and whether you deployed as a WAR file or open Web app. Again, it's still just CFML. Since I deployed the Web app as an open directory on BEA WebLogic, I can change that global_vars.cfm file as located (in my example) in C:\bea\user_projects\domains\mydomain\applications\bddemo\cfdocs\exampleapps\.

Just as you're used to with CFMX (or BlueDragon) standalone server editions, you can still edit CFML once it's deployed on J2EE. Save the changes, refresh the page, and you'll see that the page is reloaded immediately (and many find faster than on CFMX, though of course many factors in a given template and your configuration can influence performance).

Your J2EE administrators may object to your editing the files in the deployed Web application. To J2EE purists, a deployed Web app should never be changed. Instead, you should create (or edit) the Web application outside the server (in your development environment, for instance) and then redeploy it. This may involve stopping the Web application using the server's admin console, copying the newly updated Web app as you did before, and then redeploying it using the admin console's appropriate options.

Configuring BlueDragon Administrative Settings

The index.cfm page of the cfdocs/exampleapps/, as well as some of the example pages, don't use any database, so they'll run right away. For pages that need the database, you of course need to set up a data source, just as in ColdFusion. You have a couple of options for doing this. Each may appeal to different audiences and situations.

Figure 5: BlueDragon admin console

BlueDragon Admin Console

I mentioned before that there is a BlueDragon admin console that's installed as part of the Web application. That's accessible using /bluedragon/admin.cfm after the context path for your application. So for my example deployment of a /bddemo/ application on WebLogic, I'd access the admin console as follows:

http://localhost:7001/bddemo/bluedragon/admin.cfm

As in CF, this will prompt you for an admin password. Just as there is no license key by default (and it's limited to a single IP address until you arrange for a license key), so too is there no admin password defined by default (since the BlueDragon/J2EE Zip file is static, any value we might preload would be no better than loading none at all). When the admin console loads (see Figure 5), you'll see that it's very similar to ColdFusion's.

You can change the admin password in General>license & security, and you can turn on debugging for each page (as in CF) using Debugging>settings. To add a data source, follow the instructions in Data & Search>data sources. Most of the options will be familiar to CF developers and administrators, and there is online help on most pages.

> "To J2EE purists, a deployed Web app should never be changed. Instead, you should create (or edit) the Web application outside the server (in your development environment, for instance) and then redeploy it"

Some J2EE administrators will balk (again) at the notion of making changes in such an admin console, preferring that the Web application not be changed once it's been deployed. We appreciate their concern.

I mentioned that you can remove the admin console from the BlueDragon Web app before (or even after) deployment. Just remove the BlueDragon directory, as discussed earlier. You also need to remove the admin.bda file from the WEB-INF/bluedragon/ directory (within the Web application).

Bluedragon.xml File

You may wonder, then, how you can configure settings such as data sources and debugging. They're all set (some by default, some by way of the admin console) in a bluedragon.xml file, also located in the WEB-INF/bluedragon/ directory. You can edit it directly. One option is to make changes in a development version of the application and review how admin console changes lead to changes in the bluedragon.xml file.

Note that editing the bluedragon.xml or using the admin console is also the way to get the license key implemented once

you obtain one. See the option General>license & security in the BlueDragon Admin console. Once you've made these changes, you can now deploy the Web application as discussed above.

Admin Console "Build War File" Option

Still another option is to use a built-in feature of the BlueDragon admin console. See Deploy>build war file. What this does is take the application (whatever Web app the admin console is being executed within) and packages it into a WAR file (another nice feature for those J2EE admins who may want your Web app packaged that way). So you could deploy the application in a development server, make changes to its admin console, and then build the WAR file that would contain both your code and the bluedragon.xml file (and deploy that WAR file into production). More than that, this "build war file" option also automatically removes the BlueDragon admin console files.

J2EE Data Sources

Finally, rather than configure a data source in the BlueDragon admin console at all, you have the option to use J2EE data sources instead. This will appeal to J2EE admins especially. You can define a data source at the J2EE server level, using the J2EE server's admin console (such as WebSphere's or WebLogic's). When you do this, that data source's JNDI (Java Naming and Directory Interface) name is what you then would use in a CFQUERY DATASOURCE attribute. When the code runs, if BlueDragon doesn't find the data source in its admin console configuration, it will look to the J2EE server.

J2EE Sessions

Similarly, you can have the J2EE server manage sessions. In the BlueDragon admin console, under Application>variables, notice the option "J2EE Sessions", which can be set to "yes". By doing so, you will be giving control to the J2EE server to manage sessions, imparting all the benefits you may get from the J2EE server, such as replication of sessions across a cluster, or more simply (but new to many CFers) persistence of the sessions across server restarts. Note also that using J2EE sessions also instantly changes the behavior of sessions such that they end when the user closes the last instance of their browser. This is unlike traditional sessions in CF or BlueDragon, which remain persistent for a very long time because they leverage the CFID/CFTOKEN, which is also used for client variables. Using J2EE sessions tells the server to instead use a JSessionID cookie, which is kept only in the memory of the browser.

Closing Thoughts

I started the article asserting that deployment on CFML can be as easy as 1–2–3. While I've explored a few concerns that could impact some deployments, it is still true that many applications can be deployed on a J2EE server using BlueDragon in just three easy steps:

1. Duplicate the skeletal BlueDragon_webapp_61 and rename it to a suitable context path name.
2. Copy your CFML into that directory.
3. Copy/move that directory into the auto-deployment directory (for J2EE servers supporting that notion), or use the J2EE server's admin console to deploy it.

Something especially compelling about BlueDragon's approach, using a standard J2EE Web application, is that you can perform Step 3 on multiple servers: you can deploy onto WebSphere, WebLogic, Sun ONE, Oracle, JBoss, JRun, Tomcat, ServletExec, and others using the same Web app that you created in Steps 1 and 2.

And, as I discussed in my previous article, you can also use the Web app to deploy multiple times on a given J2EE server, such as in a clustered environment, or just to create different Web apps in the same virtual server, where they can coexist with others without conflict.

If a savvy J2EE administrator wonders whether the 10MB per Web app can be reduced when multiple copies are deployed – by moving some of the WEB-INF files to a directory shared by all Web apps – the answer is yes.

As I alluded to before, it's natural for J2EE administrators, along with CFML developers and their managers, to be skeptical. They've not heard of BlueDragon, and they may have an even greater aversion to CFML in general. Some other common concerns folks will have are such things as price, performance, legitimacy, etc.

Let's address a couple of these quickly. As for price, the developer and trial editions of BlueDragon/J2EE are free. A commercial license lists for $2,499/CPU, with unit prices going down with multiple CPUs, multiple servers, etc. That's less than half the retail price of CFMX Enterprise, which is the edition required to deploy onto J2EE using CFMX. A license is required for any servers on which the BlueDragon Web application (and your CFML) is deployed (other than for trial or development).

As for performance, New Atlanta has not yet published comparisons with CFMX. Customers have reported instances of applications that performed worse when moved from CF 5 to CFMX 6.1 but that performed better on BlueDragon. Still, you may find an instance where an app's performance is less than expected. There are just too many variables that can influence performance to be able to state any conclusive comparisons. New Atlanta is, however, committed to solving any performance problems that can be identified.

As for legitimacy, it's worth noting that while New Atlanta may seem new to many CFML developers, BlueDragon has been out for a couple of years. And New Atlanta has not only been in business since 1997 (with 11,000 customers in 70 countries) but they have established a reputation for engineering expertise and quality support with their server-side Java tools, JTurbo and ServletExec. The former is a SQL Server driver (offered free within BlueDragon) and the latter is a servlet engine that's embedded within the Server/Server JX editions of BlueDragon (the way JRun underlies CFMX).

It's worth noting that if you have not yet chosen a J2EE server, you can get several benefits from considering ServletExec as the server on which to deploy BlueDragon/J2EE. It, too, is available in free trial/developer editions, is priced much lower than most commercial J2EE servers, and comes with all the support New Atlanta has become known for.

But the cool thing about BlueDragon/J2EE is that it works with whatever J2EE server/servlet engine you have. More important, it makes your CFML run on such servers. If you're facing the prospect of having to rewrite/port hundreds or thousands of CFML templates to J2EE, BlueDragon/J2EE may be a very cost-effective solution that protects your investment in CFML and CFML developers.

J2EE ALTERNATIVES

One interesting area where J2EE folks have an advantage over CFML folks is in the available alternatives for running their applications. It's not unusual for an organization to consider several choices when evaluating alternative J2EE servers (or servlet engines, a difference I'll explain in a moment).

If you're in a larger organization, you'd probably lean toward IBM's WebSphere or BEA's WebLogic. If you're an Oracle shop, you'd probably choose Oracle's J2EE server. If you're fans of open source, you'd probably choose Sun ONE, JBoss, or Tomcat, to name a few. If you're a value-driven firm, you may choose Macromedia's JRun, New Atlanta's ServletExec, or any number of other alternatives.

The thing is, folks in the J2EE community have long been open to considering any of several options depending on their size, their needs, their preferences, etc. Still, they knew that any of these servers could do the most basic things they needed, running their JSPs and servlets – which are just the Java way of developing Web applications. As I discussed in my previous article, there's not much JSPs and servlets can do that CFML can't. It's just that they suit Java developers very well, since they're written in Java.

JSPs and servlets were once the sole province of servlet engines (Tomcat and ServletExec are servlet engines), but the J2EE specification and J2EE servers (such as WebLogic, WebSphere, and the like) evolved to do much more (running EJBs, supporting Java Messaging, and many other things).

Java developers (and managers) know that whatever server or servlet engine they choose, their JSPs and servlets will run on it. They can pick and choose, even develop on one server and deploy onto another. The Sun J2EE specification clarifies what JSPs and servlets, and their packaging as J2EE Web applications, should look like. It would be helpful if the CFML community also could leverage a documented specification of what CFML should do. New Atlanta would support such an effort.

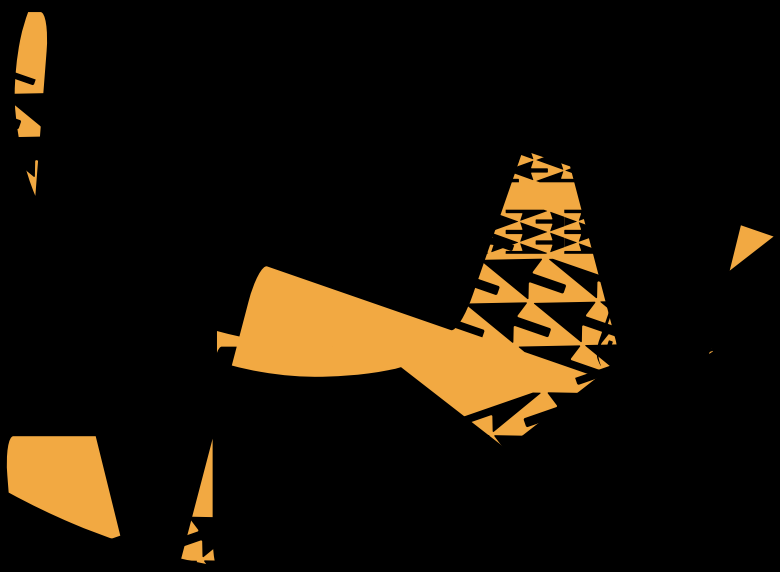
About the Author

Charlie Arehart is CTO of New Atlanta Communications, makers of BlueDragon. A Macromedia Certified Advanced ColdFusion developer and trainer, he continues to support the CFML community, contributing to several CF resources, and speaking frequently to user groups throughout the country.

charlie@newatlanta.com

FLASHFORWARD
& FLASH™ FILM FESTIVAL

NEW YORK CITY 2004, JULY 7-9
THE NEW YORKER HOTEL



THE WORLD'S PREMIER FLASH EVENT

The 12th Flashforward conference and Flash™ Film Festival, the oldest and largest Flash user conference in the world, shares the latest in design, development, education and inspiration.

FLASHFORWARD 2004 FEATURES:
- 4 In-Depth Flash Workshops
- 22 One-Hour Seminars
- 20 Q&A Sessions
- 17 "Ask the Experts" Sessions
- 11 Technology Showcase Seminars
- Flash™ Film Festival
- Macromedia Keynote
- Exhibition Area
- Flash Authors' Book Signing Event
- Networking Receptions
- Exclusive Conference Workbook with Speaker Notes

REGISTRATION:
$499 through June 11
$599 through July 6
Additional $50 Macromedia User Group discount.
$699 at the door.

ADDITIONAL INFORMATION:
- www.flashforwardconference.com
- Call toll free 1.877.4.FLASH.4
- 1.805.640.6679 outside the US and Canada

GOLD SPONSOR

macromedia®

BRONZE SPONSORS

electricrain · introNetworks · macromedia PRESS · METALIQ · New Riders · Peachpit Press

MEDIA SPONSOR

Animation · MX developer's journal

Seven Strategies for Surviving Outsourcing

Choose the one that works for you

By Hal Helms

One of the most enduring of American legends is that of John Henry, the "steel drivin' man," who pitted his strength against a machine – and won. Unlike many legends, John Henry was a real person – a former slave who was hired by the C&O Railroad to cut holes in rock into which explosives were placed in order to create tunnels. It was slow, difficult, dangerous work and John Henry did it better than anyone.

One day, a salesman came to John Henry's camp and boasted that his steam-powered drill could outwork any man, and the now-famous contest was on. John Henry won the race, drilling fourteen feet to the machine's nine, but his victory was short-lived as he died a few hours later from the stress of the competition. It's ironic, but the best thing for John Henry's reputation was his death after that victory. Had he lived, he would have seen his value as a worker diminish to be replaced by a faster, cheaper, and better method.

Today, many coders are caught up in a John Henry–like struggle. The opponent is not a steam drill but outsourcing. Five years ago, developers heard about offshore work that was being done, but the reality was still distant. Today, it is far more real as many of us have either experienced outsourcing directly or know someone who has. We can only wonder what the outsourcing situation will look like five years hence, but we do have some clues.

The Gartner Group, a highly respected IT prognosticating firm, estimates that within five years, one half or more of programming jobs will leave North America. In an interview Lou Dobbs of CNN had with a Gartner Group vice president, the VP stated that even with the problems of outsourcing, which are very considerable (and often overlooked), the cost of producing the same software offshore costs 40% of what it would cost if done domestically. But far more alarming, the quality of the offshore-born software is higher, as measured using the Capability Maturity Model (CMM), a well-known metric for establishing software quality. With the economics seeming to be so compelling, it is likely that managers will have to justify a decision to not outsource.

The upcoming U.S. presidential election has put the issue of outsourcing squarely on the table. After a brief period where programmers were the star of the "new" U.S. economy, IT jobs have steadily declined. Just how bad is the problem? Consider these facts:

- 68% of IT executives, responding to a *CIO* magazine survey, said that outsourcing would likely increase in the coming year; 30% said that it would remain the same.
- The *CIO* magazine survey further found that 11% of the responding companies had outsourced system and architecture planning and 14% had outsourced research and development – two areas that were once believed to be invulnerable to the pressure to outsource.

The great drive to outsource is based on simple economics. Compare the statistics on American versus the Indian subcontinent (India presently has about 80% of the outsourcing market):

America
GDP/capita: $35,060
Unemployment rate: 5.8%
Labor force: 142 million
Typical salary for a programmer: $70,000

India
GDP/capita: $480
Unemployment rate: 8.8%
Labor force: 406 million
Typical salary for a programmer: $10,500

Given these facts, it's no wonder that Indian software exports are over $10 billion annually – and growing at a 30% pace. Still, not everything drives software development eastwards. There are some significant negatives associated with offshore development, including:

- ***Language and proximity differences:*** Despite the fact that many offshore programmers speak English, the lack of proximity tends to bring out the inevitable differences in idioms, dialect, and pronunciation. When this leads to mis-

communication, the costs can be very high.

- ***Cultural differences:*** In some cultures, admitting to a lack of knowledge or understanding is a mark of shame, leading to the "nodding head" syndrome when frank discussion and questioning is required. This, too, has a potentially huge and negative impact on the vaunted cost savings of offshore work.
- ***Time zone differences:*** From the center of India to the center of America, the time difference is about 12 hours, a perfect offset. This means that there is no time when we share the same workday, forcing communication to be channeled primarily into e-mail and other written forms. But these have a notoriously low "bandwidth," exacerbating the other problems of outsourcing.
- ***Upward pressure on programmer salaries:*** Two years ago, programmer salaries in India were only about $8,000 annually. As India prospers, programmer salaries will naturally rise, offsetting some of the cost advantage of outsourcing.
- ***Loss of critical expertise:*** If the trend towards offshore development is too extreme, companies may find that they no longer possess needed expertise in-house. Where such expertise is critical, the danger to the company can be enormous.
- ***Endangerment of proprietary data:*** Transferring work half a world away necessarily entails giving up some of the control companies enjoy with on-site developers. Additionally, if problems arise, companies do not have the benefit of employing remedies in U.S. courts of law, but are faced with trying to work out the problem at a very long distance.

So, do we have "nothing to fear but fear itself," as former U.S. president, Franklin Roosevelt, once put it? I think the situation is considerably graver than this. While it may very well be that outsourcing turns out to be but the latest management trend du jour, its effects on us developers will be no less profound and it may take years before the ship is turned right.

Many of us concerned with the effects of outsourcing fail to see the cause of the events that distress us so greatly. The great dot-com bubble was the culmination of attitudes that many of us have held dearly, one of which was that as developers, we have an inherent right to be paid highly. And why not? Software underpins the economies of all highly-developed nations and who writes that software but us?

What we failed to see is the extent of our failure in writing this software. Repeated studies have shown that, at best, the success rate for custom corporate software is no more than 30%. Individual programmers continue to believe that their software is the exception to this rule, but this is nothing but the Lake Wobegon effect, where everyone is above average.

One CTO e-mailed me this about the reality he perceives: "I think our programmers really don't know the skill gap that exists between them and our overseas people. The local guys' knowledge really hasn't kept up with what we're seeing from India – but they still want high U.S. wages." The main lesson of outsourcing for all of us, whether we're in favor of it or not, is that this situation (low success/high wages) can't continue.

Where does this leave us? Are we laboring like John Henry against forces too powerful for us in a valiant, but doomed, effort? It will be helpful if we can more precisely define the problem. Exactly which jobs are "outsourceable?" Most analysts have identified vulnerable jobs that can be precisely defined and made into a routine. These are the "low-hanging fruit" of outsourcing. Since these jobs use workers as cogs in a machine, it's no surprise that the cogs tend to be interchangeable. Remove one expensive American cog; replace with one cheaper Indian cog.

For too long, too many of us have been relying on knowledge gained years ago. ColdFusion is the only language we know (along with a smattering of JavaScript and SQL). We write the same forms, create similar lists with drilldowns, produce edit screens, and the like. Our knowledge has not kept pace with the changes occurring in our industry. We read that within three years, over 90% of software development will be done with object-oriented languages on either the Java or the .NET platform, but still haven't found the resolve to learn them. If this describes you (to misquote Jeff Foxworthy), you may be a cog.

A biologist friend of mine is fond of saying that the law of life is: "adapt or die." In biological systems, it's the environment that goads organisms to change, to adapt, and to grow. There is, it seems, something good about an environment that pressures us to adapt or die, as American car companies found out when their cozy environment was upset by the explosion of Japanese cars into the American market. They became leaner, better focused, and, ultimately, more profitable. Welcome to the jungle.

The reality of outsourcing is the reality of the environment telling us, "adapt or die." But how? What actions shall we take? Put another way, if we are to escape being cogs, what shall we do? So what's a cog to do? Below, I've listed several "strategies for survival" specific to the new environment we face. Most of

them will require significant commitments from us and, for all of us, education – effective education – will become preeminently important. I've given them numbers, but the order in which they appear is not meant to imply that one is a better strategy than another.

Survival Strategy No. 1: Get into Project Management

It's been clear for some time that in corporate software shops, coding, as such, is losing its utility value. This is the kind of work being outsourced at breakneck speeds today. Given the low success rate for corporate software, we've got much, much bigger problems than how quickly or efficiently we can write code. The biggest problem is not that we're writing code inefficiently; it's that we're writing the wrong code altogether!

But many coders don't feel that they have much say in the ultimate success of a software project – and to be honest, many of us don't enjoy the type of work that project management entails – we just want to write code. Whatever the future of outsourcing ultimately is, I think it's a very safe bet to say that defining ourselves simply as "coders" is a losing strategy. This is the lowest of the low-hanging fruit in the IT world. And coders face the greatest threats, for not only must they be concerned with their jobs being outsourced to other humans, but the reality of machine coders draws closer each day. From smart IDEs to tools for Model Driven Architecture (MDA), the value of the ability to translate a natural language into code is being continuously downgraded.

When I teach my "Project Management with FLiP" class, I'm often asked what I think the greatest key to successful projects is. Is it the choice of language (Java versus ColdFusion versus C#)? Is it the database (Oracle versus SQL Server versus DB2)? Is it the skill of the coders?

In my experience, it's far more fundamental: neither we nor the users really know what needs to be written. The project requirements are vague and usually don't come into focus until the project is actually delivered. I've written extensively on the need for prototyping as an essential skill for project success. But the temptation to hurry the process can, for many programmers, be all but irresistible: we just want to start coding. If, though, you can successfully learn proven, successful project management skills, you can write your own ticket and will not need to worry about your job being sent anywhere.

Survival Strategy No. 2: Develop the Skills of a System Architect

The problem with project management is that it takes excellent communication skills as well as other soft skills such as empathy, warmth, and patience. Hey, if you had wanted to do that, you could have become a social worker!

For many of us, the technical challenges of software have a great appeal. We find them intellectually stimulating and the thought of giving this up is an unhappy one. The good news is that you don't need to give it up; you just need to plunge into it even more deeply.

People who can design software systems are exceedingly rare. The ability to make large-scale technical decisions has great value in any conceivable world that involves software – wherever it's actually written. If you decide to go this route, you need to understand that you're committing yourself to deepening your knowledge considerably and keeping it up to date. This is not an easy task, but it can be an extremely rewarding one. Just make sure that you don't kid yourself about your current abilities. I meet a lot of people who are good at writing code who assume that this qualifies them as system architects; it doesn't – nor does being fully acquainted with the latest buzzwords.

If you're interested in this (and after you've mastered object orientation), I advise learning about design patterns. The great physicist, Sir Isaac Newton, once said, "If I have been able to see further, it was only because I stood on the shoulders of giants." Design patterns can provide us with those "giants' shoulders."

Understanding and working with design patterns can help you avoid mistakes while guiding you into good solutions for recurring problems. The risk in design patterns is getting carried away by the buzzword aspect. While others use buzzwords and acronyms to represent a knowledge not truly gained, our goal is to gain a deep understanding of design patterns. *Warning:* the bible for design patterns, *Design Patterns*, by the so-called "Gang of Four", is not light reading. But you should expect to do a lot of this if you're going to take up the system architect strategy.

Survival Strategy No. 3: Good Enough for Government Work

If you work for a government agency (or in many cases for a government contractor), you probably don't need to be greatly concerned about outsourcing. Even if outsourcing were to succeed wildly, this work will not be outsourced. This isn't a bad strategy, although you may very well be locking yourself into doing this kind of work for a long time. If you have a constant desire for new challenges, you may feel stultified. But many of us aren't technology junkies and the rewards and benefits can be steady, if not spectacular.

Survival Strategy No. 4: Be a Small Cog

Given the considerable difficulties and expenses in establishing a viable outsourcing relationship, very small companies are unlikely to adopt outsourcing. The possible downside to this strategy is a possible lack of job stability and reduced resources compared to larger companies. Still, in the right position this can be a great strategy. It's also probably the easiest one to adopt. Be careful, though, because this "easier" environment can lull you into lethargy. You'll still need to be vigilant about managing your career.

Survival Strategy No. 5: Become a Subject Matter Expert

When I was young, I wondered why baseball players who played only a single position were paid far more than "utility" players who could play multiple positions. Wasn't more better? The answer then, and now, is "No."

Specialists are valuable because they understand the intricacies of a specific problem and can hopefully see more clearly the solution required. Because their expertise is limited in scope, it should also be deeper and that depth of experience can have a great impact on the success of a project.

That's the rationale behind the SME – the subject matter expertise. If you possess a depth of knowledge about, for example, a specific industry, you are likely to be immune to the threat of outsourcing. The key to this strategy is a true depth of both knowledge and experience. For instance, if you've worked extensively with banks, you'll recognize the unique challenges of banks and be in a position to help avoid costly mistakes.

The downside to the SME is that you may need to work more on a contract basis than as a full-time employee. If that suits you, and you possess the necessary skills, being an SME can be very lucrative. Of course, you'll be expected to keep abreast of both the current and the potential changes to your subject matter.

Survival Strategy No. 6: Become a Technology Expert

This may be the riskiest strategy of the lot. Technology experts are expected to be almost prescient in their ability to rank "winning" technologies. Further, the technological landscape is so vast that no one can be a general expert. You'll have to narrow down the scope of your endeavors.

Establishing yourself as a technology expert can also be tough. What makes someone a real expert instead of someone who understands enough about a lot to talk a good game isn't clear at first. You may find it hard to establish yourself in this strategy. If you do go ahead with it, work on getting published. You'll need to be able to count on some name recognition to make this strategy work. And remember, you're trading on people's trust in your ability to make good evaluations under great pressure. That means occasionally taking positions that may be unpopular with people above your pay grade. Better sharpen your political skills while you're at it!

Survival Strategy No. 7: Find a Company That Uses Agile Methodologies

Agile methodologies are a poor fit for the separation of specifications from programming that current outsourcing demands. The best known "agile" methodology is Kent Beck's Extreme Programming (XP) (www.extremeprogramming.org). XP asserts that software development is "about people, not processes." Agile methodologies are very much in fashion right now, but their ability to deliver the quality and type of software needed is still very much a matter of belief rather than evidence.

If you do work for an "agile" company, and if that company succeeds in its purpose for adopting agile methodologies, you'll likely have little to worry about. If you choose this path, you'll want to become as adept at agile methodologies as possible. In fact, you may be able to transform your current company into the company you're seeking. Realize, though, that this is a high-risk strategy and relies, to a great degree, on the perception of success with agile approaches within the management world.

While I don't pretend that this list is all-encompassing, it is at least interesting to note that nowhere does the most pervasive strategy appear: **Do Nothing and Hope It All Works Out**. Even if claims for the pervasiveness of outsourcing turn out to be wildly exaggerated, is it really likely that staying with a "more of the same" strategy will help us weather the next, inevitable shock to the system? In this regard, and if we allow it to, the wake-up call of outsourcing can prove to be extremely beneficial to us, whatever the future holds.

Perhaps you've come up with another strategy, one better suited to your particular skills and shortcomings. The most important thing is that you carefully consider your options, choose a strategy that works for you, and then act on it. Take very seriously how you will implement your plan. If it requires new knowledge (which it almost certainly will), map out a plan for acquiring that knowledge. In short, don't wait for a catastrophic event to compel you to action.

For further reading, I recommend all of the excellent books from Tom DeMarco. Ed Yourdon's classic, *Death March*, still has excellent insights, and Eli Goldratt's *Critical Chain* is particularly thought-provoking. Virtually anything by Edwards Deming will help you understand why projects so often fail, and what you can do about it. For continuing thoughts on, and strategies for, the challenge of outsourcing, please subscribe to my spam-free "Occasional Newsletter" at halhelms.com.

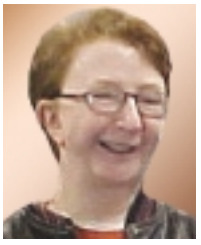
About the Author

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

hal.helms@teamallaire.com

Ensuring CF As an E-Commerce Platform

How ColdFusion developers can prove that they are the e-commerce solution



By Craig M. Rosenblum

At the company I work for, we are in the process of looking for e-commerce solutions to offer our clients – solutions developed and/or implemented in technologies other than ColdFusion. We are looking for other platforms due to the need or drive to go to the next level of stability and scalability, and to add new customization features.

The code I walked into when I started my current job was based on a very hashed-up old version of AbleCommerce. Most of my first two years was spent cleaning up code. As a department, we spent a lot of time and energy just trying to improve our current platform so we could start adding features. After awhile, it became frustrating to fix and clean up the current code and database, instead of adding all the new features, such as segmentation and personalization, that we wanted.

We researched companies that were national e-commerce solution providers. In this article, I want to show that we do not need another platform to be able to add more features, scalability, and stability; and that ColdFusion is more than capable of meeting all our needs. To do this, we need to improve the quality of our work as a community, and to better prove and demonstrate why ColdFusion is a very solid e-commerce solution.

As different technologies are introduced, or used for developing e-commerce solutions, there is a certain pattern of adaptation to move from technologies such as ColdFusion to those that are perceived as equivalent or better such as Java, .NET, ASP, or even PHP. I know from personal experience that some companies adopt new technologies not because they're the best solution, but merely because they're the newest trend or fad.

I have also seen coders not take the time or patience to really master their craft. When that happens, it makes all of us look bad – as if we are low quality, low technology, barely above HTML coders.

This article is about how we can maintain ColdFusion's hold on the market, reclaim ground recently lost to Java and .NET, improve our reputation, and improve our quality.

Improving the Quality of Work As ColdFusion Developers

Improving the quality of code they write should be the number one goal of all developers, but because so many ColdFusion developers are self-taught, as I am, it takes us more time to improve the quality of our code. As a result, we often turn in projects that meet the business requirements, but do not reach the higher quality of good solid CF coding.

Solid CF Coding Standards

Establishing standards is very important to the ease and stability of working together. The following enable someone else to pick up your code, and be able to reasonably determine logic flow, etc.

- Full documentation of code
- Clear and concise comments
- Indenting to improve readability of code
- Using SQL statements that are precise and built for scalability
- Clear project management process, to help release correct applications
- Naming conventions for variables, files, and folders that make sense to you and your fellow workers

My biggest tip, though, is thorough planning, because so many times a great application is released but is missing something, or what we thought the client wanted in the end has nothing to do with the work requested. Plan your work, follow or create a project management process, and do everything in your power to help prevent communication errors.

Improving the Reputation of ColdFusion E-commerce Sites

I believe we need to take pride in the good work that is being done on ColdFusion e-commerce sites, or even regular ColdFusion sites. Not just if it looks beautiful, but to show how using ColdFusion has improved conversion rates, sales, profitability, stability, and customer life cycles.

As ColdFusion developers who work on e-commerce sites,

we have to not only think in CF, but we must learn to understand the business of e-commerce.

There are many aspects to an e-commerce application, such as:

- Credit card validations and authentication
- Inventory status updates
- Integration between fulfillment software and Web site database
- Shipping confirmations
- Login/registration forms
- Shopping carts
- Check-out–process development/design
- Content management systems
- Personalization and segmentation of users
- Site search for products, categories, content, etc.
- Search engine optimization

To develop the best applications, we must be up-to-date, not just on code, but on e-commerce best practices, trends, and news as well.

Following are some of my favorite e-commerce resources:

- ***Internet Retailer:*** This is one of my regular visits each day, because it shows new trends in retail. What new technologies, tools, and methodologies are Internet retailers adopting? In addition, they have a very good print magazine, with a free subscription offer if you work in the e-commerce industry. www.internetretailer.com
- ***WebmasterWorld:*** The number one source for news on Google, Yahoo, and Inktomi, this is the place to learn and master search engine optimization. They also have industry representatives so you can ask questions. There are also forums on Web design, usability, and e-commerce. This is a very big community site, and it's all professionals. www.webmasterworld.com
- ***A-Clue.com:*** This is a regular e-commerce newsletter written by Dana Blankenhorn, a well-known business journalist, with good insights. www.a-clue.com
- ***Business 2.0:*** A very good subscription-based online newspaper, it's also a solid print magazine, with news in the field of online business, aka e-commerce. www.business2.com/b2/
- ***eRetailNews:*** This is another good news source, plus they have a yearly report called the "Best of eRetail"; a very good read, to learn best practices and proven methods of improving profitability. www.eretailnews.com/
- ***MarketingSherpa:*** A very well-run e-commerce/retail research organization, they offer free articles, that after 10 days become pay-per-view. But they are run using ColdFusion! www.marketingsherpa.com/
- ***Retail Forward:*** Another very professional e-commerce/retail research organization, they will e-mail the latest retail news to you daily. http://eretail.retailforward.com

E-Commerce Books

I have read a lot of e-commerce books, and the following is my selection of the best. If you have any recommen-

dations, feel free to comment. Let me explain a little bit about my collection. I am somewhat of an old school coder/developer, editing code in a text editor, always trying to improve my code quality and planning skills. In addition, being a craftsman to me means understanding more about who you are creating applications for – customers.

"We do not need another platform to be able to add more features, scalability, and stability"

As a self-taught developer, I have learned to collect books by high-quality authors who also have a lot of common sense. Think about it, what good is an application if none of the end users can use it or understand it? What good is an application if it doesn't meet all their needs? That's why it's important to understand, to grasp the whole picture, and communicate, plan well, and innovate. Part of my philosophy is that being a great e-commerce coder takes more than just mastering your syntax.

Here are my top choices of books on e-commerce, Web development, and usability:

- *Customer-Centered Design*, by Kreta Chandler and Karen Hyatt: How to design for the customers rather than for the company. I bought this book because it shows how the new trend of e-commerce development is really about satisfying our customers and improving their experience. One of the reasons is that sometimes corporate policy or opinion can indicate design/development, rather than understanding or querying customer needs.

  We cannot merely guess our customer needs, but if we want to improve reputation and sales for the long term, we must create applications that fit the end users' needs. The e-commerce trend now is identifying and fulfilling the customer's needs, so as to improve the overall customer life cycle.
- *Homepage Usability* by Jakob Nielsen: This book is one of the classics of design/development, because usability guru, Jakob Nielsen, of www.useit.com, looks at 50 top sites, e-commerce and not. He gives clear definitions of what makes Web pages more usable for customers/visitors, and includes screen shots of each major site, to provide step-by-step diagnosis, usability hotspots, and issues. Usability is not a trend; it's about making sites usable for the end user.
- *E-Business to Go – Insider Secrets* by G. Liam Thompson: G. Liam Thompson is one of the industry's earliest patriarchs, the e-business expert that the other e-business experts call when they want the straight story – or the bottom line. Thompson is also one of the few award-winning e-business developers who hasn't let it go to his head, and we like that about him, too. He is a longtime contributor to trade journals, print magazines, and online resources.
- *The Dot.Bomb Survival Guide* by Sean Carton and Christopher Locke: This is a story of success, failure, and the happenings of the dot-com explosion, from burning success to implosive failure stories. For all of us who were either victims or bystanders, this is a great book to make sure we really learned from the lessons of the dot-com era.

Conclusion

If we want to have the industry recognize ColdFusion as the premier solution for developing e-commerce sites, we must prove our viability by:

1. Adapting and sticking to solid coding standards
2. Planning each project thoroughly
3. Becoming knowledgeable about the e-commerce/online retail industry
4. Awarding and recognizing top CF e-commerce sites

Only then will the industry wake up to realize that ColdFusion dominates e-commerce!

About the Author

Craig M. Rosenblum is a Certified ColdFusion Developer, working for Rockler Companies, Inc, where he specializes in developing applications for e-commerce.

crosenblum@hotmail.com

CommonSpot 4.0 from PaperThin, Inc.

Fantastic 4

By Steve Drucker

CommonSpot Content Server, developed by PaperThin, Inc., has been simplifying Web site development and authoring since its introduction in 1998. On April 8, 2004, the newest generation, version 4.0, was released.

Web Content Management products usually seek to empower nontechnical users by allowing them to modify Web content from their desktops; enforcing standards; and categorizing, structuring, and cataloging information in a way that makes it easy to locate. With over 200 commercially available products spanning every conceivable market niche, CommonSpot has earned a reputation for ease of use and best overall value among its competitors. The fact that it's written and extensible through ColdFusion should come as no surprise to anyone.

Easy Development

I'm a big fan of Occam's Razor – a logical principle attached to medieval philosopher William of Occam that clearly states the following: "one should not increase, beyond what is necessary, the number of entities required to explain anything."

PaperThin has wisely followed this mantra as it pertains to CMS development. Developing templates with CommonSpot requires mastery of just two simple tags. The net effect of this simple architecture enables you to create a template in a very short amount of time with no learning curve. The following <CFINCLUDE> definition invokes the menu bar, as depicted in Figure 1.

```
<cfinclude template="/commonspot/pagemode/pagemodeui.cfm>
```

The second tag defines an editable content area that allows a user to define rows and columns, and insert any of the 50 "out of the box" interface widgets that ship with the product. These include the following:

- ***Layout:*** HTML table control
- ***Formatted text block:*** IE-based WYSIWYG editor
- ***Image:*** JPG, GIF, PNG image with version control and metadata
- ***PDF:*** Embed a PDF file within a page
- ***PowerPoint:*** Autoconvert a MS PowerPoint file to HTML
- ***MS Word:*** Autoconvert a MS Word file to HTML
- ***Page Index Element:*** Dynamic set of links
- ***Breadcrumb:*** Breadcrumb navigation links
- ***Simple form:*** Define a data input/update form
- ***Datasheet:*** Extract and display database content in a sortable HTML table
- ***SearchForm:*** Defines a full-text or keyword-based search form
- ***SearchResults:*** Captures data input from the searchform control and performs searching through the Verity VDK/K2
- ***QBE (new):*** Query by example allowing you to define an advanced search interface on custom-defined objects

```
<cfmodule template="/commonspot/utilities/ct-render-named-element.cfm"
    elementtype="layout"
    elementname="contentlayout"
    basetemplatename="common">
```

Once a "base template" built in ColdFusion is defined, you can create your own custom template "layers" through the CommonSpot authoring user interface. No additional coding is required. PaperThin's model for template layering is unique among CMS vendors and has repeatedly been cited by CMSWatch (www.cmswatch.com) as a best-of-breed implementation.

Enhancements to the Authoring Experience

PaperThin has added enhanced spell-check capabilities to CommonSpot. Whereas some systems allow you to only spell

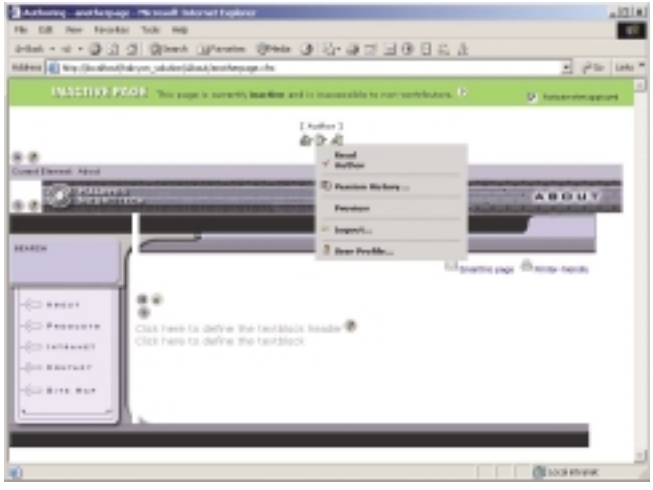


Figure 1: Web page created using CommonSpot, in "author" mode; authoring interface displayed

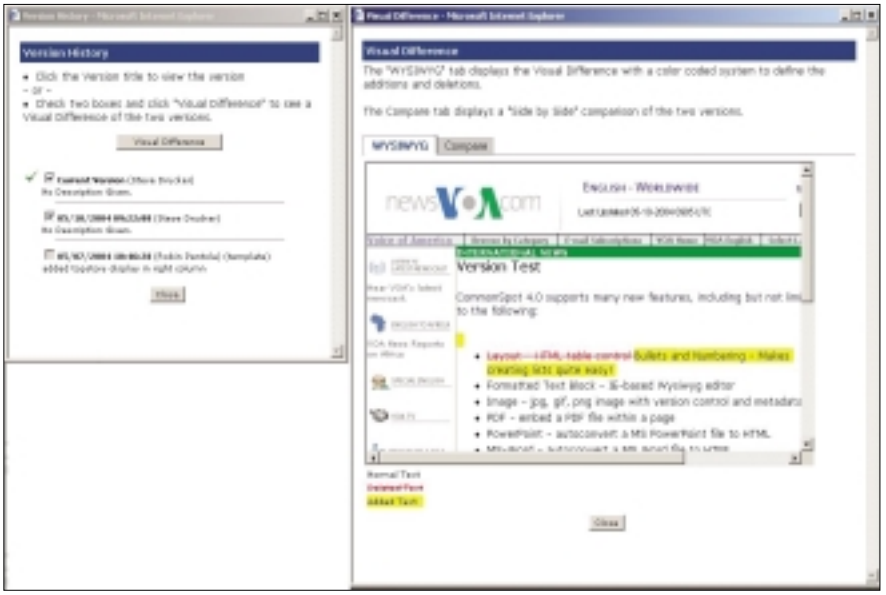


Figure 2: CommonSpot's new visual difference feature in v4.0 shows changes made to published content for easy review by approver

check content within a WYSIWYG editor, CommonSpot now allows you to spell check virtually every data input field within the system. Administrators may create custom dictionaries as well as allow users to maintain their own personal dictionaries. I found the spell-checking interface to be quite intuitive, yet somewhat frustrating, as many of the technical terms that I was entering were not part of the default dictionary. If you plan on enforcing the spell-checker popup, I strongly suggest that you add proper nouns (like your company's name, for instance) to a custom dictionary before roll-out to your content authors.

Perhaps the most visible upgrade in 4.0 from a contributor and approver perspective is the addition of a "visual difference" feature that now allows you to compare and contrast changes to an article or page quickly and easily. As depicted in Figure 2 and bearing some resemblance to Microsoft Word's track-changes feature, additions to content are highlighted in yellow while deletions are marked as red strike-throughs.

Virtually Unlimited Scalability and Flexibility in Deployment

For years, the "dirty little secret" among application developers has been the knowledge that in order to make any Web-based system scale to a virtually unlimited number of clients, you must throw out your reliance on an application server and database to process pages on demand. Even the most efficient systems cannot hope to achieve the performance of static HTML delivery. CommonSpot 4.0 now supports an add-on module that can automatically differentiate between pages that may require dynamic processing at runtime and those that do not. Pages that meet the latter criteria are generated once at publish-time and placed into a static site (i.e., http://static.figleaf.com.) whereas dynamic pages are placed in a mirrored URL (i.e., http://dynamic.figleaf.com). A user navigating through the Web site bounces between the two virtual Web sites transparently. Additionally, you can now stipulate the file extension (ASP, JSP, CGI, etc.) for generated pages. This allows you to leverage available code from practically any application server on your deployed site. The product actually supports two forms of deployment – clustering and replication. In the former, file-based content for a Web site is replicated across multiple servers via FTP or Windows Networking; however, all servers point to the same database. In a replication model, both file-based content and database entries are propagated to each server. So, you may create duplicate file systems, duplicate databases, both, or neither, depending on your network infrastructure.

Custom Support for Internationalization

CommonSpot natively supports 11 languages out-of-the-box: Danish, Dutch, English, Finnish, French, German, Italian, Norwegian, Portugese, Spanish, and Swedish. Through a little bit of coaxing, however, I was able to develop and deploy a CommonSpot-based site using ColdFusion MX and Microsoft SQL Server that supported an additional 16 eastern European languages using UTF-8 character encoding. These included Slovak, Greek, and Macedonian. You can view the fruits of this effort at the following URL: www.neweuropereview.com. A second CommonSpot site that I am developing will include content in over 40 different languages, including Arabic, Hebrew, Chinese, and Japanese. Hopefully by the time this article is published, an "officially sanctioned" language extension patch will be available. In addition, CommonSpot now supports all international date formats and creates custom field types that allow you to apply whatever formatting and form field validation you wish.

No Developer API, but You Don't Really Need One…

Over the last three years I have developed dozens of Web sites using various iterations of CommonSpot. Throughout each development cycle I have been able to meet 100% of my customer's expectations, despite the absence of a published developer API. How was this possible? For starters, CommonSpot's rich library of pre-defined elements

PaperThin, Inc.
300 Congress Street
Suite 303
Quincy, MA 02169

Phone: (617) 471.4440

Toll Free: (800) 940.3087

Fax: (617) 471.4465

E-mail: info@paperthin.com

product review

meets most of our customer requirements out of-the-box – meaning that little or no customization was required. For cases where we must capture specific content attributes or apply custom validation, we use CommonSpot's custom element definitions.

Custom elements were first introduced in version 3.0 and have been improved upon in the latest release. Through a series of wizards you can define an object – a press release, for example, which has specific fields (title, teaser, body text, publish date, etc.) and associated form input types (text field, textarea, WYSIWYG editor, calendar date selector). Version 4.0 now allows you to group fields onto different screens, accessible through a tab-bar interface making data entry a bit more intuitive. You can define output formats either graphically through a WYSIWYG editor or programmatically through a "custom render handler." Content entered through a custom element may be published simultaneously to multiple locations throughout your site. Changes to a data instance propagate to all published locations at once. CommonSpot 4.0 now allows you to create your own custom field types, meaning that you may develop your own UI widgets using Flash, client-side Java, DHTML, or virtually any other client-side technology and include them as part of your data entry form.

Drawbacks

Curiously, the only feature that is altogether absent from the product is support for XML-based technologies. While you could certainly extend the product by hacking an RSS feed in ColdFusion that reads directly from CommonSpot's database structure, it sure would be nice to see this added to the product as a point release. Also, some XML-based import/export utilities might come in handy as well.

The Value Proposition

With pricing starting at $19,500 for a base authoring license and 10 content contributors, one would be hard-pressed to label the product as "inexpensive." However, once you consider the rich out-of-the-box feature set, the associated reduction in custom development time (re: headaches), its price relative to its peers, as well as PaperThin's longstanding track record of stuffing a ton of functionality into each release, it's easy to understand how CommonSpot has prospered in a highly competitive marketplace. Thumbs up!

About the Author

Steve Drucker is the CEO of Fig Leaf Software, a Macromedia Premier Consulting and Training Partner with offices in Washington DC, Atlanta, GA, and Chicago, IL. He has been developing with CommonSpot since version 2.5 and authored an end-user and developer's training course for PaperThin, Inc. Steve is also a Macromedia-certified Master Instructor for Dreamweaver, Flash, and ColdFusion.

sdrucker@figleaf.com

cf community

— continued from page 7

Matt Williams offered another solution that is, if nothing else, humorous. He suggested renaming a few of the more prominent images on the site and adjusting the code so the images are still delivered, then creating images with the old names so that the images are still there, but these images are of porn or other unsavory content such as an image that simply says "This image was stolen from…". Part of me likes the way Matt thinks, though this is a bit extreme for my taste. The Jolly Green Giant said it best: "diabolically inventive."

There was some discussion about possible images to use until Philip Arnold brought everyone back to reality, noting that there may be legal repercussions for the owner of the server hosting the adult content. Everyone decided that if Evik goes with this solution, all images will be "Hello Kitty" or a message to the effect: "The owner of this site steals images from other sites." Back to reality.

Philip also brought up IP Cloaking, the idea being that if the HTTP Referrer is not the local server (or if it's in a blacklist of IP addresses) they will be delivered alternative content. Some companies use this approach in a technique known as "Search Engine Masking" in order to deliver specific content to search engine robots. If Evik has access to software that will do this for him using his Web server, then it's a definite possibility. If he were to take the approach of delivering images via CFM files and CFCONTENT (so that the image requests are in effect HTTP requests for .cfm files), then this could be achieved with a little CFML. One thing to keep in mind however is that it's very easy to spoof an IP address and modify the HTTP request values.

I'm not sure what Evik eventually decided to do about his problem. My guess is that he went with his initial hunch – the idea to deliver the images from a non-accessible directory. That solution carries less overhead and is more effective than any other solution that was suggested. The downside to that approach is the strain on the server. If the conditions and implementation are done right, Evik could reduce this overhead by caching the image content in a persistent scope in memory. This would be my advice to Evik, unless he is willing and able to watermark all of the images with copyright/ownership information (this would introduce no overhead, would be a one-time fix, and would solve the problem).

This thread illustrated the fact that there are often many possible solutions to a problem – even one as simple as how to store and deliver images. Perhaps there are more options that weren't explored or brought up. Join the *CFDJ-List* and let him know!

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
FLASHFORWARD	WWW.FLASHFORWARDCONFERENCE.COM	877-435-2744	27
ACTIVEPDF	WWW.ACTIVEPDF.COM		4
CFDJ	WWW.SYS-CON.COM/COLDFUSION/	888-3035282	49
CFDYNAMICS	WWW.CFDYNAMICS.COM	866.233.9626	23
CFUN	WWW.CFCONF.ORG/CFUN-04/	301-424-3903	COVER III
CLEARNOVA	WWW.CLEARNOVA.COM/THINKCAP		17
CF RESOURCE CD	WWW.SYS-CON.COM/FREECD	888-303-5282	47
EDGE WEB HOSTING	WWW.EDGEWEBHOSTING.NET	866-334-3932	15
HAL HELMS, INC	WWW.HALHELMS.COM		29
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM/CFDJ	877-248-4678	25
IS&SECURITY	WWW.ISSJOURNAL.COM	888-303-5282	33
INTERAKT ONLINE	WWW.INTERAKTONLINE.COM		21
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800.379.7729	COVER IV
IT SOLUTIONS GUIDE	WWW.SYS-CON.COM	201-802-3021	43
JAVA RESOURCE CD	WWW.SYS-CON.COM/FREECD	888-303-5282	42
MACROMEDIA	WWW.MACROMEDIA.COM/GO/DWUPDATED		COVER II
MACROMEDIA	WWW.MACROMEDIA.COM/GO/DWUPDATED		3
MXDJ	WWW.SYS-CON.COM/MX/SUBSCRIPTION.CFM	888-303-5282	35
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	11
SYS-CON ENEWSLETTER	WWW.SYS-CON.COM	888-303-5282	49
SYS-CON MEDIA	HTTP://DEVELOPER.SYS-CON.COM	888-303-5282	39
SYS-CON PUBLICATION	WWW.SYS-CON.COM		31
SYS-CON REPRINTS	KRISTIN@SYS-CON.COM	201-802-3026	41
WEB SERVICES EDGE EAST	WWW.SYS-CON.COM/EDGE	201-802-3045	39
WINMILL SOFTWARE	WWW.WINMILL.COM	888-711-6455	6

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.

ColdFusion Components and Data Abstraction

CFCs provide basic object functionality to CF developers

By Ben Forta

I've been discussing ColdFusion Components on and off since we first introduced them – which was in ColdFusion MX – and even dedicated my entire keynote time slot at our 2002 conference to CFCs, as well as covering them extensively in prior *CFDJ* columns. But, apparently, many users have yet to take advantage of these important application building blocks, or have failed to fully appreciate their necessity. So I am going to dedicate my next two columns to explaining the most important aspect of ColdFusion Components – data abstraction.

The Separation of Presentation from Content

The best way to understand data abstraction and the resulting need for CFCs is to look at an example:

```
<!--- Get employees --->
<CFQUERY NAME="employees"
         DATASOURCE="CompanyInfo">
SELECT Emp_ID, LastName, FirstName
FROM Employee
ORDER BY LastName, FirstName
</CFQUERY>

<!--- Display list --->
<UL>
 <CFOUTPUT QUERY="employees">
  <LI>#LastName#, #FirstName#</LI>
 </CFOUTPUT>
</UL>
```

This is really simple code. You've seen code like this, and have likely written something much like it many times. It uses <CFQUERY> to retrieve an employee list from a database table, and then displays the names in an HTML list. Simple enough.

So, what is wrong with this code? It is commented, uses case consistently, is indented, doesn't have extraneous # characters, and, best of all, it works! So, is there really anything wrong with it?

Well, maybe not. After all, if the code works, then there's nothing wrong with it. The fact of the matter is that applications are written to do a specific job, and if that job gets done, then the application is doing exactly what it was intended to do. That's not wrong. If it works, there is no wrong per se.

But having said that, there is no right either. There are lots of ways to write applications, with pros and cons to each (usually). And the code above has one big con going against it. Consider what would happen to your code if the table column names were changed to NameLast and NameFirst. Or if the table schema changed. Or of the user list was to be pulled from Active Directory instead of a database.

These are not far-fetched scenarios, back ends do change all the time. And when this happens, what will happen to your code? Obviously the above SQL statement will break, and will need to be changed to reflect the new table information. That will likely be a tedious but not impossible task. You'll need to find all <CFQUERY> tags (a simple site-wide search will find those) and make the code changes. A pain, but doable.

But that won't be enough. Look at the code again. The <CFOUTPUT> loop is referring to column names explicitly, and so those will break too. Now the change has gotten a bit more complicated.

What if your database query was returning columns used in <CFIF> or <CFSET> statements, values used in calculations, or flags that somehow affected data presentation? Think about it.

How many <CFQUERY> tags are used throughout your code? And how many columns are referred to within CFML code outside of <CFQUERY> tags? How much of your code will break when back-end changes occur?

It's a scary thought, but it's an important one. The truth is that database changes should never be able to break HTML formatting, that makes no sense at all. And if table schemas do change, that should not be able to throw errors in client-generated output. Even if a database is replaced with directory services, employee lists should still be rendered the same way.

I have explained this problem in detail previously (see *CFDJ*, Vol. 4, issue 10), so I'm not going to dwell on it here. Rather, I'll concentrate on how CFCs help overcome this issue.

CFCs to the Rescue

The problem described above is one that ColdFusion Components (CFCs) were designed to solve. CFCs are special files (they have a .cfc extension, instead of .cfm) designed to provide basic object functionality to ColdFusion developers.

Don't let the word "object" scare you; CFCs are as easy to use as the rest of ColdFusion. They are plain text files made up of tags, and they are called (invoked, to be precise) from other .cfm pages.

Here's a modified version of the previously seen code:

```
<!--- Get employees --->
<CFINVOKE COMPONENT="emps"
          METHOD="list"
          RETURNVARIABLE="employees">

<!--- Display list --->
<UL>
 <CFOUTPUT QUERY="employees">
  <LI>#LastName#, #FirstName#</LI>
 </CFOUTPUT>
</UL>
```

A tag named <CFINVOKE> invokes a component named emps (the actual file name would be *emps.cfc*, in the current directory), and calls a method (function) named list that returns a query named *employees*, which is used just like the query in the first example.

So where is the actual <CFQUERY> tag? It is in the *emps* component.

Basic CFC Abstraction

Now you obviously want to see what component *emps.cfc* looks like. Here's the code:

```
<CFCOMPONENT>

<!--- List employees --->
<CFFUNCTION NAME="List"
            RETURNTYPE="query"
            OUTPUT="false">
 <!--- Get data --->
 <CFQUERY NAME="employees"
          DATASOURCE="CompanyInfo">
 SELECT Emp_ID, LastName, FirstName
 FROM Employee
 ORDER BY LastName, FirstName
 </CFQUERY>
 <!--- Return data --->
 <CFRETURN employees>
</CFFUNCTION>

</CFCOMPONENT>
```

Everything in a .cfc file is enclosed within <CFCOMPONENT> and </CFCOMPONENT> tags. These define the component itself. The component contains one or more methods (functions), each defined with a set of <CFFUNCTION> and </CFFUNCTION> tags.

The method contains the exact same <CFQUERY> as used before. But this time a <CFRETURN> tag returns that query to the calling page.

So, in order:

1. Code in a .cfm page uses a <CFINVOKE> call to run the list method in a component named emps.cfc.
2. The list method contains a <CFQUERY> that obtains data from a database.
3. list returns the query to the calling .cfm page.
4. A <CFOUTPUT> loop in the .cfm page displays the query contents as it would any other query.

It's that simple.

The Value of Data Abstraction

Was it worth it? All we have done is add an extra file and more lines of code. Have we gained anything?

Let's go back to the scenario where something changed in the table layout. Perhaps the table columns LastName and FirstName are now named NameLast and NameFirst, respectively. You'd obviously need to change your SQL; there is no real way around that. Your updated .cfc code might look like this:

```
<CFCOMPONENT>

<!--- List employees --->
<CFFUNCTION NAME="List"
            RETURNTYPE="query"
            OUTPUT="false">
 <!--- Get data --->
 <CFQUERY NAME="employees"
          DATASOURCE="CompanyInfo">
 SELECT Emp_ID, NameLast AS LastName,
                NameFirst AS FirstName
 FROM Employee
 ORDER BY LastName, FirstName
```

```
 </CFQUERY>
 <!--- Return data --->
 <CFRETURN employees>
</CFFUNCTION>

</CFCOMPONENT>
```

Notice that the SELECT statement contains aliases (defined using the SQL AS keyword) that *rename* NameLast to LastName and NameFirst to FirstName. The database changed, and so the SQL had to change accordingly. And what about the presentation code? It can stay the same, as far as it is concerned nothing changed; it'll continue to work as it did before.

Even in an example as simple as this one, the benefits start to become apparent very quickly indeed.

Not Just for Data Retrieval

Most articles and tutorials on using CFCs use examples involving data retrieval, but that isn't all CFCs are for. All data access should be encapsulated within CFCs, including inserts, updates, and deletes. Here's a simple *update* method, used to update the name of an employee. This would be placed inside of the same *emps.cfc* file:

```
<!--- Update an employee --->
<CFFUNCTION NAME="Update"
            RETURNTYPE="boolean"
            OUTPUT="false">
 <!--- Arguments --->
 <CFARGUMENT NAME="Emp_ID"
             TYPE="numeric"
             REQUIRED="yes">
 <CFARGUMENT NAME="LastName"
             TYPE="string"
             REQUIRED="yes">
 <CFARGUMENT NAME="FirstName"
             TYPE="string"
             REQUIRED="yes">

 <!--- The update --->
 <CFQUERY DATASOURCE="CompanyInfo">
 UPDATE Employee
 SET LastName = '#ARGUMENTS.LastName#',
     FirstName = '#ARGUMENTS.FirstName#'
 WHERE Emp_ID = #ARGUMENTS.Emp_ID#
 </CFQUERY>
 <CFRETURN true>
</CFFUNCTION>
```

Now, to update an employee, .cfm code would simply need to invoke this new method and pass the required arguments (parameters) to it. One way to do this is by adding name=value pairs to the <CFINVOKE>, like this:

```
<CFINVOKE COMPONENT="emps"
          METHOD="update"
          EMP_ID="22"
          LASTNAME="Jones"
          FIRSTNAME="Bobby">
```

Another option is to use <CFINVOKEARGUMENT> tags, like this:

```
<CFINVOKE COMPONENT="emps"
          METHOD="update">
 <CFINVOKEARGUMENT NAME="Emp_ID"
                   VALUE="22">
 <CFINVOKEARGUMENT NAME="LastName"
                   VALUE="Jones">
 <CFINVOKEARGUMENT NAME="FirstName"
                   VALUE="Bobby">
</CFINVOKE>
```

Either way, .cfm code need simply pass values, and not care about what the update operation does internally.

It turns out that there is a *right* way to do things after all.

Conclusion

ColdFusion Components, first introduced in ColdFusion MX, provide the fundamental building blocks used to design applications the right way. In this column we looked at basic data abstraction, separating data (and data integration) from presentation. We'll continue this discussion next month when we look at encapsulating more than just data processing. Stay tuned!

About the Author

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

ben@forta.com

Reach Over 100,000

Enterprise Development Managers & Decision Makers with...

Lccbofkd ib^afkd plcq t ^ob) pbosf`bp) ^ka e^oa t ^ob sbka lop ^k lmm loqr kfqv ql
pmb^h ql lsbo . - -)- - - mro`e^pfkd ab`fpf l k j ^hbop ^_l rq qebfo mo l a r `qp) qeb bkqbomofpb
FQ j ^ohbqmi^`b) ^ka b j bodfkd qobkap `ofqf`^i q l absbi l mbop) mo l do^ j j bop) ^ka FQ j ^k^db j bkq

---

Don't Miss Your Opportunity to Be a Part of the Next Issue!

Get Listed as a Top 20* Solutions Provider

For Advertising Details Call 201 802-3021 Today!

'LKIV /- >ASBOQFPBOP TFII ?B AFPMI>VBA+ CFOPQ @L JB CFOPQ PBOSB+

SYS-CON MEDIA

*Qeb T loiaŷp Ib^afkd f*Qb`ek l i l dv Mr_ifpebo*

ColdFusion

For more information go to...

U.S.

Alabama
Huntsville
Huntsville, AL CFUG
www.nacfug.com

Alaska
Anchorage
Alaska Macromedia User Group
www.akmmug.org

California
San Francisco
Bay Area CFUG
www.bacfug.net

California
Riverside
Inland Empire CFUG
www.sccfug.org

California
EL Segundo
Los Amgeles CFUG
www.sccfug.org

California
Irvine
Orange County CFUG
www.sccfug.org

California
Davis
Sacramento, CA CFUG
www.saccfug.org

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

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

California
Long Beach
Southern California CFUG
www.sccfug.org

Colorado
Denver
Denver CFUG
www.denvercfug.org/

Delaware
Kennett Square
Wilmington CFUG
www.bvcfug.org/

Delaware
Laurel
Delmarva CFUG
www.delmarva-cfug.org

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

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

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

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

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

Georgia
Atlanta
Atlanta, GA CFUG
www.acfug.org

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

Indiana
Avon
Indianapolis, IN CFUG
www.hoosierfusion.com

Indiana
Mishawaka
Northern Indiana CFUG
www.ninmug.org

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

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

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

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

Maryland
Rockville
Maryland CFUG
www.cfug-md.org

Massachusetts
Quincy
Boston, MA CFUG
www.bostoncfug.com

Michigan
Portlalnd
Mid Michigan CFUG
www.coldfusion.org/pages/index.cfm

Minnesota
Brooklyn Park
Twin Cities CFUG
www.colderfusion.com

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

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

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

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

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

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

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

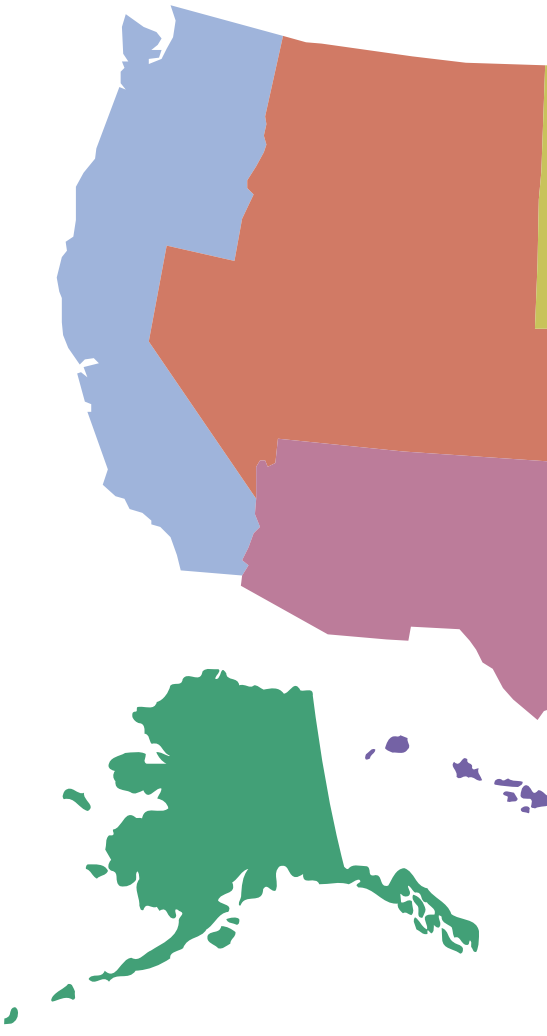
North Carolina
Raleigh
Raleigh, NC CFUG
www.ccfug.org

Ohio
Dayton
Greater Dayton CFUG
www.cfdayton.com

Oregon
Portland
Portland, OR CFUG
www.pdxcfug.org

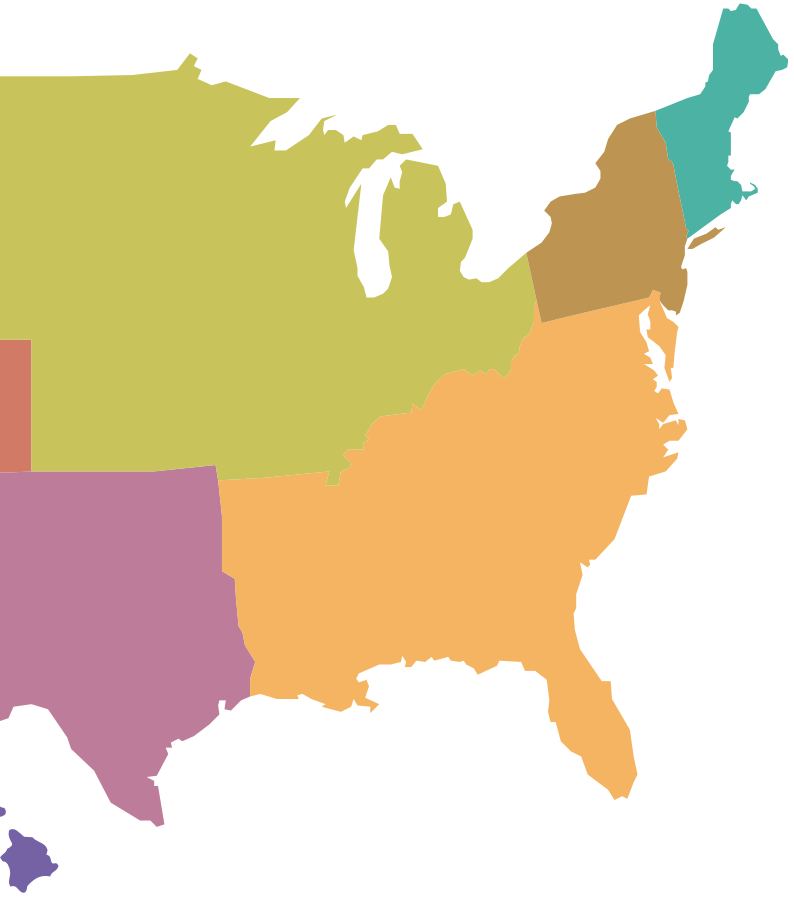
Pennsylvania
Carlisle
Central Penn CFUG
www.centralpenncfug.org

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



User Groups

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



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

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

Tennessee
LaVergne
Nashville, TN CFUG
www.ncfug.com

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

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

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

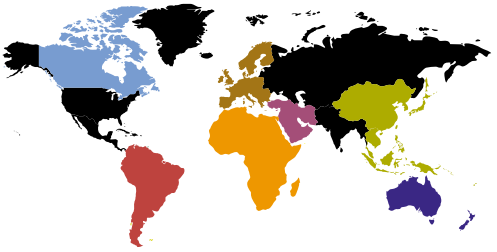
Texas
Houston
Houston Area CFUG
www.houcfug.org

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

About CFUGs

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

INTERNATIONAL



Australia
ACT CFUG
www.actcfug.com

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

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

Australia
Victoria CFUG
www.cfcentral.com.au

Australia
Western Australia CFUG
www.cfugwa.com/

Brazil
Brasilia CFUG
www.cfugdf.com.br

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

Brazil
Sao Paulo CFUG
www.cfugsp.com.br

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfugtoronto.org

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

Italy
Italy CFUG
www.cfmentor.com

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

Scotland
Scottish CFUG
www.scottishcfug.com

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

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

Spain
Spanish CFUG
www.cfugspain.org

Switzerland
Swiss CFUG
www.swisscfug.org

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

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org

Structures and Arrays PART 2

Similar but different beasts

Arrays and structures are complex data types in ColdFusion. It is important to have an understanding not only of how to use them, but also of when it is best to use one over the other.

By Jeffry Houser

This is the second of a two-part series in which I teach you about structures and arrays. In Part 1 (***ColdFusion Developer's Journal,*** Vol. 6, issue 5), I introduced structures and arrays, and investigated arrays in more detail. In this article I'll go into greater detail on structures and then compare and contrast the two complex data types.

Reviewing Structures and Arrays

A simple data type is one that contains a single value. Complex data types can be seen as a group of multiple values, all stored in a single variable. Arrays and structures are the two most common complex data types in ColdFusion. As you may remember from Part 1, you can access an individual array element by using a number, called an index. Values in a structure can be accessed in a similar manner, except that the index is a string instead of a number.

If you are familiar with other programming languages, you may have heard of something called associative arrays. Structures in ColdFusion are similar to associative arrays in other languages. The index is a name that is associated with the value you place in it. The elements that make up a structure are known as key/value pairs. The key is another name for the index and, as I said, is used to access the value. The value portion of a structure can be any valid value in ColdFusion, including integers, strings, arrays, objects, or even other structures. It might help you to conceptualize a structure as a two-column spreadsheet, with the key being one column and the value being the second column.

KEY	VALUE
Includes	inc/
Images	Images/
CustomTags	webroot/inc/CustomTags.
CFC	webroot.components

Table 1

Structures are great for storing related elements. In one of my current projects, we are using a structure to store directory information about the project, such as the location of include files, image files, custom tags, and ColdFusion Components. The elements of this structure might look like Table 1.

Throughout our application, it is easy to access the directory information we need just by grabbing the proper values out of the structure.

Creating and Using Structures

With a general understanding of what structures are, the next logical step is to show you how to create structures in ColdFusion. With arrays we used a function called ArrayNew to create the array. With structures we use a function called StructNew. This is the syntax:

```
<cfset MyStruct = StructNew()>
```

Unlike ArrayNew(), the StructNew function does not accept any arguments.

Creating a structure is easy. The next step is to learn how to add elements to the structure. To understand that, you must first know how to access elements of a structure. There are two methods for accessing elements of a structure: *dot notation* and *associative array notation* (also called *square-bracket notation*). Dot notation takes this form:

```
Structure.Key
```

It starts with the name of the structure, following that is a period, and after the period comes the key. I've also heard dot notation referred to as *object property notation* because the syntax is very similar to the way you would access properties of an object. This is the most common syntax used for accessing structure keys because it is very similar to the way you may access variables in a variable scope or columns in a query result set.

To add an element into a structure using dot notation we can use the cfset tag, like this:

```
<cfset MyStruct.Includes = "inc/">
```

The line of code starts with the cfset tag. It is followed by a space and then the name of the structure. The structure is followed by a dot and then the name of the key. After that we have an equals

sign and the value. This line of code creates the key, Includes, inside the structure, MyStruct, and gives it a string value of inc/.

Associative array notation is another way to access elements of a structure. You may remember how we accessed values in an array in Part 1. We used the name of the array, then an open square bracket "[", followed by the index number, and finished off with a close square bracket "]". To access a structure's value using associative array notation, we use the same approach. The only difference is that instead of placing a number between the brackets, we use a string. Here is an example:

```
<cfset MyStruct["Images"] = "Images/">
```

This line of code creates the key, Images, inside the structure, MyStruct, and gives it the string value of Images/.

There are quite a few benefits to using the associative array notation over the dot notation. First, your key name can have spaces or special characters. Second, you don't have to know the name of the key when setting or accessing it. The key name can be a variable or an unknown value. Dynamically named structures can be great if you don't know what you are going to be storing beforehand. A third benefit of this method is that the keys will retain their case. Using the object property notation, all keys will be converted to uppercase. Next, we'll look at an example of using array notation to dynamically name a structure.

Dynamically Named Structures

Suppose you want to keep track of how many active users are on your site. For simplicity's sake, we will assume that this is a private site and users cannot get to it without first logging in. We can use a structure in the application scope to keep track of the username and the date when they log in. We can use the username as the key and the login time stamp as the value.
The following code snippet is a simple login form:

```
<form action="loginip.cfm" method="post">
  <input type="text" name="username">
  <input type="text" name="password">
  <input type="submit">
</form>
```

The form accepts two values, a username and a password, and has a submit button. The code on the form-processing page (loginip.cfm) that will store the username and time stamp in the application scope looks like this:

```
<cfset application.Users[form.username] = Now()>
```

Under normal circumstances you'll have more code to process the login, but this line is fine for demonstration purposes. The line of code adds a new element to the users structure in the application scope. The name is the value of username, which the user entered in the login form. The value is gotten by using one of ColdFusion's date functions, now(), which returns the server's current datetime stamp. Note that before using the application scope you'll have to set it up with the cfapplication tag. Although it is beyond the scope of this article, more information on using the cfapplication tag can be found at http://livedocs.macromedia.com/coldfusion/6.1/htmldocs/tags-pa3.htm#wp1097308.

Because the "form.username" text is not inside quotes, ColdFusion recognizes it as a variable and uses the value rather than a string literal. The line of code shown earlier could also be rewritten like this:

```
<cfset application.Users["#form.username#"] = Now()>
```

It would work just as well. ColdFusion recognizes the pound signs inside the string and knows to evaluate the string. I prefer the first method because it is easier to code and easier to read. However, the second method is the one of choice if you are mixing a string literal with a variable's value to form the variable name, like this:

```
<cfset application.Users["Users#form.username#"] = Now()>
```

To help with the processing of structures, we can use the cfloop tag to loop over structures. The cfloop tag accepts two attributes for looping over a structure, the collection attribute and the item attribute. The collection attribute accepts the structure value. The item attribute specifies the name of a temporary variable. The variable can be used inside the loop to access the specific element of the structure. The following code snippet is an example of how to loop over the users structure:

```
<cfloop collection="#application.users#" item="Temp">
 <cfoutput>
  #temp#: #application.Users[temp]#<Br>
```

```
 </cfoutput>
</cfloop>
```

The snippet starts with the cfloop open tag. For the collection attribute, we specify the name of our structure, application. users. You'll notice that we specify the pound signs around it. The item attribute we specify is temp, a temporary variable. We are going to display each key and its value inside the loop. First we display the key, which is stored in the variable temp. We use the dynamic addressing and the associative array notation to get the value of the structure.

It is worth nothing that when you create a structure in ColdFusion MX, you do not have to use the StructNew function. Using a cfset tag with either type of notation will create a structure on the fly. ColdFusion will explicitly create a structure without using the StructNew function. I generally avoid this practice for two reasons. First, using StructNew to explicitly define the structure will make the code more readable when it has to be revisited in six months. The second reason is backward compatibility. Versions earlier than ColdFusion 5 do not support the creation of structures without the use of the StructNew function. However, pre-CFMX versions do support the use of periods inside a variable name, so your code will work. The difference between a structure and a variable name with a period in it is an important distinction.

Dynamically named structures can be great if you don't know what you are going to be storing beforehand

Structure Functions

ColdFusion provides a lot of built-in functions for structures. The following is a partial list:

- ***StructInsert:*** A function to insert an item into a structure. Its attributes are the structure, the key, the value, and a Boolean value that specifies whether or not to allow existing values to be overwritten. It returns true on the successful addition of the key to the structure.
- ***StructFind:*** The StructFind function determines whether the specified key is located in the structure. It returns the value associated with the specified key. Its arguments are the structure, and the key to look for.
- ***StructFindKey:*** StructFindKey is a function that will search through a structure to find the specified key. This is a recursive search, so if you have nested structures, all of the structures will be searched. Its attributes are the structure to search, the value for which to search keys for, and the scope of the search. The scope of the search is either "one" or "all". An array is returned that contains all the found values.
- ***StructFindValue:*** The StructFindValue function is similar to the StructFindKey function. It searches recursively through a structure for values that match the specified values. Its values are the structure to search, the value to search for, and the scope (one or all). It returns an array that contains all structures that have matching values.
- ***StructClear:*** The StructClear function will remove all data from a structure. It accepts one argument, the structure you want to delete. It returns a Boolean value specifying whether or not the operation was successful.
- ***StructDelete:*** The StructDelete function will remove a single key value from a structure. Its arguments are the structure, the key you want to delete, and a Boolean value that specifies what to do if the key exists. If the Boolean value is set to true, then true is returned if the key exists, and false is returned if it does not. If set to false, then true is returned regardless of whether or not the key exists.
- ***IsStruct:*** The IsStruct function will check to see whether or not a variable is a valid structure. It accepts one argument, which is the structure you want to check.
- ***StructIsEmpty:*** The StructIsEmpty function will check to see whether or not the specified structure has any keys. If it does not, then true is returned; otherwise false is returned. Its single argument is the structure you want to check.
- ***StructCount:*** The StructCount function will return the number of keys in a structure. Its single argument is the structure.
- ***StructSort:*** The StructSort function will return a sorted array of the structure's keys. It does not actually change the structure's value in memory. Its arguments are the structure you want to sort, the sort type (numeric, text, or textnocase), the sort order (asc for ascending or desc for descending), and the path to subelement. The path to subelement is for advanced sorting and is beyond the scope of this article.

You can find a complete list of all structure functions and more details on all of them in your Macromedia documentation or in the livedocs at http://livedocs.macromedia.com/coldfusion/6.1/htmldocs/functi18.htm#wp1099964.

Conclusion

Arrays and structures are similar but different beasts. You should now have a good understanding of each. Both contain multiple values underneath a single variable name. Both use an index to access an individual value, but the index is numeric for arrays and a text value for structures. Arrays are best for number crunching, table-like data, and data sorting. Structures, by their nature, cannot be sorted by value – only by key name. They are best for related data, where order is not important and direct access to an individual element is important. Many of ColdFusion's variable scopes can be accessed as structures.

About the Author

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

jeff@instantcoldfusion.com

Register123 v4.0 Includes Macromedia ColdFusion MX

(*San Francisco*) – Certain Software has launched Register123 v4.0, the company's complete online attendee management and registration service. Version 4.0 includes many new features and enhancements that were developed in direct response to client requests, including an upgrade to Macromedia ColdFusion MX.

The enhanced features of Register123 v4.0 include:

- ***Roommate matching and room sharing:*** Provides tools for organizations and attendees to save money by sharing sleeping rooms. Based on a variety of customizable profile questions such as gender, smoking preference, geographic area, job title, etc., the system can automatically pair up attendees. The planner then reviews the matches, rectifies any issues, and places the matches into available sleeping rooms.
- ***Profile change tracking:*** For improved control over the integrity of registration data, all changes to an attendee's profile are tracked and noted. Date and time, what fields were changed, who changed them (the attendee or the meeting planner), and notes for why the change was made are captured for quick reference.
- ***Invite previously registered attendees to new events:*** Planners can save time by inviting previous attendees to new events with the Profile Import Wizard.
- ***New reports interface:*** To further enhance the reports interface, the power and functionality of Register123's custom reports have been expanded. More reporting fields, filtering criteria, and layout options help planners see the information as they want it laid out.
- ***Upgrade to Macromedia ColdFusion MX:*** As part of Certain Software's commitment to keeping Register123 up-to-date with technology and functionality, the Application Server has been updated to Macromedia's ColdFusion MX, a Java-based platform. Planners will immediately notice improved performance, and Enterprise Edition clients will benefit from the ability to deploy Register123 on this J2EE-compliant platform.

Register123 provides both professional planners, and those finding themselves in the planning role, with robust tools that empower them to organize their own events and offer more control over creating event Web sites, collecting data, reviewing information, reporting on results, and managing attendee needs.

Register123 v4.0 is available now. For information, visit www.certain.com.

FuseTalk Inc. Unveils Future Release of Flagship Product on New Platform

(*Ottawa, ON*) – FuseTalk Inc. has announced the future release of its flagship product, FuseTalk 2.0, on the .NET platform. FuseTalk Inc., a provider of premium discussion forums and online collaboration applications for Macromedia ColdFusion environments, plans to target the release of FuseTalk for .NET during the third quarter of 2004. This move to a new platform will enable FuseTalk Inc. to expand its current product line and reach a larger market.

With the continued success of FuseTalk for the ColdFusion environment, the company decided to make the strategic move into the .NET market.

"FuseTalk is already reaching the J2EE platform with ColdFusion MX," says Dominic Plouffe, vice president of research & development of FuseTalk Inc. "Adopting the .NET framework is a logical progression for our products, since FuseTalk's API lends itself well to the .NET architecture, and .NET is gaining wide acceptance. Needless to say, we are still committed to serving our clients running the ColdFusion version."

Like the current FuseTalk products for ColdFusion, FuseTalk for .NET will offer the same functionality and feature set. FuseTalk for .NET will deliver tiered administration and database tools for managing large, complex, or dynamic discussion forum deployments. With FuseTalk for .NET, organizations can successfully and easily deliver as many forums as needed. The tiered approach to administration and flexible admissions-based user access, as well as the extensive feature set, will help make forum-powered online collaboration more productive and cost-effective to the organization, and make the whole experience for the user an enjoyable one.

Simply the Best: Nominations Open for *CFDJ* and *MXDJ* Readers' Choice Awards

Nominate your favorites for the 2004 ***ColdFusion Developer's Journal*** and ***MX Developer's Journal*** Readers' Choice Awards, and help highlight excellence in the solutions and services provided by the top firms in the industry. Widely referred to as "The Oscars of the Software Industry," the Readers' Choice Awards program has become the most respected industry competition of its kind. Nominations for this year's Readers' Choice Awards are open now. Voting begins on July 1, 2004 and runs through midnight December 31, 2004. Then come to applaud the winners at Web Services Edge 2005 East – International Web Services Conference & Expo.

Latest Figures Show ColdFusion Still at the Top

According to the latest monthly figures from the UK Internet services company Netcraft, the number of IP addresses with sites using ColdFusion remains higher than the number of IP addresses using any other technology, including ASP.NET, JSP, and Java Servlets.

ASP.NET is emerging as a strong challenger to ColdFusion, however, and has overtaken JSP and Java Servlets.

The number of IP addresses found with ASP.NET has shown very strong growth in the past year with a 224% increase from 17,200 to 55,800. Despite being overtaken, though, JSP and Java Servlets are the next fastest growing technology in percentage terms, with a 56% increase

CFUN04

MDCFUG

301.424.3903
info@teratech.com

Sixth Annual ColdFusion Conference

June 26th & 27th, 2004

Charlie Arehart
Jo Belyea-Doerrman
Raymond Camden
Christian Cantrell
Sandra Clark
Sean Corfield
Robert Diamond
Michael Dinowitz
Steve Drucker
David Epler
April Fleming
Ben Forta *tentative
Shlomy Gantz
Critter Gewlas
Mark Gorkin
Hal Helms
Simon Horwith
Larry Hull
Chafic Kazoun
Matt Liotta
Tom Muck
Rey Muradaz
Nate Nelson
Samuel Neff
Jeff Peters
John Quarto
Neil Ross
Stephen Shapiro
Michael Smith
Geoff Snowman
Jeff Tapper
Dave Watts

5 TRACKS!

Bootcamp - Basic ColdFusion and Flash topics
Advanced - Advanced ColdFusion topics
Empowered - Fusebox & Project management topics
Accessibility - making sites that disabled people can use, section 508
Integration - Flash, Flex and other technologies integrated with CF topics

$199
Per Person
Special Price until 3/31/04

www.cfconf.org/cfun-04/

"I learned numerous techniques last year that have helped myself and our development team to better deliver quality products to our customers. CFUN is also a great venue to meet some of the names you see in CFDJ and DevNet and talk with them one on one."

-Phillip D

be the pilot!

FREE SETUP on Shared Hosting Accounts With ColdFusion MX Support
Use Promo Code CFDJ2004

INTERMEDIA.NET

WE DARE YOU TO TAKE A FREE TEST FLIGHT!

Managing technology that runs your business is a matter of trust and control. INTERMEDIA.NET gives you both.

TRUST. Since 1995 we have been providing outstanding hosting service and technology to our clients. Don't take our word for it... take theirs.

"The support and service that you offer are nothing short of golden. The high quality of your system and service for CF customers is something one could only ever dream of." – Claude Raiola, Director, AustralianAccommodation.com Pty. Ltd.

CONTROL. We give you instant control over your site, server and account configuration changes. No more submitting requests and waiting for someone else to take action. You are in control to pilot your business through its daily needs.

BE THE PILOT. Take a free test flight and see what our HostPilot™ Control Panel offers you beyond all others. Check out our SLA guarantees. To see more testimonials and to find out about our competitive advantages, visit our Web site at www.Intermedia.NET.

Managed Hosting • Shared Hosting • Microsoft Exchange Hosting

Call us at: 1.800.379.7729 • Visit us at: WWW.INTERMEDIA.NET | HostPilot™