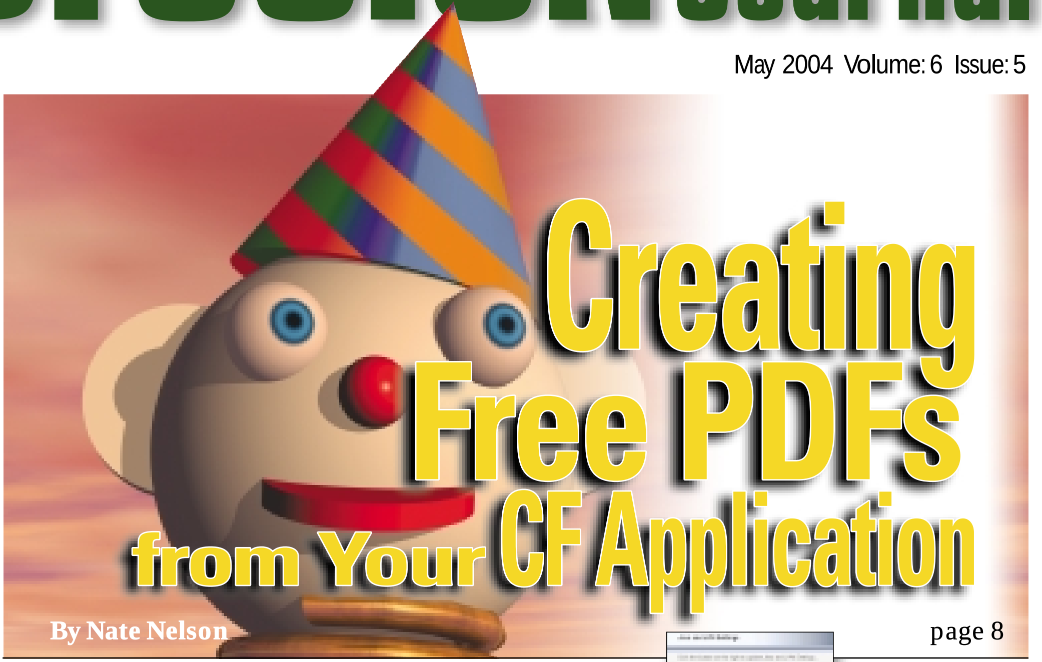
COLDFUSION MX AND .NET 101 PAGE 18

ColdFusion Developer's Journal

ColdFusionJournal.com

May 2004 Volume: 6 Issue: 5



SAVE UP TO **$400**
Subscribe today and get up to **3 FREE CDs!**
SEE PAGE 35 FOR DETAILS

Editorial
Spring Has Sprung!
Robert Diamond page 5

CF Community
Tales from the List
Simon Horwith page 7

Methods
Making the Most of J2EE Event Listeners
Eric Brancaccio page 26

CFUGs
ColdFusion User Groups
page 44

News
page 50

CFMX: ColdFusion MX and .NET 101 *COM Interop to a .NET assembly can be successfully implemented with CFMX* — *John Parrish & Jeffrey Bouley* 18

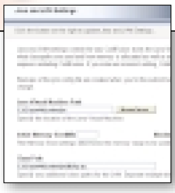
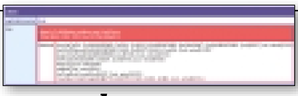
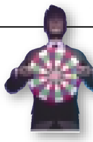
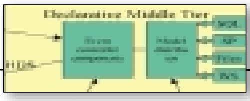
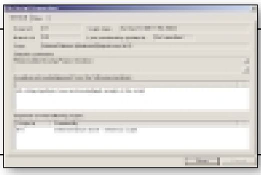
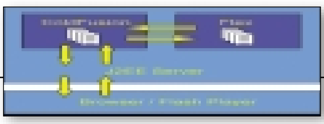
Foundations: Adventures in Encapsulation PART 4 *Taking on different roles* — *Hal Helms* 22

Frameworks: Model-View-Controller based Plug-able Declarative Framework *Simplify the development of scalable, maintainable, and robust Web applications* — *Pramod Jain & Yayati Kasralikar* 30

Product Review: Code Co-op Version Control Software from Reliable Software *Version control: We all need it* — *Selene Bainum* 36

<BF> on <CF>: On ColdFusion and Flex *What Flex can do for you* — *Ben Forta* 40

CF101: Structures and Arrays PART 1 *User arrays for processing numbers and spreadsheet-related functions* — *Jeffry Houser* 46

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*

31

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

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

THINK OF THEM AS FLIGHT INSTRUCTORS.

BECAUSE OUR CERTIFIED COLDFUSION INSTRUCTORS
WILL HELP YOUR CAREER TAKE OFF.

Learn the tools of the trade from the most talented and respected experts in the country. Our vendor-certified instructors are experienced professionals, well-versed in the latest techniques because they spend 75% of their time in the field working on real-world projects. Through an intensive hands-on lesson plan and expert knowledge transfer, you'll learn to harness the true power of ColdFusion.

FREE RESOURCE: Enjoy access to WinMill's On-Line Education Forum (OLEF), designed to simplify and speed the application of your newly acquired skills. It's like taking your instructor home with you. Ask us about it!

420 Lexington Avenue, Suite 444, New York, NY 10170

We are currently offering eight exciting courses in ColdFusion, Flash and Dreamweaver.

Call 1-888-711-MILL (6455) or email us at inquiry@winmill.com for our latest price specials and offers.
Go to www.winmill.com to view our Course Schedule.

Courses are held at our state of the art facility in New York City or on site per request.

We also offer vendor-certified training on the cutting-edge products of IBM, Check Point and Mercury Interactive.

WINMILL
SOFTWARE®
We make things happen!

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

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

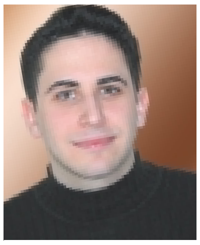
contributors to this issue

Selene Bainum, Jeffrey Bouley, Eric Brancaccio, Ben Forta, Hal Helms, Simon Horwith, Jeffry Houser, Pramod Jain, Yayati Kasralikar, Nate Nelson, John Parrish

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

Spring Has Sprung!



By Robert Diamond

Spring has finally sprung, even though you can't really tell from the odd weather that we've been having here in New York lately. Along with spring, there are a few tidbits of useful news to cover this month from Macromedia and elsewhere.

First up, there are a useful new hot fix for CFMX 6.1 that has sprung from Macromedia's development offices this month. Full details (and the download) are available at: www.macromedia.com/support/coldfusion/ts/documents/client_purge_hotfix.htm.

In a nutshell, MX runs hourly purge operations on client variables stored in the registry and data sources, and as the number of entries in the client store increases, the CPU usage needed to purge the entries increases as well. The hot fix helps these performance issues, and who doesn't like better performance?

Second is another useful hot fix which is the release of updated DataDirect JDBC drivers (3.3) for those keeping track. The new drivers fix a variety of server stability issues, including the 100% CPU utilization problem that some folks have been experiencing. Additionally, they also add database failover, and SQL Server Windows Authentication support, which some folks have been clamoring for. Full details for that one, along with the download, are available at www.macromedia.com/support/coldfusion/ts/documents/cfmx61_sqlserver_cpu.htm.

In live event news, MAX 2004, Macromedia's annual user conference covering ColdFusion, along with the rest of their product line, has been officially announced for November 1–4 in New Orleans, LA. You can follow the progress at www.macromedia.com/macromedia/events/max/ where more details will become available as they get 'em. Mark it on your calendar!

Speaking of conferences, I hope (and expect) to see a bunch of you before November, at the always well-organized CFUN '04 (www.cfconf.org/cfun-04). This event will take place June 26–27 in Maryland. They've got a great lineup as I've mentioned before (and I don't just say that because my name's in it too), with the who's who of the CF world speaking on a wide variety of useful topics. (The full schedule is available at www.cfconf.org/cfun-04/session.cfm).

You'll recognize many familiar faces from ***CFDJ*** there, including Ben Forta, Raymond Camden, Simon Horwith, Hal Helms, Charlie Arehart, Michael Dinowitz, and a slew more that you won't want to miss. It's a great way to tune up your knowledge and have some fun as well. We'll be doing our *CFDJ* Keynote Panel again this year, with the various *CFDJ* writers up on stage, ready to discuss anything and everything relating to CF. If you have any questions from "How do I..." to "Where is ColdFusion going this year?" e-mail them on over to me and we'll pose them to the panel. Hope to see you there!

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

BlueDragon [6.1]

RELEASED!

RUN YOUR CFML:

Within BlueDragon Server, or as a native .NET or J2EE web application.

On the platform, web server, and operating system of your choice.

New Atlanta COMMUNICATIONS

Get BlueDragon hosting:

CFDynamics is now offering BlueDragon hosting!
Find out more by visiting: www.cfdynamics.com/bluedragon

As one of the ColdFusion hosting community leaders, CFDynamics is constantly looking for ways to deliver new and cutting edge technologies to the ColdFusion community. We are excited to announce our premiere partnership with New Atlanta by offering BlueDragon hosting. We look forward to seeing how BlueDragon expands the possibilities of ColdFusion. Come join CFDynamics and New Atlanta in expanding the ColdFusion horizon!

POWERFUL HOSTING PLANS

· FREE SQL server access
· FREE account setup
· UNLIMITED email accounts
· GENEROUS disk space / data transfer
· 30 day MONEY-BACK GUARANTEE
· GREAT VALUE!

RELIABLE NETWORK

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

FANTASTIC SUPPORT SERVICES

· Comprehensive online support
· Knowledgeable phone support
· We focus on your individual needs

CFDynamics
A Division of Konnections Inc.

macromedia ALLIANCE PARTNER

BlueDragonCertified V.I.P. Hosting Partner

SIGN UP FOR A HOSTING ACCONT TODAY!

Use promo code: CFDJ04E and receive a FREE T-SHIRT!

WWW.CFDYNAMICS.COM

866 - CFDYNAMICS
866-233-962-6427

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

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
Christopher Croce chris@sys-con.com

online editor
Lin Goetz lin@sys-con.com

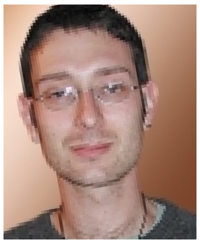
accounting
financial analyst
Joan LaRose joan@sys-con.com
accounts receivable
Charlotte Lopez charlotte@sys-con.com
accounts payable
Betty White betty@sys-con.com

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

Hello, stranger...

No, this month's title is not an attempt to meet ColdFusion developers of the opposite sex – it's all about building applications that are aware of the people using them.



By Simon Horwith

ColdFusion is a server-side programming language. Unlike client-side programming languages such as Flash, ColdFusion applications do not run stand-alone. They fit the "server" part of "client/server." In a Flash application, a developer might create business logic inside of a class file and associate that business logic with an event triggered when a user interacts with some UI element in the applications interface. There is a concept of events and listeners inherent in Flash.

In a ColdFusion application, the developer writes business logic but that business logic then waits to respond to any request sent by a client such as Flash, SOAP, or a Web browser. Though there are frameworks that mimic event-driven interaction between components, the bottom line is that ColdFusion developers do not write business logic with the idea that users will directly interact with that logic.

All of this leads to the point that ColdFusion developers don't think about the user of their application in the same way that developers in other environments do. Our applications are detached from the user – and many ColdFusion developers don't have a strong grasp of what a user is and how they can create applications that are more "user aware."

One of the recurring categories of *CFDJ-List* thread topics is how to track users, restrict their access to resources, and personalize the application (user) interface based on who is using the application. This month I will give an overview of these topics and the common suggested approaches to implementing them in a ColdFusion application.

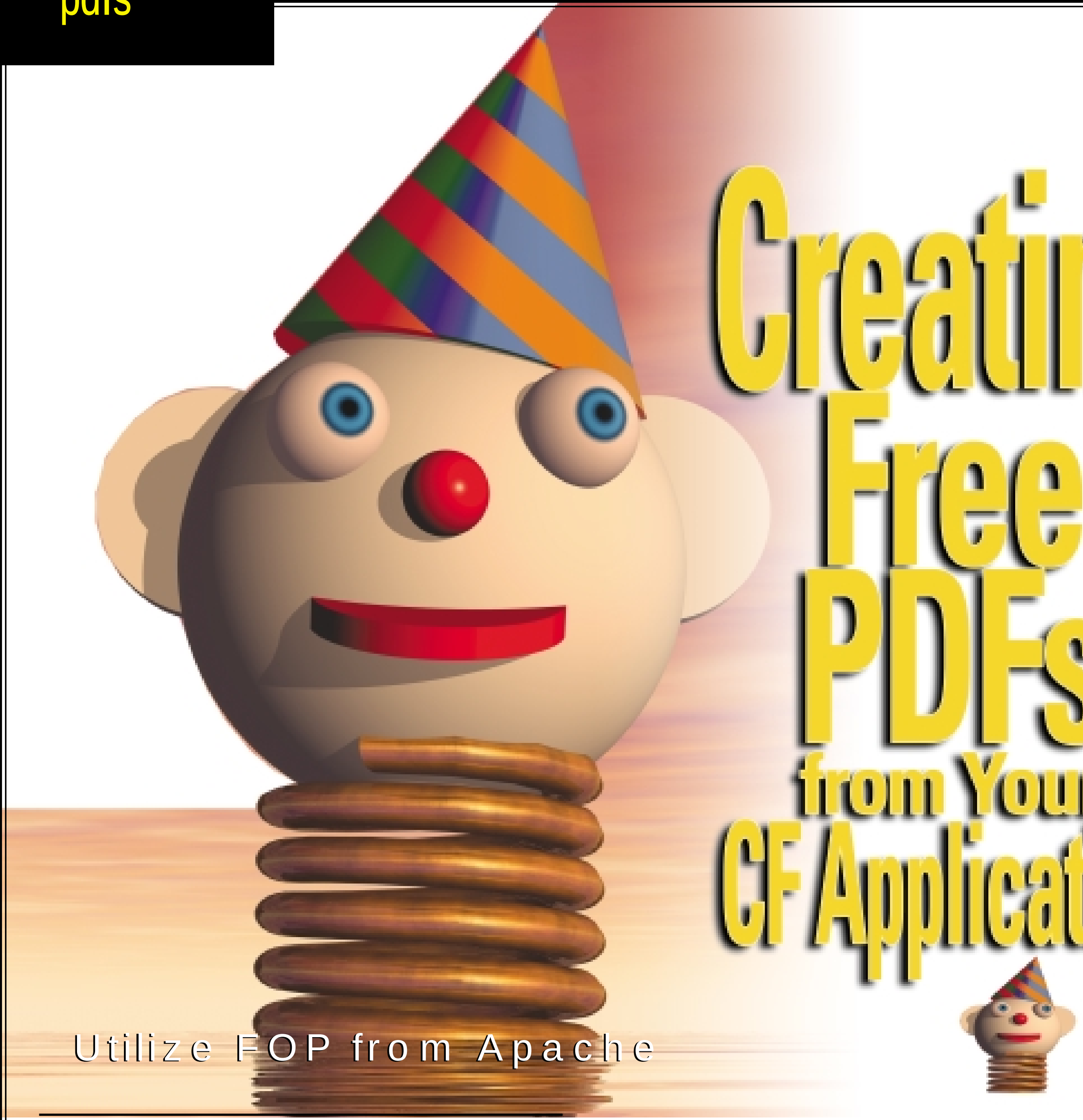
The first thing to understand is that the information that uniquely describes a user belongs in a place in memory that is unique to each user. This could be a unique row in a database or text (or XML) file, cookies, the session scope, or the client scope. Because data is read quicker from memory than from a file or database, it is not generally considered a best practice to retrieve user-specific data from a database or file on every request, though it certainly might be retrieved from an external resource when a user logs in to the application. Cookies aren't usually the ideal place to store this information because the user can disable them in his or her browser; they cannot store complex data; they have size limitations; and they aren't secure. By process of elimination this leaves the session scope, which is generally considered to be the best place to store information about a user.

There are, however, three caveats to keep in mind when storing session-specific information in the session scope. The first is that if the application is going to use J2EE sessions and will run in a cluster of session buddies, you cannot store nonserializable object instances in the session scope. Most notably this includes instances of ColdFusion Components. The second caveat to keep in mind is that ColdFusion still requires a token to identify which session belongs to which browser, so the CFID/CFTOKEN (or JSESSIONID) must be passed on the URL or in form posts from page to page.

— continued on page 38

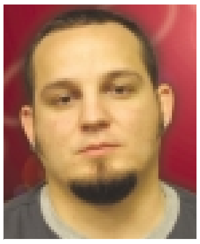
About the Author

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 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



Creating Free PDFs from Your CF Application

Utilize FOP from Apache


By Nate Nelson

I'm sure most of you have been in a situation where either your employer or customer doesn't like the way content from a Web browser looks when printed. There are many ways to provide printable content, but one of the most popular formats of printable material is PDF format. There are many solutions to creating PDF documents from your ColdFusion application, but they all cost money right?

Some PDF solutions are not cheap at all and maybe cost isn't the only issue. Maybe you want to do it yourself without using a CFX tag. What if I told you there was a way to create PDFs from CF for free without using a CFX tag? Would you be interested? If so, then read on…

In this article I will demonstrate how to create PDFs from your ColdFusion application for free. I will be utilizing FOP (Formatting Objects Processor) from Apache, which is of course free. This will integrate well with ColdFusion because it can be invoked as a Java object.

What Is FOP?

FOP is part of Apache's XML project. The latest version of

FOP (0.20.5) was released on July 18, 2003. It is a partial implementation of the XSL-FO Version 1.0 W3C Recommendation. Support for each of the standard's objects and properties are detailed at http://xml.apache.org/fop/compliance.html. FOP is the world's first print formatter driven by XSL formatting objects (XSL-FO) and the world's first output independent formatter. It is a Java application that reads a formatting object tree and renders the resulting pages to a specified output. PDF is the primary output target, but many others are also supported, listed at http://xml.apache.org/fop/output.html. The PDF version supported is 1.3, which is currently the most popular version for Acrobat Reader (4.0); PDF versions are forwards/backwards compatible. For more information about FOP visit http://xml.apache.org/fop.

The Quick Shakedown

Just so you know what you're getting into before reading on, here's a quick shakedown of what I will explain in detail through the rest of this article. The main points in this process are:

- FOP is easily installed on a CF server.
- CF generates an XML file.
- XML file is transformed by using XSL-FO.
- CF invokes an FOP component to dynamically generate a PDF.
- FOP component reads transformed XML.
- FOP component converts transformed XML into a PDF format by using Java objects.
- Browser returns a PDF document.

Installing FOP

Installing FOP is very easy and works like a charm as long as you have all the necessary components in place and of course, pay attention. Here are the detailed steps:

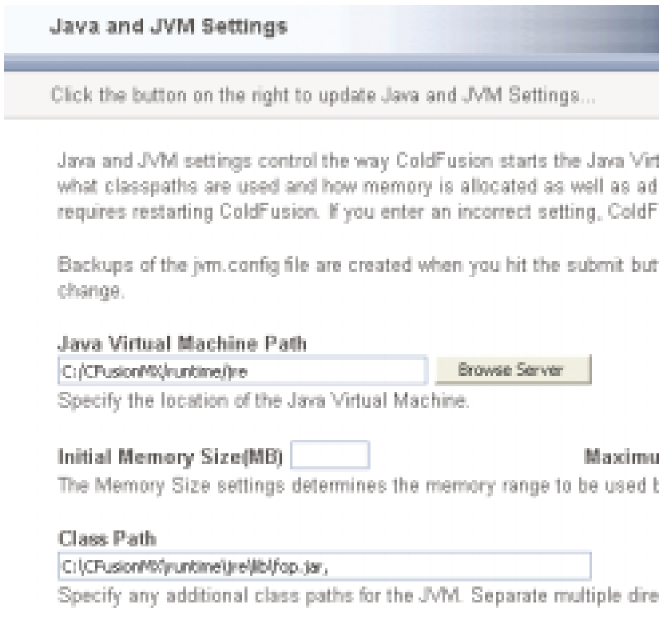
1. In order to use the provided code it is required that you have ColdFusion MX installed.
2. Since FOP is a Java application, you'll need an installed Java 2 SDK, version 1.2 or later. The latest Sun J2SE downloads can be found at http://java.sun.com/j2se/downloads/index.html. At the top you will see links to the latest J2SEs. Currently the latest stable J2SE is 1.4.2. Under the heading "Download J2SE v 1.4.2_04" find the corresponding download for your OS and then click "Download". If you need further instructions go to http://java.sun.com/j2se/1.4.2/install.html.
3. Next on the list is to download the latest binary build of FOP (0.20.5) from Apache. Go to http://apache.secsup.org/dist/xml/fop/ and click on fop-0.20.5-bin.tar.gz to download. Download the file and extract it to the directory of your choice. After the file has been extracted you need to copy the following three .jar files to your CFMX jre\lib directory; a common location is C:\CFusionMX\runtime\jre\lib.
   - <fop-extracted-dir>\build\fop.jar
   - <fop-extracted-dir>\lib\avalon-framework-cvs-20020806.jar
   - <fop-extracted-dir>\lib\batik.jar
4. After copying those files to your CFMX jre\lib directory you will then be required to add the fop.jar file to the classpath in your CF Admin. Using the common location I listed above you would add C:\CFusionMX\runtime\jre\lib\fop.jar to your classpath as shown in Figure 1. If your classpath is not empty, then separate the entries by a comma. You will then be required to restart your ColdFusion MX Application Server. If you do not know how to do this read step 5, otherwise you are now ready to use FOP! Step 6 is only important for using the code provided with this article.
5. To restart the ColdFusion MX Application Server right-click on "My Computer" and choose "Manage". Then expand "Services and Applications", click on "Services" and you should see the CFMX Application Service listed. If so, then right-click and choose "Restart". If this doesn't work for you then you will need to visit your OS help.
6. This step is only important for using the code provided with this article. Create a directory under your wwwroot to store the provided code. I will demonstrate the creation of all the necessary files except for the FOP component. I downloaded FOP.cfc from Nate Weiss at http://nateweiss.com/ and I have included the code in Listing 7.

That wasn't so bad now was it?

The Example

I will now continue by introducing the example that I will be using through the rest of the article. I have chosen an example that will help demonstrate the capabilities of FOP as well as be simple enough to not take over the article. I have created a very simple two-table database that is by no means normalized, which I do not recommend, but that will work for what I am using it for. I created a SQL database named "FOPExample" and created it as a data source in my CF Admin, named "FOPExample" as shown in Figure 2. I created one table to contain the 50 United States and another table to contain customers with their corresponding city and state. I have provided the SQL scripts to create these tables (see Listing 1).

I will start with a basic form that contains a multi-select list


Java and JVM Settings

Click the button on the right to update Java and JVM Settings...

Java and JVM settings control the way ColdFusion starts the Java Virt
what classpaths are used and how memory is allocated as well as ad
requires restarting ColdFusion. If you enter an incorrect setting, ColdF

Backups of the jvm.config file are created when you hit the submit but
change.

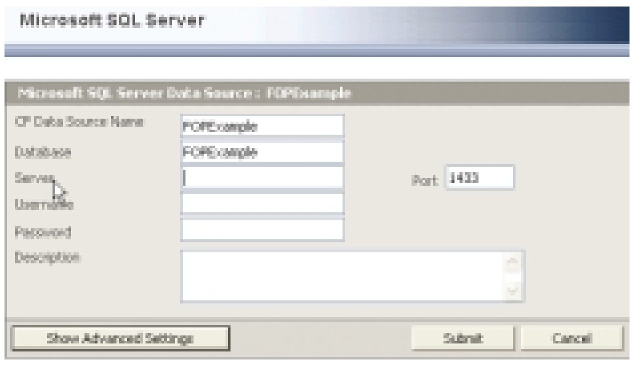
Java Virtual Machine Path
C:/CFusionMX/runtime/jre
Specify the location of the Java Virtual Machine.

Initial Memory Size(MB)
The Memory Size settings determines the memory range to be used t

Class Path
C:\CFusionMX\runtime\jre\lib\fop.jar,
Specify any additional class paths for the JVM. Separate multiple dire

Figure 1: Adding fop.jar to the ColdFusion MX classpath

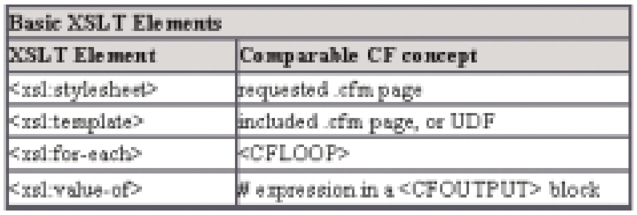
of the 50 states (see Listing 2). This basic form will submit to generatexml.cfm (see Listing 3), a file I have created that will read the example database to fetch a list of the customers located in the chosen state(s). Then I will output these customers into an XML file. The PDF could be automatically created next, but to show a step-by-step process I provide a link to generate the PDF. After clicking the link, the FOP component will be invoked to create the PDF file. To show the capabilities of XSL-FO I have designed this to group by state on the PDF output. Sounds pretty cool, right? If I haven't bored you to death yet, then please continue as I will now explain the step-by-step process in more detail.



Microsoft SQL Server Data Source : FOPExample

CF Data Source Name: FOPExample
Database: FOPExample
Server:
Port: 1433
Username:
Password:
Description:

Figure 2: Creating a data source in ColdFusion MX



Basic XSLT Elements	
XSLT Element	Comparable CF concept
<xsl:stylesheet>	requested .cfm page
<xsl:template>	included .cfm page, or UDF
<xsl:for-each>	<CFLOOP>
<xsl:value-of>	# expression in a <CFOUTPUT> block

Figure 3: Basic XSLT elements explained in CF

Generating the XML

Now the fun begins as I will demonstrate the dynamic creation of the XML file to be transformed into PDF content as shown in Listing 3. In this file I query the FOPExample database to get the customers and their info for those that are in the states chosen from the form. In this case I chose California, Colorado, and Minnesota. To easily create a properly formatted XML string I use the <cfxml> tag by wrapping it around the XML content that I will generate. I have used only XML in this output because all of the display formatting will come from the XSL file in the XML transformation.

The first XML tag I will create in this file is the very top level output, in this case <project>. If you are familiar with the way XML tags work you will notice I have named the project "FOPExample". I was a little lazy and "hardcoded" this value, but it could be dynamic if you had multiple projects. I will display the project line as a header in the PDF file.

To generate the rest of the XML content I use a simple <cfoutput> and take advantage of the group attribute in order to group the XML by state. After outputting the state I then move on to customers by using <customers> as a wrapper with <person> inside. Wrapped in the <person> tag I output the names of each person and all of the other attributes of that person are added by adding a corresponding attribute to the <person> tag. Take note of these attribute names and tag names because I will refer to them when creating the XSL file.

After generating my XML string I input it into the <cffile> tag to save it as an XML file on the server. The FOP process could be done without actually saving the XML file, but I have done so to easily show the step-by-step process. Saving the XML file is not recommended if you are generating sensitive data. If you are not generating sensitive data, this could be an advantage if you are generating a very large "one time" report because you won't have to hit the database again as you can just render the XML file. After generating the XML file I suggest opening it in your favorite text editor to see how it is formatted (see Listing 4) and to make sure it is formatted the way you expected. Next I use a simple HTML link to fire off the PDF generation process. Before actually firing off the process we are still missing one very important piece – the XSL file.

Creating the XSL file

FOP generates a PDF file by reading transformed XML code. In order to provide this we must now create an XSL-FO file. Open your favorite editor that hopefully supports XSL code. If you do not have one don't worry, it's just easier with one. I actually prefer to use TextPad, which you can try for free and which is very cheap to buy. TextPad has syntax file plugins for quite a few languages and runs very light on the processor. You can download TextPad at www.textpad.com and you can get the syntax files at www.textpad.com/add-ons/files/utilities/synfile.zip.

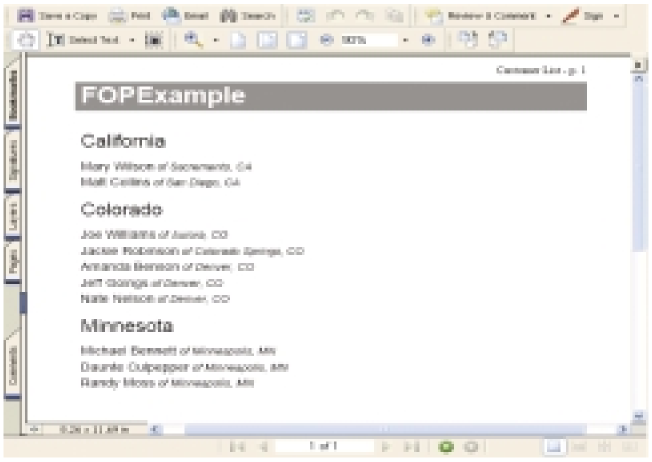
Now we are really cruising as we will dip into FOPExample.xsl. In order to have a correctly formatted XSL-FO document it must contain all of the following (except for my comment of where the content goes).

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:fo="http://www.w3.org/1999/XSL/Format"
   version="1.0">

   <!—content related code goes here -->

</xsl:stylesheet>
```

Now for the content. I will start by giving a few simple comparisons that really helped me understand the basic XSLT elements (see Figure 3).

I will start the content with <xsl:template match="/project"> as you may notice the value to the match attribute equals the name of the top level XML tag I used when generating the XML output. Next comes another required tag for XSL-FO followed by the page layout properties of the template, commented as "defines page layout" in Listing 5.

I'm not going to actually go into detail about each of these attributes, but you can see that it is very customizable. The


Customer List - p. 1

FOPExample

California

Mary Wilson of Sacramento, CA
Matt Collins of San Diego, CA

Colorado

Joe Williams of Aurora, CO
Jackie Robinson of Colorado Springs, CO
Amanda Benson of Denver, CO
Jeff Gorings of Denver, CO
Nate Nelson of Denver, CO

Minnesota

Michael Bennett of Minneapolis, MN
Daunte Culpepper of Minneapolis, MN
Randy Moss of Minneapolis, MN

Figure 4: Finally, the results in PDF

next tag used is for the page numbering that you display in the PDF: <fo:page-sequence master-reference="only" initial-page-number="1">. After this tag you will see a static content <fo:block> commented as "Header" in Listing 5.

If you look at the numerous <fo:block> tags, you may notice these are at each level of content and are used for formatting. Wrapped inside the beginning and end tags can be actual text, or if you want to output a variable like I have done in this example you would use: <xsl:value-of select="@variable"/>. Variable is actually the name of the attribute to the XML tag in the generated XML output. You will also see the tag: <xsl:for-each select="usstate">. This is used similarly to the way you would use <cfloop> while looping over each state, and inside that loop you are going to display the related customers.

So is the XSL-FO effort making sense now? You can see the entire FOPExample.xsl in Listing 5. I commented the important pieces to help you understand where everything is. If you need more specific help with the XSL file see http://zvon.org/xxl/xslfoReference/Output/index.html.

Transforming the XML

Now that I have my XML and XSL files I can transform my XML to prepare it for FOP. This process is so much easier than it sounds thanks to the XML functions in ColdFusion. To do this I will use the XMLTransform() function as shown in Listing 6. I will use the <cffile> tag to read the contents of both my XML and XSL files. I will then pass the output from both of these files into the XMLTransform function. The XMLTransform function asks for my XML string, then my XSL string. I will set the transformed outout to a variable. Now you have a transformed XML string to pass into the FOP component.

Invoking FOP

Now that the hard part is done we will invoke the FOP component that I downloaded from Nate Weiss. This is done by using the <cfinvoke> tag. I will invoke the FOP component and I will be using the ConvertStringToPDF method. In order to use this method it is required to pass in values to two arguments. FoString, which takes the transformed XML string and pdfFile, which will be used to give the name and location for the PDF file to be created. The other method available in this component is ConvertFileToPDF, which works the same way but instead of passing the transformed XML string into it, you will give the name and location of a file that contains the transformed XML.

If you look at the code for the FOP component in Listing 7 you may notice another method that is actually used internally to generate the PDF. This method uses proper locking procedures since FOP is not thread safe. Because it is not thread safe a new instance of the Apache driver must be created each time it is used. Thanks to an already provided component this process was simple too. We are just moving right along now. Next you will actually see the results that you have been waiting for.

Displaying the PDF

In order to display the PDF file in the browser I will use the <cfcontent> tag. This is once again a very simple process as I pass into it the name and full path to the generated PDF file. The other attribute I must use for this tag is the type attribute, which I will set to "application/pdf" for the PDF file format. Now that the code is all explained and created it's time to click that link that we added to the bottom of generatexml.cfm in Listing 3. A new window should pop up with your dynamically generated content. The content should look like it does in Figure 4. If yours didn't work, then hopefully there was just a simple step missed. Remember this process can be very simple, but it is picky and it won't work at all if something is missing.

Conclusion

Well, now that you know FOP is not scary to use, you will be on your way to creating PDFs for free. I only touched on the essentials of FOP in this article. There are many other capabilities such as PDF encryption, SVG support, and including a linked Table of Contents. The example that I demonstrated was a very simple task; there are many other possibilities this could be used for. Look forward to Part 2 of this article as I will explore some of the other FOP capabilities. As a fellow developer, I enjoy informing other developers that there are many possibilities when it comes to finding solutions to your projects. I know it seems like this is difficult and takes a long time to learn, but once you try it for yourself you will realize that it's actually fairly easy. Good luck!

Resources

- *The FOP project* http://xml.apache.org/fop/
- *Pawson, Dave: XSL-FO Making XML Look Good in Print (O'REILLY) ISBN: 0-596-00355-2*
- *An XSL-FO reference* http://zvon.org/xxl/xslfoReference/Output/index.html

About the Author

*Nate Nelson is a senior software developer for I*LEVEL, Inc., a software startup in Englewood, Colorado, where he leads development efforts of a commercial software product. He is a member of the Denver CFUG administration.*

natenelson@comcast.net

(Code listing begin on page 14)

http://www.gulfstream.com/

When your product delivers at the highest level, appearances count: the Gulfstream website. Built with Dreamweaver. Which, in its latest update, is up to 70% faster. Hey, everyone likes to fly.

LISTING 1

FOPExampleTables.sql

```
CREATE TABLE [dbo].[customers] (
  [customerid] [int] IDENTITY (1, 1) NOT NULL ,
  [customerfirstname] [varchar] (50) NOT NULL ,
  [customerlastname] [varchar] (50) NOT NULL ,
  [customercity] [varchar] (50) NOT NULL ,
  [customerstatecode] [varchar] (2) NOT NULL
) ON [PRIMARY]
GO
CREATE TABLE [dbo].[usstates] (
  [usstateid] [int] IDENTITY (1, 1) NOT NULL ,
  [usstatecode] [varchar] (2) NOT NULL ,
  [usstatename] [varchar] (25) NOT NULL
) ON [PRIMARY]
GO
```

LISTING 2

basicForm.cfm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head><title>Basic Form</title></head>
<body>
<cfquery name="getstates" datasource="FOPExample">
  select usstatecode, usstatename
  from usstates
  order by usstatename
</cfquery>
<form action="generatexml.cfm" method="post">
Select 1 or more states:<br>
<select name="usstates" multiple>
  <cfoutput query="getstates">
  <option value="'#getstates.usstatecode#'">#getstates.usstatename#
  </cfoutput>
</select>
<input type="submit" value="submit">
</form>
</body>
</html>
```

LISTING 3

generateXML.cfm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head><title>Generate the XML</title></head>
<body>

<cfquery name="getcustomers" datasource="FOPExample">
  select c.customerfirstname, c.customerlastname, c.customercity, c.cus-
tomerstatecode, u.usstatename
  from customers as c
  inner join usstates as u
        on c.customerstatecode = u.usstatecode
  where u.usstateid IN (#FORM.usstates#)
  order by u.usstatename, c.customercity, c.customerlastname
</cfquery>

<!--- Create an XML string named MyXML --->
<cfxml variable="MyXml">
  <project name="FOPExample">
    <cfoutput query="getcustomers" group="usstatename">
      <usstate name="#getcustomers.usstatename#">
        <customers>
          <cfoutput>
<person city="#getcustomers.customercity#" statecode="#getcustomers.cus-
tomerstatecode#">#getcustomers.customerfirstname# #getcustomers.customer-
lastname#</person>
          </cfoutput>
        </customers>
      </usstate>
    </cfoutput>
  </project>
</cfxml>

<cffile action="WRITE" file="#ExpandPath('Output.xml')#"
output="#ToString(MyXML)#">
<p> The XML has been generated.</p>
<a href="TransformXML.cfm" target="_blank">View Generated PDF</a>

</body>
</html>
```

LISTING 4

Output.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="FOPExample">
   <usstate name="California">
     <customers>
         <person city="Sacremento" statecode="CA">Mary Wilson</person>
         <person city="San Diego" statecode="CA">Matt Collins</person>
     </customers>
   </usstate>
   <usstate name="Colorado">
     <customers>
         <person city="Aurora" statecode="CO">Joe Williams</person>
         <person city="Colorado Springs" statecode="CO">Jackie
Robinson</person>
         <person city="Denver" statecode="CO">Amanda Benson</person>
         <person city="Denver" statecode="CO">Jeff Goings</person>
         <person city="Denver" statecode="CO">Nate Nelson</person>
     </customers>
   </usstate>
   <usstate name="Minnesota">
     <customers>
     <person city="Minneapolis" statecode="MN">Michael Bennett</person>
         <person city="Minneapolis" statecode="MN">Daunte
Culpepper</person>
         <person city="Minneapolis" statecode="MN">Randy Moss</person>
     </customers>
   </usstate>
</project>
```

LISTING 5

fopExample.xsl

```
<?xml version="1.0" encoding="utf-8"?>

<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:fo="http://www.w3.org/1999/XSL/Format"
   version="1.0">

   <xsl:template match="/project">
      <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
       <!-- defines page layout -->
         <fo:layout-master-set>
            <!-- layout for the first page -->
                  <fo:simple-page-master
                     master-name="only"
                     page-height="29.7cm"
                     page-width="21cm"
                     margin-top="1cm"
                     margin-bottom="2cm"
                     margin-left="2.5cm"
```

Think .NET development is more productive than J2EE?

Think **again**.

Due to delivery pressures on our last project,
we thought about moving to .NET.
I suggested we stick with Java, but use ThinkCAP.

Think **better**.

ThinkCAP™

ClearNova's ThinkCAP is a comprehensive application platform that simplifies and accelerates the development and maintenance of J2EEE-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

```
                        margin-right="2.5cm">
                        <fo:region-body margin-top="3cm"/>
                        <fo:region-before extent="3cm"/>
                        <fo:region-after extent="1.5cm"/>
                     </fo:simple-page-master>
          </fo:layout-master-set>
          <!-- end: defines page layout -->

<!-- actual layout -->
<fo:page-sequence master-reference="only" initial-page-number="1">

                     <!-- header -->
                     <fo:static-content flow-name="xsl-region-before">
                                   <fo:block
                           text-align="end"
                                 font-size="10pt"
                                 font-family="serif"
                                 line-height="14pt" >
                              Customer List - p. <fo:page-number/>
                        </fo:block>
                     </fo:static-content>

  <fo:flow flow-name="xsl-region-body">
         <!-- defines text title level 1-->
                      <fo:block
                           font-size="24pt"
                                 font-family="sans-serif"
                                 line-height="30pt"
                                 space-after.optimum="15pt"
                                 background-color="grey"
                                 color="white"
                                 font-weight="bold"
                                 padding-left="5px"
                                 padding-top="2px">
                                          <!-- output level 1 name -->
                                    <xsl:value-of select="@name"/>
                        </fo:block>

  <xsl:for-each select="usstate">
  <!-- defines text title level 2-->
          <fo:block
                     font-size="18pt"
                     font-family="sans-serif"
                     line-height="20pt"
                     space-before.optimum="10pt"
                     space-after.optimum="10pt"
                     text-align="start"
                     padding-top="0pt">
<!-- output level 2 name -->
<xsl:value-of select="@name"/>
         </fo:block>

<xsl:for-each select="customers/person">
<!-- defines the final text level -->
                     <fo:block
                     font-size="12pt"
                     font-family="sans-serif"
  line-height="15pt"
  space-after.optimum="3pt"
  text-align="start">

<!-- output the contents wrapped in final level -->
  <xsl:value-of select="."/>
  <!-- inline formatting -->
  <fo:inline font-style="italic" font-size="10pt">
  of <xsl:value-of select="@city"/>, <xsl:value-of select="@statecode"/>
                     </fo:inline>
                   </fo:block>
                  </xsl:for-each>
                          </xsl:for-each>
                        </fo:flow>
                     </fo:page-sequence>
                 </fo:root>
              </xsl:template>
</xsl:stylesheet>
```

LISTING 6

transformXML.cfm

```
<!--- turn off debugging to prevent errors when displaying pdf content -
-->
<cfsetting showdebugoutput="No">

<!--- set filenames --->
<CFSET MyStylesheet = "FOPExample.xsl">
<cfset pdfFile = "Result.pdf">
<cfset MyXml = "Output.xml">

<!--- get contents of xml file and xsl file --->
<CFFILE ACTION="READ" FILE="#ExpandPath(MyXml)#"
VARIABLE="XMLFileContent">
<CFFILE ACTION="READ" FILE="#ExpandPath(MyStylesheet)#"
VARIABLE="XSLFileContent">

<!--- perform xml transformation with XMLFileContent agaisnt
XSLFileContent --->
<CFSET TransformedXmlCode = XmlTransform(XMLFileContent, XSLFileContent)>

<!--- now invoke the FOP component - pass in transformed xml and pdf
file to be created --->
<cfinvoke component="FOP" method="ConvertStringToPDF"
foString="#TransformedXMLCode#" pdfFile="#ExpandPath(pdfFile)#">

<!--- No display PDF file to browser --->
<cfcontent file="#ExpandPath(pdfFile)#" deletefile="yes" type="applica-
tion/pdf">
```

LISTING 7

fop,cfc

```
<cfcomponent>
  <!--- Initialize the FOP driver from xml.apache.org --->
  <cfset THIS.driver = CreateObject("java",
"org.apache.fop.apps.Driver")>
  <cfset THIS.lockname = CreateUUID()>

  <cffunction name="ConvertFileToPDF" access="public">
    <cfargument name="foFile" type="string" required="true">
    <cfargument name="pdfFile" type="string" required="true">

    <!--- Set up Java input source --->
         <cfset var input = CreateObject("java",
"org.xml.sax.InputSource")>
         <cfset input.init( ARGUMENTS.foFile)>

    <!--- Proceed with PDF generation --->
    <cfset private_generatePDF(input, pdfFile)>
  </cffunction>

  <cffunction name="ConvertStringToPDF" access="public">
    <cfargument name="foString" type="string" required="true">
    <cfargument name="pdfFile" type="string" required="true">
  <!-- set arguments values -->

  <!--- Set up Java input source --->
  <cfset var reader = CreateObject("java", "java.io.StringReader")>
  <cfset var input = CreateObject("java", "org.xml.sax.InputSource")>
         <cfset reader.init(foString)>
```

HostMySite.com™

Built for ColdFusion Pros by ColdFusion Pros

- 24 / 7 / 365 Phone Support
- 99.9+% Uptime
- Macromedia Alliance Partner
- "Full Control" Panel
- CFMX 6.1 or CF 5.0
- SQL Server 2000 or 7.0
- Custom Tags Welcome

plans from $8.95 / mo.

FREE Domain Name*
FREE Setup
FREE 2 Months

Visit www.HostMySite.com/cfdj for:
2 Months Free
FREE Setup and FREE Domain Name
*on any annual shared hosting plan

call today 877•248•HOST (4678)

pdfs

```
          <cfset input.init(reader)>

     <!--- Proceed with PDF generation --->
     <cfset private_generatePDF(input, pdfFile)>
   </cffunction>

   <cffunction name="private_generatePDF" access="private">
     <cfargument name="input" type="any" required="true">
     <cfargument name="pdfFile" type="string" required="true">

   <!--- Output stream for writing PDF file --->
   <cfset var output = CreateObject("java", "java.io.FileOutputStream")>

   <!--- Get ready to do the PDF generation --->
   <cflock name="#THIS.lockname#" type="exclusive" timeout="10">
   <cflock name="#ARGUMENTS.pdfFile#" type="exclusive" timeout="10">
     <!--- Turn on the output stream --->
          <cfset output.init( ARGUMENTS.pdfFile )>
   <!--- Hook the FOP driver to the input/output --->
          <cfset THIS.driver.setInputSource(input)>
          <cfset THIS.driver.setOutputStream(output)>
          <!--- Perform the actual PDF generation --->
                   <cfset THIS.driver.run()>
         <!--- Turn off the output stream --->
          <cfset output.close()>
          </cflock>
   </cflock>
   </cffunction>
</cfcomponent>
```

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

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

Java for ColdFusion Programmers?

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

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

ColdFusion MX and .NET 101

COM Interop to a .NET assembly can be successfully implemented with CFMX

By John Parrish

By Jeffrey Bouley

Some of you may be thinking, ColdFusion (CF) and .NET? Why do it? We hear you. It is understandable that CF, running on top of J2EE, adds a layer of complexity when integrating with interfaces written with Microsoft-centric technologies. The mission of this article is to decipher the complexities and shed some light on how CF can be coupled with .NET solutions to allow for interoperability throughout your IT universe.

.NET is breaking in big in the IT arena and many organizations are adopting it as a standard for developing the business modules associated with their applications. A benefit that .NET brings with it is a true object-oriented methodology to developing software. I liken the C# language as the fraternal twin to Java; though with these benefits, it also brings an increased level of complexity for doing routine tasks that ColdFusion programmers have grown to take for granted.

You can't knock Joe programmer for wanting to learn a new technology. This outlook is preached continually from both Macromedia and Microsoft. With constant new releases and their advantages, the standouts must increase their understanding in all facets of the industry to stay a millimeter ahead of the curve. "What's that you say?" "You're not certified in CFMX…?" "Wait, what?" "You're not working on your MCAD/MCSD?" "Well, what are you waiting for?" Enough with our attempts at lighting a fire under you to expand your horizons, you're reading this article and we commend you for it.

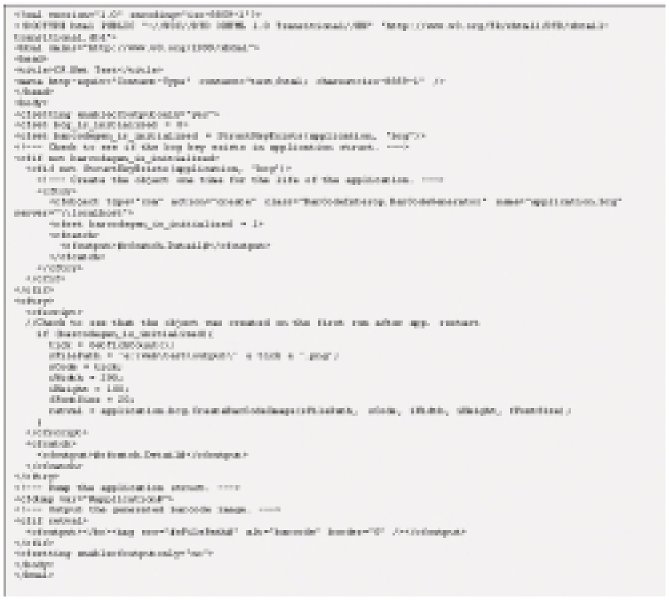
With the J2EE release of ColdFusion MX (CFMX) and the new version 6.1, code-named "Red Sky," Macromedia has continued to support COM (Component Object Model)-based solutions via a COM bridge. CFMX uses a Java-to-COM wrapper that was developed by Intrinsyc. The package is known as J-Integra and details on this component can be found at http://ja.net.intrinsyc.com/. Their product info area has some excellent examples on how to implement J-Integra with various combinations of Java/COM components. J-Integra is instantiated through the cfobject tag or with cfscript utilizing the function CreateObject.

In our example in Figure 1, the cfobject tag was used. Notice the class that is called and the name we assign to the object. Once the object is successfully created, the CreateBarCodeImage method is accessed and the appropriate required parameters are passed to the component. The resulting page output is seen in Figure 2. The application structure is dumped, displaying the contents of the applicationname key and bcg (barcodgen) key. Notice how the object information and available related methods are displayed. CreateBarCodeImage is the method used to generate the barcode image seen in the output.

The ColdFusion code used for the example is an enhanced methodology for instantiating a COM component with ColdFusion. This enhanced methodology outlined in the following Macromedia TechNote, www.macromedia.com/support/coldfusion/ts/documents/tn18210.htm, allows the initial creation of the object to be stored in memory via application or session variables. This also enables the object to be shared across requests and easily accessed with a custom tag or cfmodule. This solution came about due to the observation that there was an increase in creation time in CFMX compared to CF 5. For the sake of argument, this technique is being demonstrated only to provide information in case you come across this issue. No performance issues were witnessed with the .NET/COM component utilized in this article.

So you can create an instance of a COM object in CFMX; ColdFusion has always made this an easy task. This gets interesting when you look at how this is actually achieved. Let's peek under the hood, shall we? Figure 3 shows how CFMX and J-Integra hook into various automation components. There is a similar diagram at http://ja.net.intrinsyc.com/j-integra/info/.

Now for a little light on what might be happening behind the scenes when you call into a .NET object through COM. Just to get the high-level talk out of the way, all .NET languages are compiled into an intermediate language called Microsoft Intermediate



```
[illegible]
```

Figure 1: ColdFusion code used to instantiate the .NET assembly and display output

```
[illegible]
```

Figure 2: Output generated from .NET assembly

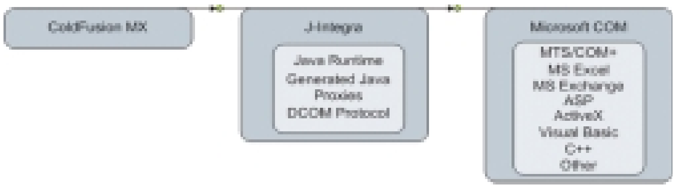


Figure 3: Path taken by ColdFusion when calling into COM

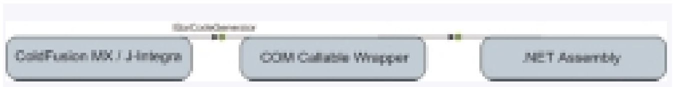
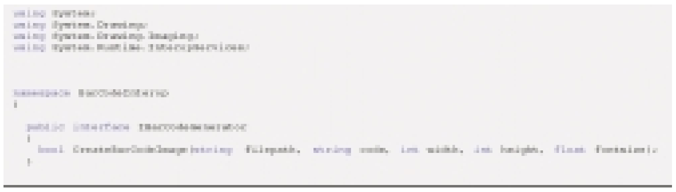


Figure 4: Path highlighting the interaction of the COM callable wrapper



```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;


namespace [illegible]
{

  public interface [illegible]
  {
    bool CreateBarCodeImage(string filepath, string code, int width, int height, float fontsize);
  }
```

Figure 5: .NET assembly's public interface

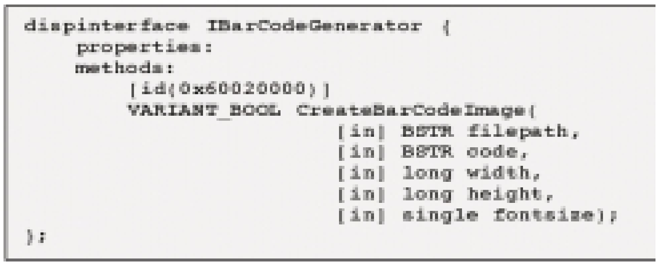
Language, also known as MSIL. Like Java, MSIL runs in a virtual machine, called the Common Language Runtime (CLR).

For those who have not dealt with COM before, it provides a binary specification allowing interaction between languages. COM also provides facilities for dynamic discovery of interfaces that an object has exposed.

COM was designed to provide a standard means of marshalling between languages; both platforms must perform some degree of marshalling between types in the respective runtimes and the facilities for accessing them specified by COM. The CLR provides a transparent proxy allowing .NET assemblies to be called from COM clients, called the COM Callable Wrapper (CCW). It is through the CCW that ColdFusion is able to indirectly interface with a .NET assembly (see Figure 4).

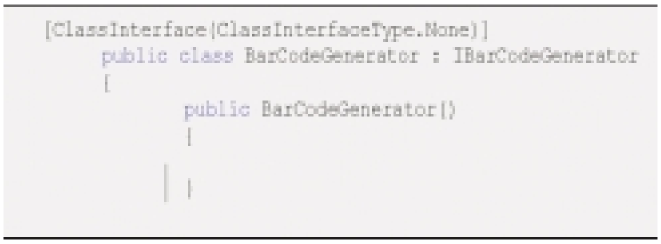
For this example we have chosen to implement a very simple component in C#, which when given a series of parameters, will generate an image of a barcode for use within a Web page. The component happily creates images of barcodes on request; it is not the job of this component to manage the lifespan of the files it creates. That task is better left to the consuming application (CFMX).

There are only a few steps needed to have a .NET component ready to be consumed directly from CFMX. First, the code must be written. Second, the assembly must be signed with a strong name



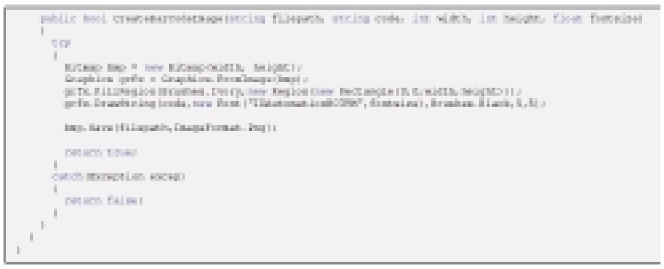
```
dispinterface IBarCodeGenerator {
    properties:
    methods:
        [id(0x60020000)]
        VARIANT_BOOL CreateBarCodeImage(
                        [in] BSTR filepath,
                        [in] BSTR code,
                        [in] long width,
                        [in] long height,
                        [in] single fontsize);
};
```

Figure 6: .NET assembly's exposed IDispatch interface



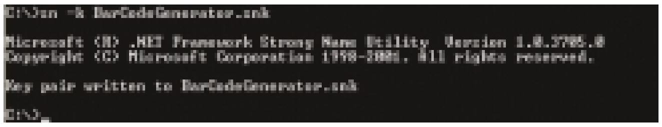
```
[ClassInterface(ClassInterfaceType.None)]
    public class BarCodeGenerator : IBarCodeGenerator
    {
            public BarCodeGenerator()
            {

            }
```

Figure 7: Class interface declaration exposing the .NET assembly to COM (notice class interface type is set to None)



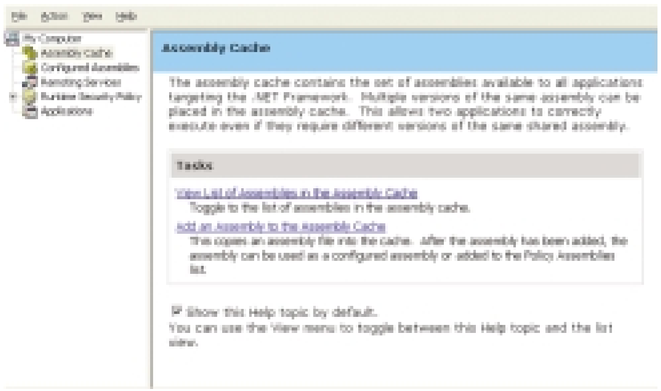
```
    public bool CreateBarCodeImage(string filepath, string code, int width, int height, float fontsize)
    {
      try
      {
        Bitmap bmp = new Bitmap(width, height);
        Graphics grfx = Graphics.FromImage(bmp);
        grfx.FillRegion(Brushes.Ivory, new Region(new Rectangle(0,0,width,height)));
        grfx.DrawString(code, new Font("IDAutomationHC39M", fontsize), Brushes.Black, 5,5);

        bmp.Save(filepath, ImageFormat.Png);

        return true;
      }
      catch(Exception excep)
      {
        return false;
      }
    }
  }
}
```

Figure 8: Main function in .NET assembly, generating the barcode image



```
C:\>sn -k BarCodeGenerator.snk

Microsoft (R) .NET Framework Strong Name Utility  Version 1.0.3705.0
Copyright (C) Microsoft Corporation 1998-2001. All rights reserved.

Key pair written to BarCodeGenerator.snk

C:\>_
```

Figure 9: Creating a strong name key example



The assembly cache contains the set of assemblies available to all applications targeting the .NET Framework. Multiple versions of the same assembly can be placed in the assembly cache. This allows two applications to correctly execute even if they require different versions of the same shared assembly.

View List of Assemblies in the Assembly Cache
Toggle to the list of assemblies in the assembly cache.

Add an Assembly to the Assembly Cache
This copies an assembly file into the cache. After the assembly has been added, the assembly can be used as a configured assembly or added to the Policy Assemblies list.

[✓] Show this Help topic by default.
You can use the View menu to toggle between this Help topic and the list view.

Figure 10: Configuration screen for global assembly cache

and compiled. We generate a strong name key and associate it with the assembly by using the command-line utility sn.exe. Third, as with any .NET component, in order to be visible to COM clients, the assembly must be registered as a COM component. In our example we use regasm.exe to fulfill this task. Finally, we add our assembly to the Global Assembly Cache (GAC) where it is cached and ready to be called by a COM client.

Let's start by taking a look at the C# source file. We will break down each section with a brief explanation and detail any portions of the code that have particular significance to COM interop.

In the code block shown in Figure 5, we are specifying the namespaces that we will be utilizing, such as System.Drawing. We then specify that our class will be placed in the "BarCodeInterop" namespace. The namespace we use has significance in determining the common name we will use to refer to the component. In keeping with the binary interface concepts of COM, it is required that we define a public interface for the method that we will be implementing for our functionality.

COM components implement the IDispatch interface. The IDispatch interface exposes the contents of an object to all programs that support Microsoft Automation. The COM callable wrapper that is responsible for implementing IDispatch for our component will use the interface we have specified to return a COM signature for our method "CreateBarCodeImage". When we use OLEView to inspect the interface our COM object exposes, we get the method signature detailed in Figure 6.

In Figure 7 the first notable item is an "attribute" that provides direction to the compiler for the type of COM interface we want to implement. Using a class interface type of none, our class can provide access only through a late bound interface exposed by IDispatch. Note that our class implements the interface "IBarCodeGenerator" that we defined earlier.

Figure 8 contains the implementation of the required interface method "CreateBarCodeImage". The method creates an image file through the file path specified, with a width, height, and font size specified.

The steps that follow the writing of our code include signing the assembly, adding the assembly to the global assembly cache, and registering the assembly for COM interop. In order to sign our assembly we must first generate a strong name key. We use Microsoft's sn.exe to generate the key (see Figure 9).

Once we have a strong name key, we place the file in the root of our project and associate it with the assembly by editing the "AssemblyInfo.cs" source file. AssemblyInfo.cs contains attributes that allow you to specify component information such as company, version, etc. There are also three attributes that are involved in signing an assembly. We specify our strong name key file by editing the AssemblyKeyFile attribute as:

```
[assembly: AssemblyKeyFile("..\\..\\BarCodeInterop.snk")]
```

After signing the assembly and compiling it, the assembly is ready to be added to the global assembly cache. The .NET framework installs the .NET Configuration wizard located under Control Panel -> Administrative Tools, which allows the addition and configuration of assemblies to the global assembly cache, among other things (see Figure 10).

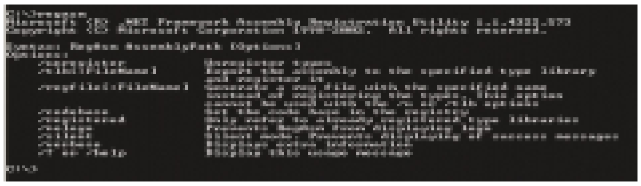
Simply click "Add an Assembly to the Assembly Cache" task link to browse to the assembly location and complete the step of adding the assembly to the global assembly cache.

The final step needed to expose our component to COM is registering the component as a COM component in the system registry. This task is accomplished using the command line utility regasm.exe (see Figure 11).

A typical registration would look like this: "regasm C:\BarCodeInterop.dll".

With the .NET component wrapped and registered, CFMX is able to see the objects' methods and their input parameters.

There you have it! An example that proves COM interop to a .NET assembly can be successfully implemented with CFMX. Why do it? This question has been answered.... It works, and it works well. Organizational leadership must assess costs



```
C:\>regasm
Microsoft (R) .NET Framework Assembly Registration Utility 1.1.4322.573
Copyright (C) Microsoft Corporation 1998-2002.  All rights reserved.

Syntax: RegAsm AssemblyPath [Options]
Options:
    /unregister          Unregister types
    /tlb[:FileName]      Export the assembly to the specified type library
                         and register it
    /regfile[:FileName]  Generate a reg file with the specified name
                         instead of registering the types. This option
                         cannot be used with the /u or /tlb options
    /codebase            Set the code base in the registry
    /registered          Only refer to already registered type libraries
    /nologo              Prevents RegAsm from displaying logo
    /silent              Silent mode. Prevents displaying of success messages
    /verbose             Displays extra information
    /? or /help          Display this usage message

C:\>
```

Figure 11: Regasm options

respective to redesign with the ever-tightening budgets associated with IT. An environment that has been structured around COM and ColdFusion can now easily be converted to .NET and ColdFusion.

With all that goes into the base concepts of integrating CFMX and .NET solutions, we look forward to the next article on the subject of using a native .NET interface for CFMX known as Black Knight. We will also be discussing Web services solutions and their performance parameters. Until then...

About the Authors

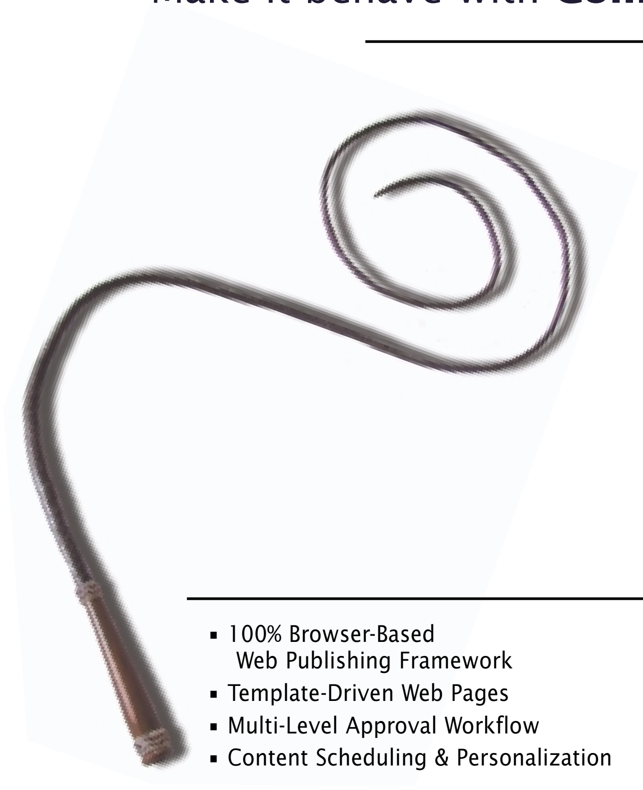
John Parrish has spent his career developing component-oriented systems for the Web. He also serves as director for Jikapa Interactive Inc, an interactive solutions firm providing software and corporate design solutions (www.jikapa.com).

Jeffrey Bouley is the founder of Strikefish, Inc., in Orlando, Florida. He is a certified ColdFusion developer and a former Macromedia Consultant. He currently provides software architecture and development solutions for ColdFusion, Java, .NET, XML, SQL Server, and Oracle integration (www.strikefish.com).

jparrish@jikapa.com

jeffrey_bouley@strikefish.com

Content can be unruly.
Make it behave with **CommonSpot™**



CommonSpot™ Content Server.
Put content management in the hands of content owners.

To tame your unruly web site, whip your content into shape with CommonSpot. With a rich, out-of-the-box feature set, you'll be up and running in weeks – not months. Built in ColdFusion, it's easy to integrate and customize CommonSpot to meet your requirements. CommonSpot's intuitive, browser-based interface makes authoring a snap, while its template and data-driven architecture and granular permissions allow for flexible yet powerful control.

Find out why CommonSpot was voted Best Web Content Management Tool in the CFDJ Reader's Choice Awards. Visit www.paperthin.com, or call today to schedule an online demonstration.

With CommonSpot, you'll have content tamed in no time.

- 100% Browser-Based Web Publishing Framework
- Template-Driven Web Pages
- Multi-Level Approval Workflow
- Content Scheduling & Personalization
- Rich Custom Metadata Support
- 508 Accessibility Compliance
- Seamless ColdFusion Integration
- More than 50 powerful, out-of-the box features

Paper Thin

800.940.3087
www.paperthin.com

Adventures in Encapsulation PART IV

Taking on different roles

By Hal Helms

I was playing a particularly tough game of online Texas Hold'em with someone who called himself "all_or_nothing". I looked at my two hole cards: a 10 of spades and a 10 of clubs. It was down to just the two of us, vying for the total prize money of $5.00. I was considering betting all my chips on this single hand. Or, rather, I was trying to consider that option. My attention was divided between this hand and the voice in my headset droning on…

CALLER: So after I got this big fine from the European Union, it hit me: hey, we *are* monopolists! And then I thought, maybe I should just open source Win...
HAL: Hold that thought, Bill.

I pushed the "All In" button and waited for "all_or_nothing" to respond. My caller kept going on and on with me barely listening.

CALLER: …and then maybe I could use my money to create a school for wayward CEOs. With courses like "Ethics for Dummies" and "Non-Creative Accounting" and "Pension Funds and Piggy Banks: Similar, but Different" – stuff like that. Whaddya think?

Ha! I watched as "all_or_nothing" waited and then pushed all his chips in. By the end of this hand, we would have one winner and one loser. And I knew where I wanted to be. But, now I had to concentrate.

HAL: Bill? Listen, lemme get back to you on that…that thing you were talking about.
BILL: Well, I just thought, I've made all this money; why not just do the right thing? What do you...
HAL: (making crackling noises into the phone) You're breaking up, Bill. We have a bad connection. Call me back on a landline.
BILL: I *am* on a landline. Say, you're not trying...

I quickly hung up. I watched the screen as the flop came out: 2 of diamonds, 7 of hearts, 7 of clubs. I breathed a sigh of relief: the only person who would be helped out much by this flop would have to be holding onto the worst hand in poker: an unsuited 2–7. The next card came out: a 4 of diamonds. That would be no help to either of us, I was sure. The final card was an ace of clubs. Was it possible that all_or_nothing was holding an ace? If so, his pair of aces would beat my pair of 10s.

He flipped over his cards: he was holding an unsuited 2–7, giving him a full house and crushing my lowly pair of 10s. I was stunned. I had lost to someone who should never have been playing the hand at all. This was what poker pros called a "bad beat." The phone rang again.

HAL: Yeah?
CALLER: Yeah? That's how you answer your phone now?
HAL: Oh, hi, John. Sorry, I just took a bad beat.
JOHN: Yeah, me too. But mine was a different kind of bad beat. It happened when I got my March copy of ***CFDJ*** only to find out that you had turned our phone call into an article!
HAL: Yeah, that was pure inspiration. A pretty good article, I thought.
JOHN: Funny how you came across as Mr. KnowItAll and I sounded like a dunderhead.
HAL: Yeah, John, that was my editor's fault. I was shocked when I saw what she had done with the final copy. I can tell you, I plan to protest most vigorously.
JOHN: Oh, please…
HAL: Look, John, I can understand how you might be upset and all, but I assure you this won't happen again.
JOHN: What won't?
HAL: Me, using our conversations as fodder for articles.
JOHN: So you won't be using *this* conversation, I trust?
HAL: John, what did I just say? Have you ever known me to go back on my word?
JOHN: Well, yes, actually. Let's see: there was the time we were in Chicago and you told me "Just pay the hotel bill and I'll reimburse you." And when you invited me to stay at your house – and then billed me for it. And again...
HAL: Now, John, I'm sure you didn't call me just to stroll down memory lane. But I'm a big enough man to admit that, in the words of a recent U.S. President, "mistakes were made." So, now that's all cleared up, what can I help you with?

JOHN: In the article, you said that you didn't like the way I was planning to do the Element CFC.
HAL: Refresh my memory…
JOHN: This is for the tool I'm building for requirements gathering. I have the idea of a storyboard that has multiple pages and each page has different elements on it.
HAL: And what were these elements?
JOHN: Things like articles, forms, images. So, I wanted to do this app using OO. I identified a Storyboard CFC, a Page CFC, and an Element CFC.
HAL: Yeah, I remember. And I questioned why you were using just one Element CFC.
JOHN: And I explained that I was going to have an "elementType" property that would indicate the type of element it was.
HAL: Yeah – that's not too good.
JOHN: So I gathered from reading your article. But, what's wrong with it?
HAL: Well, that's not easy to explain.
JOHN: I'm a good listener…
HAL: All right. Well, forget about Storyboards and Pages and Elements for now. Let's take something that's very simple to think about. Let's say that we're modeling a zoo.
JOHN: Okay.
HAL: The zoo will have animals, of course. So let's create an Animal CFC. But what kind of animal?
JOHN: Well, we could do what I was going to do with Element. Instead of having "elementType", we could have "animalType".
HAL: Now, there's an idea! And "animalType" would be…
JOHN: It could be "tiger" or "giraffe" or whatever.
HAL: So it's a string.
JOHN: Yeah, sure.
HAL: Okay. Now, let's say that we have a ZooKeeper class and one of its responsibilities is to feed the animals, one at a time.
JOHN: By looping over an ArrayList of zooAnimals, right?
HAL: Something like that, yes.
JOHN: Okay, that's pretty simple. First, I'd give the Animal class an eat() method. Then the feed() method of ZooKeeper would look like this:

```
<cffunction name="feed" access="public" returntype="void"
output="false">
   <cfargument name="animal" type="Animal" required="true" />
   <cfset arguments.animal.eat() />
</cffunction>
```

HAL: That was simple enough. But you aren't actually giving them anything to eat.
JOHN: Yeah…
HAL: Let's keep it simple and say that an animal gets either meat pellets or veggie pellets.
JOHN: Okay. How about this? We'll assign each animal an attribute called "foodType" which will be either "veggie" or "meat". I'll redo ZooKeeper's feed method:

```
<cffunction name="feed" access="public" returntype="void"
output="false">
   <cfargument name="animal" type="Animal" required="true" />
   <cfif arguments.animal.getFoodType() IS "meat">
         <cfset arguments.animal.eat("meat pellets") />
   <cfelseif arguments.animal.getFoodType() is "veggie">
         <cfset arguments.animal.eat("veggie pellets") />
   </cfif>
</cffunction>
```

HAL: Your method is getting a little more complicated there.
JOHN: Well, it's just reflecting the fact that the world is kind of complicated.
HAL: All right, then. Let's say that some animals are dangerous animals and that the feeding process for DangerousAnimals involves:
1. Locking them in their cages
2. Placing food for them
3. Unlocking them so they can get to the food
JOHN: Okay…
HAL: Now what do you do?
JOHN: Let's have a "ferocity" attribute that gets set to "friendly" or "dangerous". Then, we would have to add "lockAnimal" and "unlockAnimal" methods to ZooKeeper, I suppose.
HAL: And what does the ZooKeeper's feed method look like?
JOHN: Um…let's see…

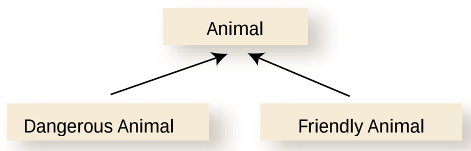
```
<cffunction name="feed" access="public" returntype="void"
output="false">
   <cfargument name="animal" type="Animal" required="true" />
   <cfswitch expression="#arguments.animal.getFerocity()#">
         <cfcase value="friendly">
               <cfset arguments.animal.eat("veggie pellets") />
         </cfcase>
         <cfcase value="dangerous">
               <cfset lockAnimal(arguments.animal) />
               <cfset arguments.animal.eat("meat pellets") />
               <cfset unlockAnimal(arguments.animal) />
         </cfcase>
   </cfswitch>
</cffunction>
```

There, that will do it.
HAL: Now, we tend to think of dangerous animals as meat eaters, but some dangerous animals are vegetarians.
JOHN: Like what?
HAL: Like hippos, for example.
JOHN: Okay. Well, then…

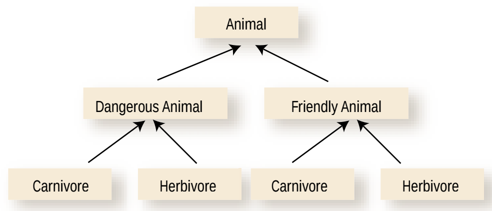
```
<cffunction name="feed" access="public" returntype="void"
output="false">
   <cfargument name="animal" type="Animal" required="true" />
   <cfswitch expression="#arguments.animal.getFerocity()#">
         <cfcase value="friendly">
               <cfif arguments.animal.getFoodType() is "meat">
               <cfset arguments.animal.eat("meat pellets") />
               <cfelseif arguments.animal.getFoodType is "veggie">
               <cfset arguments.animal.eat("veggie pellets") />
               </cfif>
         </cfcase>
         <cfcase value="dangerous">
               <cfif arguments.animal.getFoodType() is "meat">
               <cfset lockAnimal(arguments.animal) />
               <cfset arguments.animal.eat("meat pellets") />
               <cfset unlockAnimal(arguments.animal) />
               <cfelseif arguments.animal.getFoodType() is "veggie">
               <cfset lockAnimal(arguments.animal) />
```

```
                        <cfset arguments.animal.eat("veggie pellets") />
                        <cfset unlockAnimal(arguments.animal) />
                        </cfif>
            </cfcase>
    </cfswitch>
</cffunction>
```

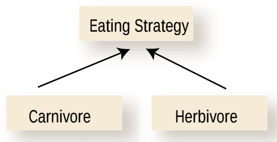
HAL: Wow, that's…ugly.
JOHN: What – it works!
HAL: I didn't say it wouldn't work; I said it was ugly. It's ungainly and clunky. And John, have you ever noticed that every so often, after you think you're all done, the requirements change a little?
JOHN: That only happens on days the sun rises in the *east*.
HAL: Yeah, and when those requirements change, do they tend to make the application *more* or *less* complex?
JOHN: Why do I feel I'm being set up?
HAL: Your Honor, please instruct the witness to answer the question.
JOHN: All right, *more* complex.
HAL: And so your code, which right now only a mother could love, may very well get even uglier. For example, there are animals that are omnivores. They eat both meat and veggies, so your code will need to get more complicated still to accommodate that.
JOHN: Hey, that's life.
HAL: No, that's procedural code.
JOHN: It's not procedural: I'm using CFCs!
HAL: Yeah, and I'm wearing a Yankees baseball cap, but that doesn't make me Alex Rodriguez.
JOHN: So what's the right way to do it?
HAL: I'll leave it up to you to decide what the right way is, but I can tell you a more object-oriented way to handle this sort of situation.
JOHN: I'm listening.
HAL: First, let's introduce some inheritance relationships.
JOHN: So, we start with Animal and then have MeatEater and VeggieEater that extend Animal?
HAL: We could try that out, but I like this one better:

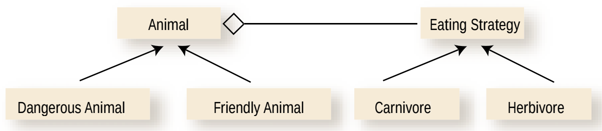


JOHN: Oh, I see where you're going:



HAL: Not quite. I don't like that duplication. It's just…
JOHN: Ugly?
HAL: Precisely. I like your idea of a Carnivore and an Herbivore class, but let's put them in a different inheritance hierarchy:

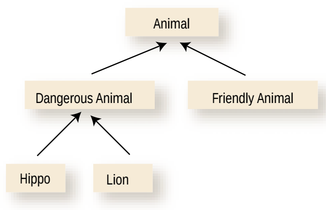




Now, let's relate this diagram to the first one I drew:
JOHN: That's UML and the hollow diamond means something…
HAL: It means that an Animal object has an EatingStrategy object as part of it. Like this:

```
<cfcomponent displayname=Animal>
   <cfset variables.eatingStrategy = "" />

   <cffunction name="init" access="public" returntype="Animal" out-
put="false">
         <cfargument name="eatingStrategy" type="EatingStrategy"
required="true" />
         <cfset setEatingStrategy(arguments.eatingStrategy) />
   </cffunction>

   <cffunction name="getEatingStrategy" access="public"
returntype="EatingStrategy" output="false">
         <cfreturn variables.eatingStrategy />
   </cffunction>
   <cffunction name="setEatingStrategy" access="public"
returntype="void" output="false">
         <cfargument name="eatingStrategy" type="EatingStrategy"
required="true" />
         <cfset variables.eatingStrategy = arguments.eatingStrategy />
   </cffunction>
</cfcomponent>
```

JOHN: How does that work?
HAL: Let's say that every EatingStrategy accepts an animal and knows what kind of food to pass it. For a carnivore, it passes meat pellets; for an herbivore, it passes veggie pellets. The EatingStrategy classes are deliberately kept kind of dumb – they just know about how to accept an animal and feed it. They don't know what kind of animal will make use of them.
JOHN: And that's a good thing?
HAL: Absolutely. We can create a Hippo subclass and a Lion subclass and just assign different eating strategies to each.



HAL: And wouldn't it be even nicer if you could have this code in ZooKeeper?

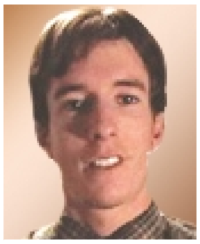
```
<cffunction name="feed" access="public" returntype="void"
output="false">
   <cfargument name="animal" type="DangerousAnimal" required="true">
   <cfset lockAnimal(arguments.animal) />
   <cfset getEatingStrategy().eat(arguments.animal) />
   <cfset unlockAnimal(arguments.animal) />
</cffunction>

<cffunction name="feed" access="public" returntype="void"
output="false">
   <cfargument name="animal" type="FriendlyAnimal" required="true">
   <cfset getEatingStrategy().eat(arguments.animal) />
</cffunction>
```

JOHN: Hey, you have the same method – feed – written twice.
HAL: Yes, but notice the different type that each version of the method takes? One accepts a DangerousAnimal and the other accepts a FriendlyAnimal. That's called *method overloading*: the same method with different argument types and/or numbers implemented differently.
JOHN: But don't you need a switch statement – or at least an if statement – to determine which one to call?
HAL: No, that's handled automatically at runtime, depending on what type of argument is passed to the feed method.
JOHN: So, method overloading is the secret to object orientation?
HAL: No, the real secret to OO is polymorphism. It's the ability for one type to take on different roles, so to speak. It's what enables us to declare that Animal has a generic EatingStrategy, while actually it receives a specific type of EatingStrategy.
JOHN: Either a Carnivore or an Herbivore.
HAL: Yes, so a Carnivore object belongs to both Carnivore and EatingStrategy.
JOHN: Two data types in the same object.
HAL: That's what polymorphism means: "many forms."
JOHN: I've heard that term before, but didn't know exactly what it meant. So just in case the requirements change...
HAL: And we end up with an Omnivore...
JOHN: Yes, we can just extend EatingStrategy with an Omnivore class and we won't need to change Animal at all. And that's what I need to do with Element! I'll say that all pages have an ArrayList of Elements, and then assign individual element types at runtime, right?
HAL: You have it exactly right.
JOHN: And then, I'll just overload some methods...
HAL: Err…John. That part won't exactly work.
JOHN: Why not? What's wrong with overloading my methods?
HAL: Nothing. It's a great idea and it really leverages polymorphism. But ColdFusion doesn't support method overloading. You can still do most of what we did, but you'll need some additional...
JOHN: As in "kludgey"...
HAL: As in kludgey – code to sort out the difference between FriendlyAnimal and DangerousAnimal.
JOHN: I don't like that nearly as well.
HAL: I don't either.
JOHN: ColdFusion should have method overloading! I bet Java has method overloading.
HAL: Yes and yes. Maybe someday I'll write an article that points out how useful method overloading would be…
JOHN: But the whole EatingStrategy, that works just fine and that's pretty useful in all sorts of situations.
HAL: Right.
JOHN: I started off thinking we would have to use inheritance.
HAL: Everybody does when they first come to OO.
JOHN: Well, that really is different than the way I was approaching the project.
HAL: Yeah, it's very possible to keep writing procedural code inside OO constructs like classes. But you don't get the benefits of OO when you do that. That's why I tell people: tackle OO now, before you absolutely need it. Because it's going to take a while for it to really get into your system and before you stop trying to write procedural code in OO.
JOHN: And the stuff like method overloading that ColdFusion doesn't have?
HAL: Yes, there's quite a lot like that. Some very important stuff, too. That's why I also recommend that people who want to learn OO start with Java. It's not just because Java is so popular. It's that it's a very clean implementation of OO. Starting with Java will help you work with other OO languages. Because you already know the "reference implementation" of OO, so to speak. You can then adapt to the differences in languages, whether it's CFCs, C#, Python, even ActionScript.
JOHN: Well, thanks a lot. And I forgive you for turning our last conversation into an article.
HAL: Yeah, sorry about that. Really, I promise not to do that again.
JOHN: Well, I've gotta run and get back to this game I was playing.
HAL: What game?
JOHN: I was just playing online poker with some clown who went all in with a pair of tens. I was holding a full house! What a loser!
HAL: You…what?
JOHN: Yeah, I was playing 2–7 off suit and I flopped a full house. Then this joker goes all in and I took all his money! Oh, yeah!
HAL: Why…were…you…staying…in…with…a…2–7?
JOHN: Worst hand in poker, right? Yeah, but from watching this guy play, I figured I could beat him with anything.
HAL: Wow, he does sound like a…loser. But what a bad beat you put on him! That was unforgivable!
JOHN: Yeah, sure. Not my problem, though!
HAL: And this guy – this loser, as you say – he probably would really like to get back at you. I mean, he would probably feel justified in doing all sorts of things to get his revenge.
JOHN: Like what?
HAL: I don't know – lie about you, make you look bad, break a promise…

– continued on page 43

Making the Most of J2EE Event Listeners

Listen and respond to session events


By Eric Brancaccio

In the Servlet 2.3 specification, Web application events were introduced. These Web application events pertain to both the application and session life cycle. Along with these events came a group of listeners. As you might imagine with all the talk about the integration of Java and ColdFusion, these J2EE event listeners exist on the CFMX platform, and proper use of them can be very beneficial to a developer.

I hope to provide an introduction and basic understanding of these event listeners and to examine some ways to listen for session events and respond with custom code.

The ColdFusion MX application server contains several built-in listeners that we can leverage to our advantage. These listeners are Java interfaces that we must implement with our own custom classes. Using these listeners makes it possible to respond to various events in the application and session life cycle. For example, we can respond to a session-destroyed event by writing all user data to a database. Some of these same tasks can be accomplished by coding the same functionality in each individual ColdFusion template, but making use of the server's built-in event listeners allows the developer to centralize code to easily respond to all events on the server. Clearly the possibilities are numerous. The goal of this piece is not to explain the best way to take advantage of these event listeners. Rather it is to provide some basic examples of how to configure a custom event listener on the CFMX platform.

How does the process of listening and responding to a Web application event work? When an event occurs, the server passes event objects to each method of the appropriate listeners. These listeners are Java interfaces, so the methods that receive these event objects currently do nothing. We will write some listener classes to handle session creation and invalidation by logging the events in a log file and writing them to a database. The listener that we will be implementing is the javax.servlet.http.HttpSessionEventListener, which we will use to manage various events in the session life cycle. To leverage this listener in the manner that we want, we will have to write custom classes that implement this HttpSessionListener interface.

HTTPSessionEventListener

The events that we are going to listen and respond to are session creation and invalidation (destruction). When one of these HTTPSessionEvent instances occurs, the methods of the HTTPSessionEventListener interface are triggered, and through these methods we can access the HttpSession object. For example an HTTPSession event would occur if a user navigated to a ColdFusion application in which sessionmanagement was enabled.

Once we have access to this HttpSession object we can call various methods to get all the information pertaining to the session that we will need. The two methods of the HTTPSessionListener interface that can be triggered by session events are the sessionCreated() method and sessionDestroyed() method. When we implement this interface, we provide our own functionality for these two methods. The sessionCreated() method is notified when the server creates a new session object by calling request.getSession(true). The sessionDestroyed() method is notified when a session times out or is destroyed from a call to session.invalidate.

There are many ways we may choose to respond to session creation and session destruction events. For example we may want to log these events or we may want to store active sessions in a database that can be queried at any time to give the number of active sessions on a particular server. Logging these events is very straightforward:

```
public void sessionCreated(HttpSessionEvent sessionevent) {

// get the session object
HttpSession session = sessionevent.getSession();

//HttpSession method getId() returns the sessionid
String sid = session.getId();

//the standard output will write to cfusionmx\runtime\logs\default-
out.log
System.out.println("Session " + sid + " created");
 }

public void sessionDestroyed(HttpSessionEvent sessionevent) {

HttpSession session = sessionevent.getSession();

String sid = session.getId();
System.out.println("Session " + sid + " destroyed");
 }
```

With this code in place, the server will now respond to all session creation and destruction events by logging them. The System.out.println calls will write to cfusion-mx\runtime\logs\default-out.log. Next we will take it a step further by writing the new sessions to a SQL server database and deleting them when the session is destroyed. This will allow us to keep track of all active sessions on our server:

```
public void sessionCreated(HttpSessionEvent sessionevent) {


HttpSession session = sessionevent.getSession();

String sid = session.getId();
long time = session.getCreationTime();

//create the sql string to insert the session id and creation time
String sql = "insert into activesessions(sessionid, createtime)
values(' "+sid+" ', ' "+time+" ')";
//create the connection string for the SQL server DBLOG
String url= "jdbc:odbc:dblog";


try{
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

 Connection conn =
DriverManager.getConnection(url,"username","password");
 Statement stmt = conn.createStatement();
 stmt.executeUpdate(sql);
}
catch (Exception e)
{e.printStackTrace();}

System.out.println("Session " + sid + " created");
}


public void sessionDestroyed(HttpSessionEvent sessionevent) {

String sid = sessionevent.getSession().getId();

String sql = "delete from activesessions where sid = (' "+sid+"
')";
String url = "jdbc:odbc:dblog";


try{
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

 Connection conn =
DriverManager.getConnection(url,"username","password");
 Statement stmt = conn.createStatement();
 stmt.executeUpdate(sql);
}
catch (Exception e)
{e.printStackTrace();}

System.out.println("Session " + sid + "destroyed");

}
```

With this code added to our custom listener, we can do a count on the activesessions table to get the current number of active sessions. This session counting code exists in one location and can be used for all current and future applications on the server. These are just some of the many ways that a developer can implement the HTTPSessionListener interface (Listing 1 is a compilation of all the code snippets in this article up to this point.)

Configuring Your Server to Use Your Listener

There are a few things that you will need to do on your CFMX application server in order to have it use your listener. First you must check the "Use J2EE session variables option," which is under the memory variables section within the ColdFusion Administrator. If the server is only using the cfid and cftoken to track user sessions, you cannot make use of the HttpSessionEvents. Next you must place the compiled class in the CFusionMX\wwwroot\WEB-INF\classes\ directory. The last step involves modifying the web.xml file so that it uses your class file as a listener. I encourage all developers to familiarize themselves with the web.xml file and its functionality. The web.xml can be found in the CFusionMX\wwwroot\WEB-INF\ directory. The following must be added to the XML document within the <web-app> tags with the classpath of the compiled class file in the <listener-class> tags (see Listing 2):

```
<listener>
  <listener-class>
    MyHttpSessionListener
  </listener-class>
</listener>
```

We may add multiple listeners as long as they are all registered in this fashion. Once this is completed, restart the CFMX server, and it will use the new configurations.

Conclusion

We now have a fairly simple custom class for listening and responding to session events. Leveraging the power of existing Java classes and interfaces has opened up a world of potential for ColdFusion developers. These events are being fired. So why not take advantage of them? Other listeners include the HttpSessionAttributeListener, the HttpSessionActivationListener, and the ServletContextListener. The ServletContextListener deals with the application life cycle. Each one can be implemented in a manner similar to the HttpSessionListener.

About the Author

Eric Brancaccio is a Certified Advanced ColdFusion Developer. He is a senior software engineer for the American Chemical Society in Washington, DC.

E_brancaccio@acs.org

LISTING 1: MyHttpSessionListener.java

```
// Example implementation of the HttpSessionListener Interface

package com.embconsulting;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.http.HttpSessionListener;
import java.sql.*;



public class MyHttpSessionListener implements HttpSessionListener {

public void sessionCreated(HttpSessionEvent sessionevent) {


HttpSession session = sessionevent.getSession();

String sid = session.getId();
long time = session.getCreationTime();


String sql = "insert into activesessions(sessionid, createtime) val-
ues(' "+sid+" ', ' "+time+" ')";
String url = "jdbc:odbc:dblog";


try{
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

 Connection conn = DriverManager.getConnection(url,"username","pass-
word");
 Statement stmt = conn.createStatement();
 stmt.executeUpdate(sql);
}
catch (Exception e)
{e.printStackTrace();}

System.out.println("Session " + sid + " created");

}

public void sessionDestroyed(HttpSessionEvent sessionevent) {

String sid = sessionevent.getSession().getId();

String sql = "delete from activesessions where sid = ('"+sid+"')";
String url = "jdbc:odbc:dblog";


try{
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

 Connection conn = DriverManager.getConnection(url,"username","pass-
word");
 Statement stmt = conn.createStatement();
 stmt.executeUpdate(sql);
}
catch (Exception e)
{e.printStackTrace();}

System.out.println("Session " + sid + "destroyed");

}

}
```

LISTING 2: Code to add to web.xml file

```
<!-- code to add web.xml document -->

<listener>
  <listener-class>
    com.embconsulting.MyHttpSessionListener
  </listener-class>
</listener>
```

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

Reach Over 100,000

Enterprise Development Managers & Decision Makers with...

>Linux >Java >Web Services >.NET >XML >Wireless >Storage >Security
The Premier Resource for Today's Corporate & IT Decision Makers
IT solutions >GUIDE
WWW.SYS-CON.COM/IT
Top IT Solution Providers!
> Candle
> ClearNova
> Computer Associates
> Creative Science
> DralaSoft
> GraphOn
> InterSystems
> ITKO
> Merant
> O'Reilly
> Parasoft
> QuadBase
> SAP
> Wily Technology
> Zero G
TECHNOLOGIES YOU NEED NOW!
How to Manage Your Ideas Using Today's i-Technologies

Offering leading software, services, and hardware vendors an opportunity to ***speak to over 100,000 purchasing decision makers*** *about their products, the enterprise IT marketplace, and emerging trends critical to developers, programmers, and IT management*

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!

*ONLY 20 ADVERTISERS WILL BE DISPLAYED. FIRST COME FIRST SERVE.

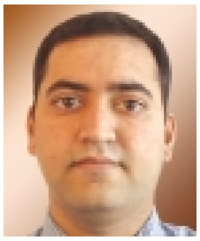
SYS-CON MEDIA

The World's Leading i-Technology Publisher

Model-View-Controller based Plug-able Declarative Framework

Simplify the development of scalable, maintainable, and robust Web applications

By Pramod Jain


By Yayati Kasralikar

A Model-View-Controller based Plug-able Declarative Framework (MVC-PDF) will be described, in which developers use the full power of ColdFusion MX (CFMX) components to create MVC applications. The declarative nature of the framework will allow the controller and the model to be specified declaratively in configuration files, versus custom coded in ColdFusion.

A MVC-PDF will allow developers in a team to write no ColdFusion components; instead, these developers will code the business logic in controller and model XML configuration files.

The pluggable nature of the framework will allow developers to create components for custom business logic that will then be called declaratively. These model components extend an abstract model component provided by a MVC-PDF.

The benefits of a MVC-PDF are higher productivity and quality when building large, scalable, robust, and maintainable ColdFusion MX Web applications. This is because in most situations, MVC-PDF developers will write code at the two ends of an application: the front end (CFM pages with only presentation logic) and the back end (for instance, SQL, stored procedure, and conditional logic). A robust declarative and pluggable middle tier takes care of the rest. A MVC-PDF will lead to a very high degree of component reuse in both the model and controller layers. In an extreme case of reuse, developers will (a)write only CFM files that contain only the presentation logic, and (b) write the desired business logic in two configuration XML files. Note that developers are not writing business logic in CFM files and in most cases are not writing components (CFC files).

Background

ColdFusion MX provides the capability to create applications that use the MVC paradigm – separation of presentation from business logic. One method of using the MVC paradigm is to encapsulate business logic in model components, and to call these components from submit events of CFM pages. An example of business logic implemented in a component is (a) execute a select statement; (b) based on the data returned by select, execute an insert or update statement; and (c) run another select to retrieve data for sending to the next page.

Mach-II (www.mach-ii.com) is a popular component-based framework that uses the implicit invocation architecture for developing Web applications in CFMX. Its focus is on creating reusable listeners. As the name suggests, listeners are components that (a) listen to events submitted by CFM pages; (b) connect the events to a sequence of actions that must be performed by models to process an event; and (c) return a view page with dynamic data from a model. Mach-II provides a reusable listener component that is configured using an XML file. However, developers still write model components to execute business logic. MVC-PDF uses the Mach-II framework as a listener; however, it significantly extends Mach-II by providing reusable model components for business logic.

As an illustration, consider an application that has 10 CFM pages, each requiring business logic to be executed on submit. If the Web application requires 30 separate SQL statements (10 submit events and 3 SQL statements per event), in Mach-II then developers would either write 30 components each with one method, or 10 components each with 3 methods. In either case, developers would be writing a substantial amount of ColdFusion code to manipulate variables, run a query, and return data. This code would be largely repetitive, but subject to developer experience, whims, and styles.

The MVC-PDF offers high reusability. For instance, in the above example, developers would configure prebuilt SQLModel and ConditionModel components of the MVC-PDF. The entire Web application is built without writing any components or methods. Writing of components is replaced by configuration, using XML files of two prebuilt components provided by the MVC-PDF.

Developers can write additional reusable components and plug them into the MVC-PDF to accomplish business logic that cannot be implemented with the available components. This approach eliminates the proliferation and management of large numbers of business components and methods.

Another aspect of the MVC-PDF is a single Hierarchical Data Structure (HDS) that is used by all MVC-PDF components to retrieve and store data, and is used by the presentation tier to extract and render dynamic data. This significantly simplifies the programming of CFM pages, because the developer works with a single data structure that contains all the data required by the presentation page.

The MVC-PDF is based on a design pattern for building large systems that consist of:

- Assembling and declaratively configuring a small number of robust reusable components
- Single hierarchical data structure that is used by these components to retrieve and store data, and used by the presentation tier to retrieve dynamic data and put into a CFM page

What Is MVC-PDF?

To understand the MVC-PDF and how it changes the development of Web applications, consider the examples shown in the listings that follow.

In the first example, a Web page is submitted with <form action="index.cfm?event=getAllProjects &userid=2" method="post">. An event listener of the Mach-II framework is configured as shown in Listing 1. It specifies that the event **getAllProjects** will be processed by component **com.indent.controller.IndentController** and method execute. This method will call a business component specified as **bc1=getProjectsForUser**. After the business component is executed, control will pass to the **projectDisplay** view page. This view page is specified in the <page-views> element.

Next, **getProjectsForUser** is specified in business-components.xml.

```
<business-component-defs>
   <bc name="SQL" type="com.indent.model.SQLModel"/>
</business-component-defs>

<business-components>
   <bc name="getProjectsForUser" type="SQL" parent="getAllProjects.bc1">
   <parameter name="desc" value="Illustration of SQL business compo-
nent"/>
   <parameter name="db" value="DBAlias"/>
   <parameter name="method" value="select pName as projectName from proj-
ects
            where uid=#hds.userid#" />
   </bc>
</business-components>
```

getProjectsForUser is specified as a component of type SQL with three parameters: description, database alias, and select statement. Notice, the userid is passed in from the URL and replaced in the select as #hds.userid#. hds is single data structure that contains all the URL parameters and data created by the business components. Business components of type=SQL are processed by the com.indent.model.SQLModel business component, as specified in business-component-defs; the attribute parent contains a reference to the Mach-II events.

The SQL business components may contain select, insert, update, or delete statements. The output of the SQL statement will be stored in **hds.getProjectsForUser**, where getProjectsForUser is the name of the business component. In the following snippet, the output of select is stored in **hds.getProjectsForUser.projectName**.

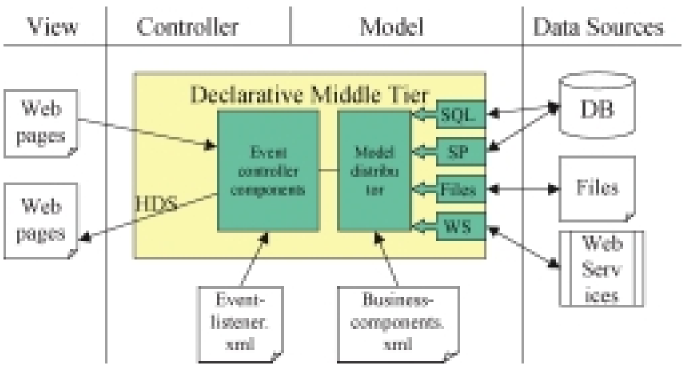


Figure 1: Components of the MVC-PDF framework. When a Web page is submitted with event parameters, the controller connects to the listener for the event. An event calls the Model Distributor (MD) to execute a sequence of business components that is specified in event-listener.xml. MD collects all the data from the multiple business components and returns in a hierarchical data set (HDS) to the return CFM page. The business components are defined and configured using the business-components.xml file.

ProjectDisplay.cfm, the view page specified above will look like this:

```
…
<table><cfoutput
query="hds.getProjectsForUser">
<tr><td>#projectName#</td></tr>
</cfoutput></table>
…
```

In Listing 2, the following example will be used: user is submitting a project name, task name, and task description that need to be inserted or updated into a database. If the project does not exist, then project and task are inserted in respective tables; otherwise they are updated.

The submitted page looks like this:

```
<form
action="index.cfm?event=updateProject&projectN
ame=Sample1&userid=2" method="post">.
    <input type="text" name="taskName"
value="Project kickoff">
    <textarea name="taskDesc">Project Kickoff
contains several activities</textarea>
</form>
```

The listener is configured for the updateProject event as shown in Listing 2.

If any of the business components, bc1 to bc9, throws an exception, then all subsequent executions are stopped and a failure event is executed.

Business components – bc1 to bc9 – inside <event-handler> element are executed in sequence, until one of the business components changes the sequence. The first business component is **getProject,** followed by **checkSelectProject**. As specified in the business-components.xml, **checkSelectProject** checks if **getProject** returned any rows. If true, then bc4, bc1, bc8, and bc9 are run. Otherwise, bc3, bc6, and bc7 are run. The complete flow is shown in Figure 2.

Two types of business components are used in Listing 3: **SQL** and **Condition**. Compared to Listing 1, the SQL definition here contains the database alias and the class that will handle the exception. The database alias in definition removes the need to specify it in each <bc> element. However, if one is specified in <bc>, then it overrides the database alias parameter in the business component definition. By default, transactions are rolled back if any of the business components in the event fail.

Each model component can be associated with its own exception class. If none is specified, then “any” exception is handled by the class com.indent.exception.AbstractException. An example of the exception class is provided in the next section.

The **Condition** business component type provides an “if … else if … else” type of logic. If a condition is satisfied, then the corresponding bclist provides the list and sequence of business components to be executed next. The value attribute of the condition uses the same syntax as condition in <cfif>.

Notice, URL parameter “projectName” is replaced in the where clause of the first select. The result of the query is stored in hds.getProject. In the next business component, **checkSelectProject**, #hds.getproject.recordcount# is used to find the number of rows returned. This business component has two conditions that correspond to the existence of projectName in the project table. If the recordcount is 0, then the sequence of business components is changed to the one contained in bclist1. Notice how URL parameters and values returned by **getProject** are used in the update statement in bc4 (updateProject) and others. hds, therefore, is the repository of all the URL parameters, form fields, and results of the business components that can be used in subsequent business components and in the output CFM page.

If there are no exceptions, then the success event is called and it outputs projectDisplay2.cfm. A simple projectDisplay2.cfm will look like:

```
…
<table>
    <tr><td colspan="3">List of Projects and
Tasks for #hds.userid#</td></tr>
    <cfoutput query="hds. getTasks">
<tr><td> #projectName#</td>
<td>#taskName#</td>
<td>#startDate#</td></tr>
    </cfoutput>
</table>
…
```

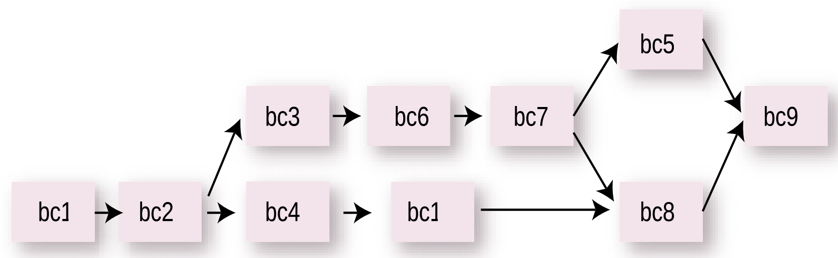


Figure 2: The logic of business components in Listing 2

Model Components in MVC-PDF

The pluggable aspect of a MVC-PDF allows business component developers to create custom model components that can be reused. A MVC-PDF provides an abstract class, **com.indent.model.AbstractModel**, from which all model components must extend. The AbstractModel provides the following facilities to a model component:

- ***Access to bcParams object:*** This object contains the XML object of the model's business component. This is the XML contained inside <bc> … </bc> in business-component.xml
- ***Access to bcDefParams object:*** This object contains the XML object of the model definition. This is the XML contained in <bc-def>…</bc-def> in business-component.xml
- ***Access to hds***

A developer of a business component can therefore access the configuration parameters as bcParams.method, bcDefParams.db, hds.variable. The developer can now focus on writing the business logic, while using all the configuration parameters and input variables (URL parameters, form fields, and results of prior business components). In return, the developer must put all the data he or she wishes to return into hds with the key as bcParams.bcName. As an example, the SQL model code is shown in Listing 4.

Sample code for SQLException is listed below. session.hds.errormsg is a string with the message in <cfthrow > and any other messages specified in the SQLException component. This string is output in the exception.cfm page.

```
<cfcomponent displayname="SQLException" hint="SQL
Model Error Handler" extends="com.indent.excep-
tion.AbstractException">
   <cffunction name="execute" access="public">
         <cfargument name="errormsg"
type="string" required="true"/>
         <cfargument name="detailmsg"
type="string" required="false" default=""/>
         <cfset session.hds.errormsg =
errormsg & "<br> SQLException" & detailmsg/>
   </cffunction>
</cfcomponent>
```

The framework and the code presented here may be obtained from www.indent.org/cfmxdj.htm. Such reusable model components that are plugged into a MVC-PDF make the task of developing even complex Web applications a matter of configuration of business-components.xml.

Hierarchical Data Sets in MVC-PDF

A common data structure called **hds** (hierarchical data set) of type struct is shared by all the business components. **hds** is stored as a session variable, and the architecture is akin to a pipeline. This pipeline starts with a data structure that contains the URL arguments and submitted form field elements. If the URL contains parameters like &param1=value1&param2=value2, then these are stored in a struct hds like: hds.param1 = value1 and hds.param2 = value2. These URL parameters are then available for use through #hds.param1# and #hds.param2#.

If a business component is a SQL with a select statement, for example **getProjectsForUser** of illustration 1: select pName as projectName from projects where uid=#hds.userid#", then the records returned by the SQL are inserted into hds like: hds.getProjectFromName=query. Values of projectName may then be accessed as hds.getProjectFromName.projectName[i].

The MVC-PDF provides a method for storing variables in **hds** that persist over the entire session. Keys in hds that start with "global_" are saved and persist for the entire session, while other keys in hds are destroyed after the display page is processed.

Conclusion

The MVC-based Plug-able Declarative Framework provides:

- A library of reusable model components that are configured using XML files
- A pluggable infrastructure that allows developers to build custom model components
- A unified hierarchical data structure (HDS) for returning data from the middle-tier to CFM pages; HDS contains all the dynamic data required by the outbound CFM page

The MVC-PDF should significantly simplify the development of scalable, maintainable, and robust Web applications.

About the Author

Pramod Jain is president of Innovative Decision Technologies, Inc. (INDENT, www.indent.org), in Jacksonville, FL. Their clients include Recruitmax, NASA, and NIH. Pramod has a PhD from the University of California, Berkeley.

Yayati Kasralikar is lead programmer at INDENT. He has an MS from the University of Central Florida, Orlando.

pramod@indent.org
yayati@indent.org

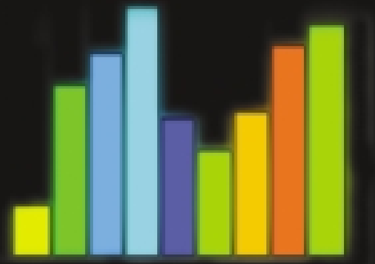
Listing 1:

```
<listeners>
   <listener name="IndentController"
type="com.indent.controller.IndentController">
   <invoker
type="MachII.framework.invokers.CFCInvoker_Event"
/>
   </listener>
</listeners>
…
<event-handler event="getAllProjects">
    <event-arg name="bc1"
value="getProjectsForUser" />
    <notify listener="IndentController"
method="execute" />
    <view-page name="projectDisplay" />
</event-handler>

<page-views>
   <page-view name="projectDisplay"
page="/views/projectDisplay.cfm"/>
```

Breakthrough Technology for the 21st Century

jVisgen Wizards chart wizard

Dynamic Interactive Charting
Bringing Java 2D and 3D to Cold Fusion Developers. Visually create Java chart servlets without writing a single line of code.



Dynamic Model
JDBC database connection

Dynamic Updates
Pure Java Servlets allow live updates to charts from your browser.

Create or Edit
With our WYSIWYG Editor and List boxes.

Flexible Licensing
Have it your way! $10 & $20 Chart Servlets Licensing, Consulting Services.

Simple to complex, charts that work. We Guarantee it!

Mention this ad to qualify for your introductory offer.

jvisgenwizards.org

A Java™ Sun Microsystems iForce Development Partner.

```
</page-views>
```

Listing 2:

```
<listeners>
   <listener name="IndentController"
type="com.indent.controller.IndentController">
  <invoker type="MachII.framework.invokers.CFCInvoker_Event" />
   </listener>
</listeners>
…
<event-handler event="updateProject">
  <event-mapping event="success" mapping="newProject" />
  <event-mapping event="failure" mapping="exception" />
  <event-arg name="bc1" value="getProject"/>
  <event-arg name="bc2" value="checkSelectProject" />
  <event-arg name="bc3" value="updateProject"/>
  <event-arg name="bc4" value="insertProject"/>
  <event-arg name="bc5" value="updateTask"/>
  <event-arg name="bc6" value="getTaskID" />
  <event-arg name="bc7" value="checkSelectTask"/>
  <event-arg name="bc8" value="insertTask"/>
  <event-arg name="bc9" value="getTasks"/>
        <notify listener="IndentController" method="execute" />
…
</event-handler>
<event-handler event="exception">
  <view-page name="exception" />
</event-handler>
<event-handler event=" newProject">
  <view-page name="projectDisplay" />
</event-handler>

<page-views>
   <page-view name="projectDisplay" page="/views/projectDisplay2.cfm"/>
   <page-view name="exception" page="/views/exception.cfm"/>
</page-views>
…
```

Listing 3:

```
<business-component-defs>
   <bc-def name="SQL" type=" com.indent.model.SQLModel">
<parameter name="db" value="DBAlias"/>
  <parameter name="modelException"
value="com.indent.exception.SQLException"/>
   </bc-def>
   <bc-def name="Condition" type=" com.indent.model.ConditionModel">
  <parameter name="modelException"
value="com.indent.exception.ConditionException"/>
  </bc-def>
</business-component-defs>

<business-components>
 <bc name="getProject" type="SQL" parent="updateProject.bc1">
  <parameter name="method" value="select projectid from project where
  createdby='#hds.userid#' and name='#hds.projectName#'" />
 </bc>
 <bc name="checkSelectProject" type="Condition"
parent="updateProject.bc2">
        <parameter name="condition1" value="#hds.getproject.recordcount#
eq 0" />
        <parameter name="bclist1" value="bc4,bc1,bc8,bc9"/>
        <parameter name="condition2" value="#hds.getproject.recordcount#
gt 0" />
        <parameter name="bclist2" value="bc3,bc6,bc7"/>
  </bc>
  <bc name="updateProject" type="SQL" parent="updateProject.bc3">
         <parameter name="method" value="update project set startdate =
#hds.startDate#,
  createdby='#hds.userid#' where projectid='#hds.projectid#'" />
   </ bc>
  <bc name="insertProject" type="SQL" parent="updateProject.bc4">
         <parameter name="method" value="insert into project (startdate,
createdby,
  name) (#hds.startDate#, #hds.userid#, '#hds.projectName#')" />
   </bc>
  <bc name="updateTask" type="SQL" parent="updateProject.bc5">
         <parameter name="method" value="update task set

tdesc='#hds.taskDesc#',
  sdate=#hds.startDate#,uid='#hds.userid#' where tid=#hds.taskid#" />
   </bc>
  <bc name="getTaskID" type="SQL" parent="updateProject.bc6">
         <parameter name="method" value="select tid as taskid from task
where
  pid=#hds.projectid# and tname='#hds.taskName#'" />
   </bc>
  <bc name="checkSelectTask" type="Condition" parent="updateProject.bc7">
  <parameter name="condition1" value="#hds.getTaskID.recordcount# eq 0"
/>
  <parameter name="bclist1" value="bc8,bc6,bc9"/>
  <parameter name="condition2" value="#hds.getTaskID.recordcount# gt 0"
/>
  <parameter name="bclist2" value="bc5,bc9"/>
   </bc>
    <bc name="insertTask" type="SQL" parent="updateProject.bc8">
         <parameter name="method" value=
  "insert into task (tname, tdesc, sdate, uid, pid) values
                    ('#hds.taskName#', '#hds.taskDesc#', #hds.startDate#,
'#hds.userid#', #hds.projectid#)" />
   </bc>
   <bc name="getTasks" type="SQL" parent="updateProject.bc9">
         <parameter name="method" value="select pname as projectName,
tname as
  taskName, sdate as startDate
             from task where uid=#hds.userid# group by projectName" />
   </bc>
</business-components>
```

Listing 4:

```
<cfcomponent displayname="SQL" hint="Indent's generic SQL model compo-
nent" extends="com.indent.model.AbstractModel">
  <cffunction name="execute" access="public">
         <cftry>
                    <cfif not (StructKeyExists(bcparams,"db"))>  <!—get-
ting db parameter ‡
                              <cfif (StructKeyExists(bcdefparams,"db"))>
                                        <cfset dbName = #bcdefparams.db#
/>
                              <cfelse>
                                        <cfthrow type="dbAliasException"
message="Database alias not defined">
                              </cfif>
                    <cfelse>
                              <cfset dbName = #bcparams.db# />
                    </cfif>
                    <cfquery datasource="#dbName#" name="dataset"> <!—
evaluating the query ‡
                              #Evaluate(DE("#bcparams.method#"))#
                    </cfquery>
         <cfcatch type="dbAliasException">
                    <cfthrow type="modelException"
message="#cfcatch.Type#:#cfcatch.Message#">
         </cfcatch>
         <cfcatch type="database">
                    <cfthrow type="modelexception"
message="#cfcatch.Type#:Error in executing the Query:
#Evaluate(DE("#bcparams.method#"))#">
         </cfcatch>
         <cfcatch type="expression">
                    <cfthrow type="modelexception"
message="#cfcatch.Type#:Error in the constructing Query Expression:
#bcparams.method#">
         </cfcatch>
         </cftry>
         <cfset StructDelete(session.hds, "#bcparams.bcName#") />
         <cfset SQLType = LCase(gettoken("#bcparams.method#", 1, " "))
/>
         <cfif SQLType eq "select"> <!—put rows returned by select into
hds ‡
                    <cfset StructInsert(session.hds, "#bcparams.bcName#",
#dataset#) />
         </cfif>
  </cffunction>
</cfcomponent>
```

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

A LIMITED TIME SAVINGS OFFER FROM *SYS-CON MEDIA*

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

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

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

3-Pack
Pick any 3 of our magazines and save up to $275⁰⁰
Pay only $175 for a 1 year subscription plus a FREE CD
- 2 Year – $299.00
- Canada/Mexico – $245.00
- International – $315.00

6-Pack
Pick any 6 of our magazines and save up to $350⁰⁰
Pay only $395 for a 1 year subscription plus 2 FREE CDs
- 2 Year – $669.00
- Canada/Mexico – $555.00
- International – $710.00

9-Pack
Pick 9 of our magazines and save up to $400⁰⁰
Pay only $495 for a 1 year subscription plus 3 FREE CDs
- 2 Year – $839.00
- Canada/Mexico – $695.00
- International – $890.00

CALL TODAY! 888-303-5282

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

☐ 3-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 6-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 9-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.

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

☐ **MX Developer's Journal**

U.S. - Two Years (24) Cover: $143	You Pay: $49.99 /	Save: $167 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $29.99 /	Save: $60
Can/Mex - Two Years (24) $168	You Pay: $79.99 /	Save: $137 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $49.99 /	Save: $40
Int'l - Two Years (24) $216	You Pay: $89.99 /	Save: $127 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $59.99 /	Save: $30
Digital Edition - One Year (12)	You Pay: $19.99	

☐ **Linux World Magazine**

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

☐ **Java Developer's Journal**

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

☐ **Web Services Journal**

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

☐ **.NET Developer's Journal**

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

☐ **XML-Journal**

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

☐ **WebLogic Developer's Journal**

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

☐ **ColdFusion Developer's Journal**

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

☐ **Wireless Business & Technology**

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

☐ **WebSphere Developer's Journal**

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

☐ **PowerBuilder Developer's Journal**

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

*WHILE SUPPILES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

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

SYS-CON MEDIA

Code Co-op Version Control Software from Reliable Software

Version control: We all need it

By Selene Bainum

Question: Who needs version control?

Answer: Every developer. Most people think of versioning control software as something that should be left to big companies and large teams. However, version control is a good idea for everyone.

Even if you are working on an application by yourself, it is too easy to lose important changes forever, because you accidentally saved over a document. Not to mention how nice it would be to go back to see an earlier version of a file. Besides, how many files can you have in your directory that end in "_backup.cfm" or ".old"?

There are several major benefits to version control software, for teams that range in size from 1 to 100: retaining a history of files; being able to roll back an earlier version of a file; marking or labeling a set of files for a particular release; ensuring a file is being worked on only by one person at a time; and keeping track of who did what.

Comparing the Options

Typically, version control software can be expensive – such as Visual Source Safe (VSS) from Microsoft or Merant Version Manager. Others, like CVS, are free (support not included). The features of these products vary, and your particular needs may determine what you are already using – or not using.

Code Co-op, from Reliable Software, is a lower-cost version control software, that they refer to as "The Peer-to-Peer Version Control Software System for Distributed Teams." The comparison matrix between Code Co-op, VSS, and CVS certainly shows that Code Co-op is not short on features. All the things that you expect to see are there: integration using SCC API, support of all file types, parallel development, visual differencing, merging, restoring from file history, reporting, branching, and change notification. You can even collaborate with your team through e-mail. However, Code Co-op runs only on Windows-based systems so Unix/Linux users will have to stick with CVS.

From my experience with development teams, there are several ways to organize code:

- The code rests in a centralized location and developers all work from that code base. Any changes made are immediately available to the entire team, for better or for worse. This method is good if you don't want to have to worry about synching local development environments with the main repository, but is bad if someone is breaking something that affects the entire system.
- The code rests in a centralized location, but each developer has a local development environment with copies of the code. The centralized location is updated only when a developer checks files back in. This is very helpful in that developers can test on their own environment before synchning files with the main repository, but the team must remember to get the latest version of files or synch in order to make sure their environment is up to date.
- Each developer has his or her own code base and there really isn't one centralized location in which the files are stored.

The type of environment your organization wishes to follow will help narrow down your version control software choices. For instance VSS works with the first two environments while Code Co-op works with the last two.

Getting Started

To get started, you install Code Co-op on all the machines that will be accessing projects. Machines can be either voters (can check files in/out, approve scripts, etc...)

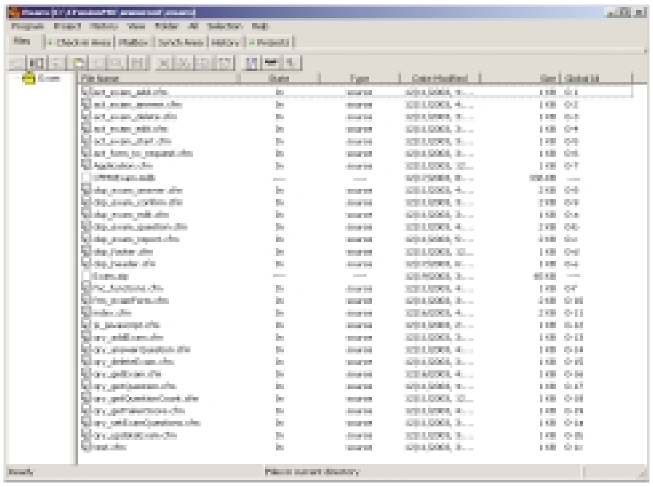
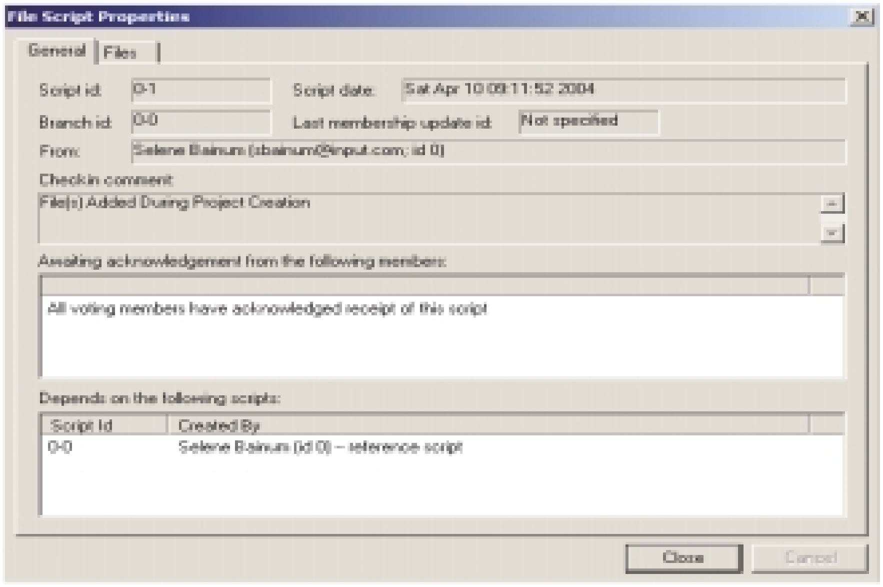


Figure 1: Files view



Script id: 0-1
Script date: Sat Apr 10 08:11:52 2004
Branch id: 0-0
Last membership update id: Not specified
From: Selene Bainum (sbainum@input.com; id 0)
Checkin comment:
File(s) Added During Project Creation
Awaiting acknowledgement from the following members:
All voting members have acknowledged receipt of this script
Depends on the following scripts:

Script Id	Created By
0-0	Selene Bainum (id 0) – reference script

Figure 2: File script properties

VITALS

Code Co-op: Peer-to-Peer Version Control
Reliable Software

Address:

1011 Boren Avenue PMB 206
Seattle, WA 98104

Phone: (206) 361-6679
Fax: (206) 367-6085

E-mail: info@relisoft.com

Web: www. relisoft.com

Pricing: $159/seat for 1–10 seats; $125/seat after 10. There is also a 31-day free evaluation and reduced upgrade prices.

Product Snapshot Target Audience: Developers looking at good entry-level versioning control software that has most of the features of other packages with a lower cost and free support options, yet don't require a centralized application database or project administration.

or observers (only receives synchronized scripts). Observer machines don't need paid licenses, which is a good thing. Synchronization can occur two ways: via e-mail or via a LAN. You can even set up for both, though e-mail is the easiest way to get started.

Each member of the team has a copy of the code base on his or her computer. Code Co-op controls the properties of the files (read-only versus writable) on that computer only. When a developer wants to make changes, he or she checks the file(s) out and gets to work. After files are checked back in, synching scripts will be sent to the other members of the project. Those members can then view the changes, accept/reject them, and merge them into their files. The only way to have a centralized code base is to install Code Co-op on another machine (such as a development/staging server) and have it act as an observer to automatically accept all scripts. That way there is a set of code that can be backed up or used as a way for anyone to access the code from a centralized location.

Distributed Code Base Model

There are several benefits to a distributed code base model, such as being able to check files in/out very quickly regardless of network connectivity status – you can do it without being connected at all. If anyone has ever tried using VSS over a remote connection – even a high-speed VPN, you know what a help this can be. It also eliminates the problem of another developer introducing a bug that affects you while he or she is developing. At least you would hope that members of your team would do some sort of tests before checking files in and sending out synch scripts, but we all know how well we test our own code.

There are drawbacks to this model as well, the largest being that there is no way to see an overall project snapshot. Records of a developer having a file checked out is only on their machine, there is no centralized reporting – though Reliable Software hopes to add that feature to the next version. Reliable Software touts the absence of an application database on its comparison matrix as an advantage to both VSS and CVS; however, that is a matter of opinion. A centralized database/administration tool gives you that project snapshot as well as many other features. Users are not tied to a computer, as you can usually log in from any computer with the source control software installed, and you aren't at a loss when the project administrator's computer fails along with all the project information.

"The type of environment your organization wishes to follow will help narrow down your version control software choices"

User Experience

While Code Co-op is less expensive than products like VSS, you would expect a better user interface for $159. The interface has the feel of a shareware application as opposed to a commercial product that is aiming to play with the big boys. For instance, there are three ways to perform actions: using a pull-down menu, using the menu bar buttons, and right-clicking on an item and selecting an option. However, the options available differ among the three ways to access them, making it difficult to know exactly how to perform a particular action. The feature set is solid; you just have to learn how to use it.

Community-Based Support

Those of us familiar with Allaire/Macromedia's support forums will certainly appreciate Reliable Software's support options. There is a free support forum that is moderated by employees – even on the weekends. My queries for help were responded to quickly, even on Easter Sunday.

– continued on page 39

— continued from page 7

ColdFusion MX introduced the URLSessionFormat() function that will append the appropriate token(s) to a URL for any user that has cookies disabled – I highly recommend using it. The third thing to be aware of is that, aside from using a service factory hack, there is no way to access any session other than the current session in your CFML pages. The workaround for this is to track all of the active sessions in the application scope, identifying each session by a unique token (such as its JSESSIONID). Accessing data about a user is done by simply referencing "#session.some_user_property#" where "some_user_property" is the name of the session variable that holds the desired bit of information unique to the current user.

There is also the topic of security – how to restrict access to functionality and/or data based on who is currently logged in to the application. Obviously, you could store some information about the user in the session scope and use conditional logic to determine whether or not a user is allowed to see a specific bit of information or execute a specific block of code, but ColdFusion MX introduced a better way to do this. These new tags are <CFLOGIN>, <CFLOGINUSER>, and <CFLOGOUT>.

I am not going to get into the nitty-gritty details of these tags, but the idea is this: on every request (in the Application.cfm file) a <CFLOGIN> block is executed. If a user is not currently logged in, that block will display a login form or do nothing depending on your business requirements. When a user submits a login form, the code in the <CFLOGIN> block compares the form login credentials (usually a username and password) with a repository such as a database. If the credentials are OK, a <CFLOGINUSER> tag logs the user into the application and the <CFLOGIN> block is ignored for all subsequent requests by that user (until they log out). When the user wants to log out (he or she clicks a button or link), the application executes a <CFLOGOUT> tag, and the user is no longer logged in. If the application server is using J2EE sessions, closing the browser window will also log the user out. If J2EE sessions aren't used, it is important to keep in mind that when a user is logged out of a ColdFusion application, his or her session is still active. What about restricting access to resources and/or data?

Earlier, I mentioned that users log in to an application in order to be identified and I vaguely described the way in which login can be forced and a user is flagged as "logged in" by using the <CFLOGINUSER> tag. In addition to identifying a user by username and password (which are passed to the <CFLOGINUSER> tag and typically come from an external repository such as a database or LDAP server), a user's "roles" are also specified at login time.

Roles offer a way in which users can be categorized – these categories are used to determine permissions in an application. Like any other variable, the names of roles are user-defined when an application is developed. A medical application may have roles "doctor, nurse, and patient" and a content management application might have roles "administrator, copy editor, legal, editor, and end user." The point is, it's up to a developer to analyze business requirements and use cases, and determine what and how many roles are required in an application. Roles are assigned to a user by passing all of the roles that a user is a member of as a comma-delimited list to the <CFLOGINUSER> tag (via its "roles" attribute).

There are two significant ways that roles can be used to control access to data and functionality in an application. The first is by using conditional logic to test whether or not a user is in a role or a list of roles. The #isUserInRole()# function returns true or false, indicating whether or not the current user is in the role name passed to the function. If a list of role names is passed to the isUserInRole() function, the function returns true if, and only if, the user is in all of the roles in the list. The second way that an application can use roles to secure functionality and data requires the use of ColdFusion Components. If you specify the name of a role or a list of roles in the "roles" attribute of the <CFFUNCTION> tag within the definition of a CFC function, only users in that role are able to execute that function. An exception will be thrown if any user not in the list of allowed roles attempts to execute a function. Applications don't typically need to display an error message when users attempt to access a function that they are not authorized to execute, but by surrounding method calls with a <CFTRY>/<CFCATCH> block, control of flow and UI elements can be regulated.

As I said before, developers could write all of the code and create their own session variables and mimic the same behavior without using the <CFLOGIN>, <CFLOGINUSER>, and <CFLOGOUT> tags, but there is one other advantage that these tags offer. ColdFusion uses the JAAS framework (the industry standard for Java security), which means that a user that is flagged as logged in and assigned roles by a Java application is also seen as authenticated by ColdFusion, and vice versa.

Security is an important aspect of every application, as is the ability to store user-specific data – especially when creating personalized applications. By thoughtful use of the session scope and the security tags introduced in ColdFusion MX, developers can build ColdFusion applications that are both secure and flexible. I recommend regularly checking the materials released by Macromedia on their Web site and their DevNet Resource Kits for white papers and code examples regarding the development of personalized and secure Web applications.

Changing subjects briefly, I also wanted to take a moment to mention that Macromedia has officially announced the dates and location of MAX 2004. The conference will be held in New Orleans on November 1-4. Yes, that's right, I said New Orleans... yahoo! I urge all of you to begin making room in your training budgets and/or persuading your bosses to send you now. With the anticipated release of the next version of ColdFusion, the recent release of Flex, and all of the other exciting products and releases over the past six months and anticipated in the next several months, it ought to make for an awesome conference. I plan to fly in at least one day early. After all, what could be better than Halloween in New Orleans to kick off the conference? Whether you go a day early for Halloween or not, I hope to see all of you at MAX 2004...

product review

– continued from page 37

The company also provides an e-mail address directly to their support team. The responses feel helpful and informal, letting you know that the people developing this product are on the same level as you, which is a nice feeling.

Is Code Co-op Right for You?

Like most applications and collaboration tools, many factors must go into determining which version control software is right for your organization. Teams looking to get started with version control can look to Code Co-op as a way to get most of the features of the other major packages at a lower price and with easier configuration/installation. Organizations that already use systems such as VSS will feel limited by the lack of centralized administration, but not everyone needs that. Code Co-op would be a life saver for smaller teams that need source control. The price is low enough that it is even well worth the investment for individual developers who are tired of overwriting saved work.

About the Author

Selene Bainum created and maintains webtricks.com, a premier ColdFusion tutorial site. She is currently a senior Web developer with INPUT.

selene@webtricks.com

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
21C JVISGENWIZARDS	WWW.JVISGENWIZARDS.ORG	650-641-1744	33
ACTIVEPDF	WWW.ACTIVEPDF.COM	866.GoToPDF	4
CFDYNAMICS	WWW.CFDYNAMICS.COM	866.233.9626	6
CLEARNOVA	WWW.CLEARNOVA.COM/THINKCAP		15
EDGE WEB HOSTING	WWW.EDGEWEBHOSTING.NET	1.866.EDGEWEB	Cover II
HAL HELMS, INC	WWW.HALHELMS.COM		17
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM/CFDJ	877.248.HOST	17
INTERAKT ONLINE	WWW.INTERAKTONLINE.COM		Cover III
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800.379.7729	Cover IV
MACROMEDIA	WWW.GULFSTREAM.COM	415-252-2000	12-13
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	21
TERATECH	WWW.CFCONF.ORG/CFUN-04/	301-424-3903	49
WINMILL SOFTWARE	WWW.WINMILL.COM	888-711-6455	3

On ColdFusion and Flex

What Flex can do for you

By Ben Forta

Macromedia Flex is a brand new server product, one that is poised to forever change the way we coders think about building rich and engaging user interfaces.

Yes, I did say "server." No, Flex does not compete with ColdFusion. Yes, Flex and ColdFusion are designed to work together.

Having now dispensed with the obligatory preliminaries, I'd like to present Macromedia Flex to the ColdFusion audience from a distinctly ColdFusion angle. And so this month I will:

- Explain exactly what Flex is, from a ColdFusion user's perspective.
- Discuss the ColdFusion/Flex relationship.

So, What Exactly Is Flex?

The best way to understand Flex is to think about how you build Web applications right now. If you are a ColdFusion developer (or a developer who uses ASP, JSP, PHP, and the like) you probably build applications comprised of lots of linked pages. These pages contain client code (at a minimum, HTML and JavaScript) as well as server-side code (CFML, SQL, and so forth). You probably try to separate content (or back-end data) from presentation (the front-end UI), but you still use both to create your application.

So, in the case of a ColdFusion page, a browser requests a CFML file, ColdFusion parses and processes the CFML in that file, and then returns client code to the browser. The browser renders the client code on the client, but it is ColdFusion on the server that embeds that client code in the page. Put differently, ColdFusion is a server that dynamically generates client-side code.

So why all this analysis? Simply put, Flex works in much the same way. Whereas ColdFusion (despite being somewhat client independent and agnostic) generates client-side HTML, Flex generates client-side Macromedia Flash SWF files. Perhaps that's a bit too simplistic, but the analogy is sound: ColdFusion uses plain text files containing CFML tags to render browser content, and Flex uses text files containing MXML tags to render Macromedia Flash content. Compare the flows of both processes in Figures 1 and 2.

But that is where the similarities end. Unlike ColdFusion, Flex and its language, MXML, do not provide tags for database integration, sending and receiving e-mail, connecting to LDAP servers, or creating Web services. Whereas ColdFusion is more of an all-purpose application development platform, Flex has a far more focused objective and a very specific goal. Flex is designed to facilitate the creation of rich and engaging presentation layers, letting developers create well designed and optimally architected n-tier applications that adhere to standards and best practices. With Flex, coders can take advantage of the ubiquitous Flash platform from a very code-centric starting point, but what Flex generates is still Flash content. In other words, Flex is all about presentation and user interaction, and that's it.

Flex is actually similar to ColdFusion in one other way. Flex, like ColdFusion, is a Java application. Flex is deployed on top of standard J2EE servers, such as Macromedia JRun, IBM WebSphere, and so forth, in much the same way that you deploy ColdFusion, but more on that soon.

All of the previous discussion means that Flex does not compete with ColdFusion; rather, Flex needs ColdFusion (or Java, or .NET, or something, anything, providing back-end integration). A Flex application needs a back end, and that back end can indeed be ColdFusion (as in Figure 2).

How do ColdFusion and Flex interact? The answer is that it depends. There are two distinct ways to go about using them together.

ColdFusion As a Flex Back End

So, Flex developers write MXML code to render user interfaces, and rely on external systems for any and all transactions, business logic, and back-end processing. How does Flex access these external systems? Flex provides tags that developers can use to:

- Invoke Web services
- Make HTTP requests
- Call remote Java objects

It is worth noting that Flex itself does not really invoke Web services or make HTTP calls or call remote Java objects; Flash does that (possibly via a server-side Flex proxy). Flex generates a complete Flash application – a SWF file that runs in Macromedia Flash player – that makes all the back-end calls.

At this point, ColdFusion/Flash integration becomes obvious, Flex calls ColdFusion pages via HTTP requests, or invokes ColdFusion components (CFCs) as Web services. You can use any ColdFusion code that you can invoke through HTTP or as a Web service within Flex applications. To do this, you don't even have to have ColdFusion installed on the same server as Flex, although that configuration works too.

In Figure 3, the flow works something like this:

1. A user requests a Flex MXML URL.
2. Flex generates a SWF, which it sends to the user.
3. The user's Flash player executes the SWF.
4. The SWF makes HTTP or Web service calls from the player and obtains data.
5. The Flex-generated SWF uses that data on the user's machine.

This type of integration delivers true n-tier applications, and is best suited for actual applications rather than embedded controls, for example.

This integration is also ideal for applications with multiple presentation layers. For example, if you needed a straight HTML version of your application in addition to the more engaging Flex version, you could simply provide multiple presentation interfaces to your back-end code. You would put your application logic, without any UI at all, in ColdFusion components; CFML pages would invoke the CFCs and render the HTML version of the application; and Flex MXML pages would render the Flash version of the application. The bonus is that both use the same back-end code.

ColdFusion-Generated Flex

Earlier I compared ColdFusion rendering client-side HTML to Flex rendering client-side Flash. But the integration just explained is not really the same. When ColdFusion renders client-side HTML, for example, the CFML pages programmatically render the HTML, perhaps even embedding dynamic data at the same time. This is quite different from the integration described in the "ColdFusion as a Flex Back End," section where ColdFusion does not generate client-side Flash, it simply provides data to Flash, generated by Flex.

Can you use ColdFusion to dynamically generate Flash apps in much the same way as it can HTML? Yes, you can, through the Flex JSP Tag Library interface. Flex comes with JSP tags that you can use inside of JSP pages. JSP tags are much like CFML tags, except that they are written in JSP instead of in CFML. With Flex JSP tags, developers programmatically embed Flex tags such as the Tree tags or the Grid tag in their JSP pages so that when a user requests a JSP URL, the generated page has Flash embedded within it.

How does this help ColdFusion developers? ColdFusion code can call JSP tags too. CFML has a tag named <CFIMPORT> that developers use to import or include JSP tags; once imported, developers can use these tags like any other ColdFusion tag.

To be able to do this, ColdFusion and Flex must be installed on the same J2EE

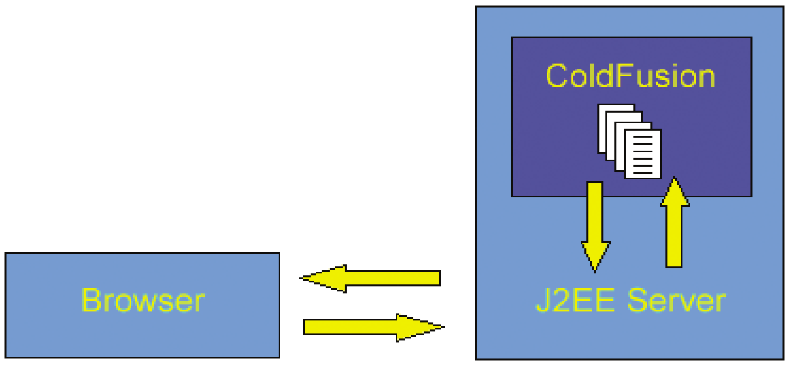


Figure 1: A typical ColdFusion application flow

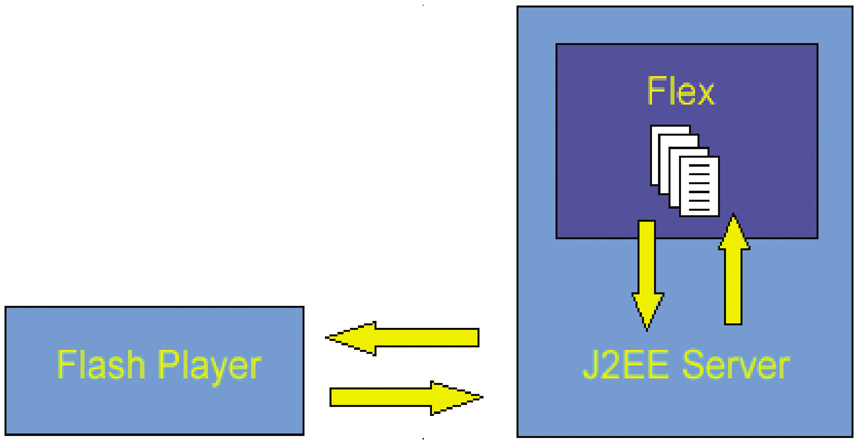


Figure 2: A typical Flex application flow

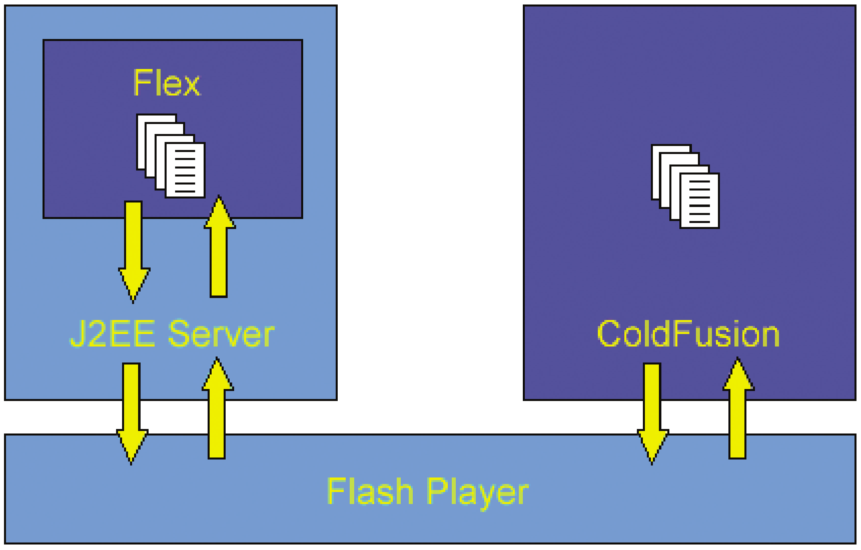


Figure 3: Application flow with ColdFusion as a Flex back end

server. While this installation process is not terribly difficult, it's a manual process. This obviously requires ColdFusion Enterprise, and will not work with a Standalone ColdFusion installation, since the Standalone ColdFusion installation does not expose a J2EE server that could run Flex.

Figure 4 shows the workflow:

1. A user requests a ColdFusion CFML URL.
2. ColdFusion imports the Flex JSP tags.
3. ColdFusion generates a page, which contains both HTML and a Flex-generated SWF, which it sends to the client.
4. The page displays in the user's browser; an embedded SWF executes in the user's Flash player.

This type of integration is best for dynamically generated content, such as a dynamically generated and populated tree control. This type of integration is less suitable for full-blown applications.

Where to Use Flex

ColdFusion developers can think of Flex as being a replacement or an extension for the parts of the application that they'd have written in HTML, JavaScript, DHTML, and so forth. But that does not mean that you should write every HTML application in Flex. In fact, while some applications are great potential Flex candidates, others are decidedly not.

- Flex is not intended to be used to replace specific form controls. For example, if you have an HTML form and use a pop-up calendar, do not replace the calendar with Flex code. The cost, the overhead, the bandwidth usage – none of those factors would make Flex an efficient option. Using Flex for whole parts of a form probably does not make sense. However, if you have a complex multipart form with lots of screens, form controls, and dynamic elements, then yes, Flex could be a good candidate.
- Applications involving raw reporting, data browsing, or working with large amounts of data – where the emphasis is less on the presentation and more on raw content – are also not Flex's sweet spot. While Flex provides controls, such as the data grid, that you can use within these applications, the overhead, performance, and amount of coding needed may not be a fit for Flex.
- Flex is not designed to replace Flash. If you use Flash for complex animation or sophisticated design work, you'll still use Flash.
- Where Flex shines is in rich presentation of information, with an emphasis on "rich." Dashboard type applications are perfect for Flex, as are innovative ways to display and interact with specific data, such as shopping carts.

Of course, these are high-level generalizations, and there is no hard-and-fast rule here. The key is that Flex empowers developers to build rich and highly interactive user experiences. It is best suited for applications where this type of interaction is a requirement and where the project plan can accommodate the overhead associated with this type of experience.

In case I did not state this clearly enough, Flex does not replace Flash. In fact, almost anything that can be done in Flex can be done in Flash too; the major difference is the development model (including code management, the use of design patterns and best practices, team development, versioning and editioning, and more). As a Macromedia developer, you now have one more weapon in your arsenal.

Where to Go from Here

To learn more about Flex, and to

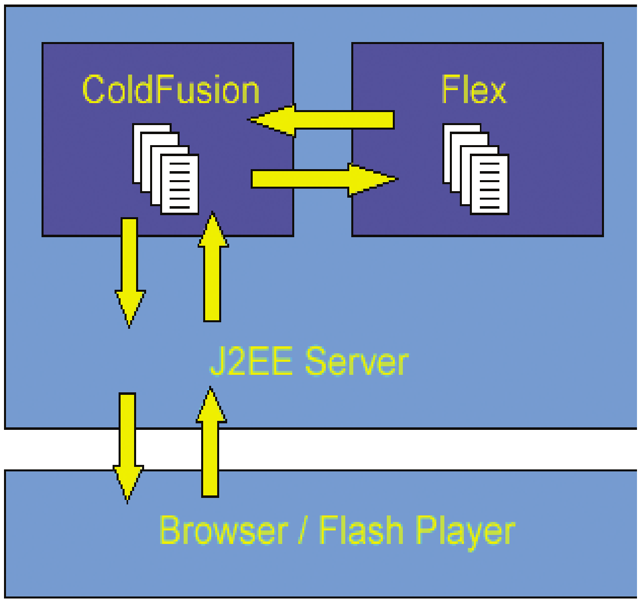


Figure 4: Application flow for ColdFusion-generated Flex

obtain a copy of the Developer Edition, visit www.macromedia.com/software/flex/. Also visit www.macromedia.com/devnet/flex/ for articles, columns, and example applications (including ColdFusion-related content).

In addition to overviews, positioning, and technical content, you'll also find practical and honest discussions on Flex itself to help you determine if it is right for you and your projects. Macromedia wants to help you build a better user experience, and that may involve Flash, it may involve Flex, and it may involve both. Whatever it is, we'll help you get there.

Conclusion

Flex empowers code-oriented developers, allowing us to build rich and engaging user experiences using development models and principals that we can understand and relate to. Flex is new, and still evolving, but even now it can have a dramatic impact on your development, both what you develop, and how. Flex may be right for your project, and it may not, but either way it pays to have a basic understanding of what this new technology is, and of what it can do for you.

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

– continued from page 25

Adventures in Encapsulation PART IV

JOHN: I suppose. But really, this guy's so lame I don't think I have anything to worry about. Besides, he doesn't know me as anything other than my screen name!
HAL: Ah, well there's that.
JOHN: Well, look, you take it easy and thanks again for helping me. And remember – no articles out of this!
HAL: John, you have my solemn promise. And John? You have a *great* day…

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

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

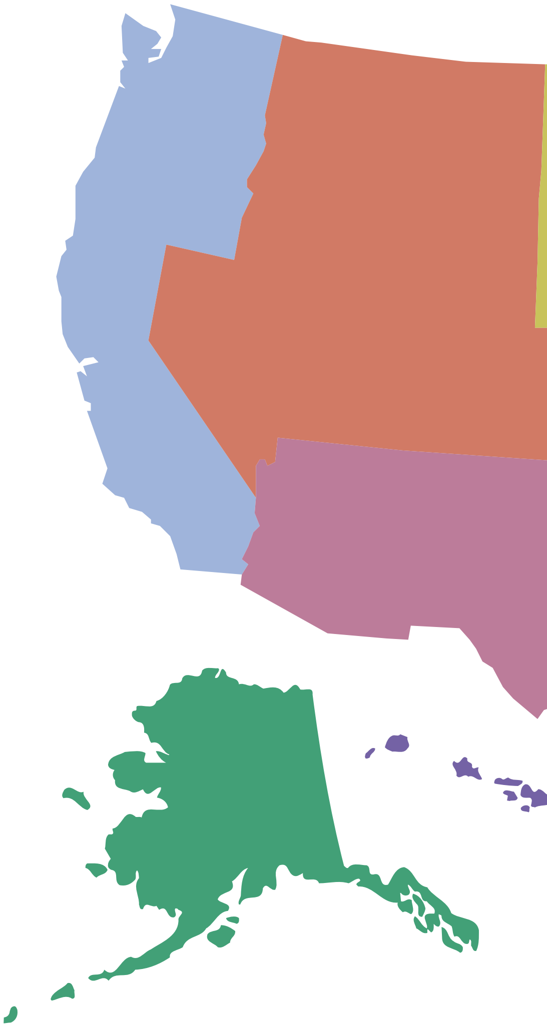
Missouri
O'Fallon
St. Louis, MO CFUG
www.stlmmug.com/

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

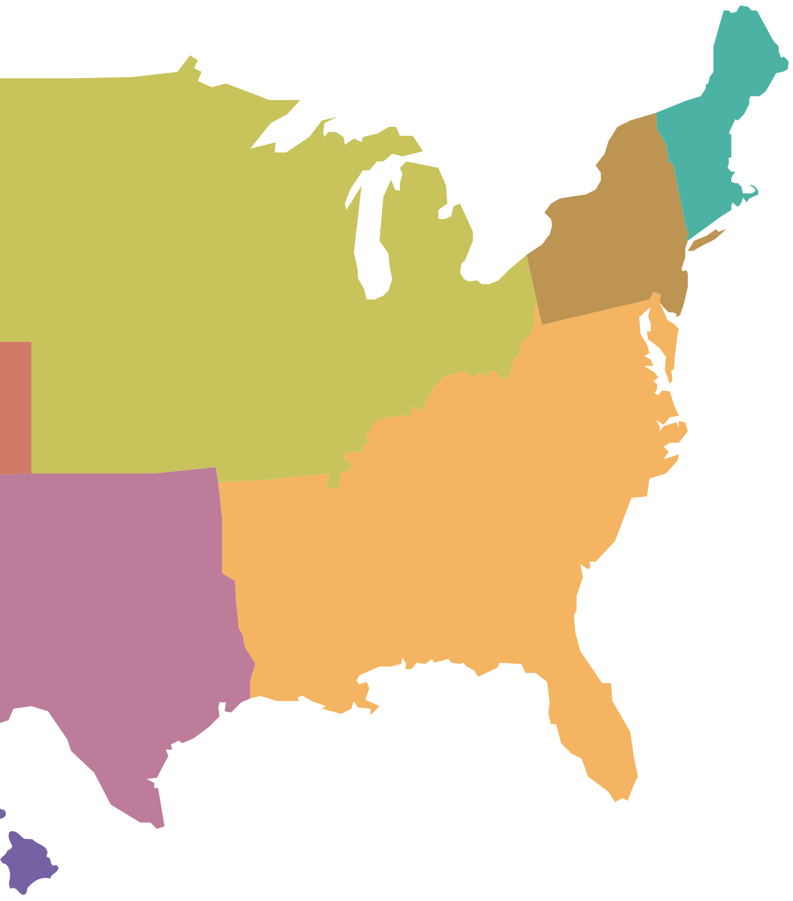
Oregon
Portland
Portland, OR CFUG
www.pdxcfug.org

Pennsylvania
Carlisle
Central Penn CFUG
www.centralpenncfug.org

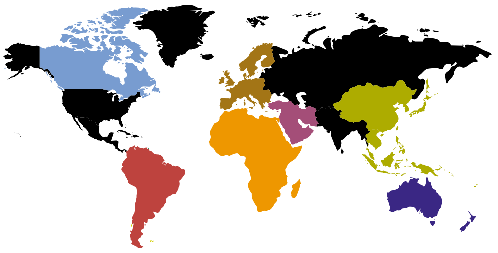
Pennsylvania
Exton
Philadelphia, PA CFUG
www.phillycfug.org/

User Groups

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



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

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.

Structures and Arrays PART 1

Use arrays for processing numbers and spreadsheet-related functions

By Jeffry Houser

This is the first of a two-part column in which I'll teach you about structures and arrays, two of the complex data types in ColdFusion. To start, I'll introduce structures and arrays and then go into some details on how to make use of arrays in ColdFusion.

Next month I'll go into detail on structures and then compare and contrast the two. Along the way I'll provide plenty of examples for you to learn from. In short, both structures and arrays give you a way to group a lot of related pieces of data together.

Understanding Structures and Arrays

Structures and arrays are considered complex data types in ColdFusion. Simple data types by contrast are ones that contain a single piece of data, such as a number, string, or Boolean value. A complex data type can contain multiple pieces of data, usually related. All the pieces of data are referenced under a single variable name. Think of a complex variable as a variable that has other variables inside it.

You can access a single data element of an array (a variable inside the array) by using a number. This number is often called the "index" of the array. A single data element of a structure, by comparison, is accessed using a string. If you are familiar with other languages, you may have heard of a structure referred to as an associative array. That is because a string value is used to associate with the single data element. So arrays reference each of the values within themselves with a number (representing their position in the array), and structures reference each of their values by name.

Arrays are great for number crunching since many of the built-in array functions will perform math functions on a set of numbers. They also work well when processing multiple elements of data in a loop, but don't need immediate access to an individual element. An example of this is a shopping cart application. An array is an ideal way to store all the items in a shopping cart. When you add something to the cart, it does not matter where in the cart it ends up. When you go to check out, you are going to process all the items in the cart as a group, and not individually. You pick up an item from the cart, run it through the scanner, and move on to the next one. There is no reason to process one item before the other.

Structures are great when you are going to need to access data elements by name. An example of this is each item residing in your shopping cart. Although it doesn't matter in what order you process the items, when you get an individual item (a value in the array) you'll want easy and direct access to properties of the element such as the price, quantity, name, or UPC value.

ColdFusion does not place any limitations on the data that can be put into an array or structure. It can be simple variables, such as an integer or string; or even complex variables such as another array, structure, or component. In practice, many shopping cart applications use an array of structures to store the data.

Array Dimensions

One major difference between arrays and structures is that arrays have something called "dimension", but structures do not. Most often arrays have one dimension, which is akin to a list of items, similar to a single row in a spreadsheet. Our shopping cart is an example of a one-dimensional array. It contains a list of items, stored sequentially. This is an example of a one-dimensional array:

Index	**Data**
1	Carrots
2	Bread
3	Candy Bar
4	Milk

If you were to access the first element, it would be carrots. The second would be bread. The third would be a candy bar and the fourth, milk. For those who are knowledgeable about arrays in other languages, such as JavaScript, ColdFusion arrays start with an index of 1, not an index of 0.

We can access the individual elements of the array using the name of the array, an open (square) bracket '[', the array index we want to access, and a close bracket, ']'. If we had created an array called MyArray, we could access the third element of the array like this:

```
MyArray[3]
```

Using the sample array above we would get the value candy bar.

Arrays can also have two dimensions, similar to a spreadsheet with multiple rows. One dimension specifies the row and another dimension specifies the column. Sometimes it's good to think of a two-dimensional array as an array inside an array. This is an example of a two-dimensional array.

Index	1	2
1	Carrots	Beans
2	Bread	Flour
3	Candy Bar	Butter
4	Milk	Salt

In the two-dimensional array you must retrieve data elements using a dual index, specifying the value you want in the first and second dimension. Each index is specified using the square brackets, just as if you were referencing a single-dimension array. This is an example:

```
MyArray[1][1]
```

This retrieves the value at row 1, column 1 (carrots). If you want to retrieve row 1, column 2, you'll get beans. Row 2, column 1 will give you bread, and so on.

One way to think of two-dimensional arrays is as a one-dimensional array, where each data element of the array is another array. In the array above, if we were to retrieve the first element of the two-dimensional array, we'd receive this one-dimensional array:

Index	Data
1	Carrots
2	Beans

Using ColdFusion's array creation functionality we can create arrays with up to three dimensions. If you think of 2D arrays like a spreadsheet, a 3D array would be like a cube. You can create arrays beyond three dimensions, but they become hard to conceptualize and are beyond the scope of this article. You can read more about them at http://livedocs.macromedia.com/coldfusion/6.1/htmldocs/arrayst8.htm.

Creating and Using Arrays

It's time to write some code. Our first step is to create an array. Simple variables, such as strings or numbers, don't require any sort of declaration before use, but arrays do. We can use the cfset tag in combination with the ArrayNew function to define the array (see this column in the February 2004 issue for more on CFSET). The function has one parameter, which defines the number of dimensions of the array. As previously discussed, we can enter a value of 1, 2, or 3. This will create a one-dimensional array:

```
<cfset MyArray = ArrayNew(1)>
```

Now we have an empty one-dimensional array. The next step is to add some elements to it. We can access the individual elements of the array using the name of the array, MyArray, an open (square) bracket '[', the array index we want to access, and a close bracket, ']'. This will come across more clearly with an example. We can set an array value like this:

```
<cfset MyArray[1] = "Carrots">
```

The value can be output on a page like this:

```
<cfoutput>#MyArray[1]#</cfoutput>
```

This is fine when ColdFusion is accessing a single dimensional array as in our current example, but I wanted to make sure you knew how to apply this to multiple dimensions. Just add on another set of brackets and the number for the second index. To access the first value of the first array, you can use this:

```
MyArray[1][1]
```

To access array values from arrays with other dimensions, you can just tack on additional indexes until you reach your value. A three-dimensional array can be accessed like this:

```
MyArray[1][1][1]
```

In a multidimensional array, as you add a new index to a dimension, it is immediately created as an array. Back to our one-dimensional array example, this code will finish creating our example array from above:

```
<cfset MyArray[2] = "Bread">
<cfset MyArray[3] = "Candy Bar">
<cfset MyArray[4] = "Milk">
```

It's worth noting here how ColdFusion handles arrays internally. They are sized automatically as you add and remove elements. When you create an array with ArrayNew, there are no elements in it yet. As you start to add elements, ColdFusion will grow the array. When you remove elements ColdFusion will automatically shrink the array. This helps save memory because you aren't storing empty data references. As you add and remove elements, the index of existing elements in an array may change. Do not use arrays in situations where this will cause a problem.

Now you have a variable, MyArray, which refers to a one-dimensional array. The array has four elements. As discussed before, one of the primary reasons to use an array is to perform processing on all elements. The simplest form of processing is to output something. We can loop over an array using the cfloop tag and an index loop. I'll probably dedicate a future column to the cfloop tag; however you can read about index loops at http://livedocs.macromedia.com/coldfusion/6.1/htmldocs/tags-p77.htm#wp1101087. Index loops are used to loop a specific number of iterations. In this case we want to loop from 1 to the length of the array.

You'll need three attributes to the cfloop tag to create an index loop for looping over all elements of the array. The first is the "index". The index specifies the name of a variable that will contain the current loop iteration. The "from" attribute specifies the first value of the index (which is "1" to loop from the first element of an array). The "to" attribute specifies the last value of the index before looping terminates. Unless otherwise specified (with the "step" attribute) the increment each time through the loop will be +1.

For the "index" we can use any valid variable name, such as "ArrayIndex". To start processing at the first index of the array, set the "from" attribute to "1". You'll need to know the size of the array to loop over all of the elements. ColdFusion provides a function, ArrayLen. It accepts one attribute, the array you want the size of. In multidimensional arrays, it will only specify the size of the first dimension. This is the code to loop through the array and display its values:

```
<cfloop index="ArrayIndex" from="1"
             to="#ArrayLen(MyArray)#">
 <cfoutput>#MyArray[ArrayIndex]#</cfoutput><br>
</cfloop>
```

If you put all the code segments from this section into a single template and load it in your browser you'll see each element of the array displayed to the browser.

Array Functions

The previous section introduced you to creating arrays with the ArrayNew function and finding the length of an array with the ArrayLen function. There are many other functions that can be used in conjunction with arrays. Here are a few of the more common ones:

- ***IsArray:*** The IsArray function will check whether or not a given value is a valid array. It accepts two parameters – the array and the dimension.
- ***ArrayIsEmpty:*** The ArrayIsEmpty function accepts an array as a parameter and returns true if the array does not yet have any elements in it. It returns false if it does.
- ***ArrayAppend:*** The ArrayAppend function is used to add an element to the end of the array. Its parameters are the array and the value you want to add.
- ***ArrayPrepend:*** The ArrayAppend function is used to add an element to the beginning of the array. Its parameters are the array and the value you want to add.
- ***ArrayInsertAt:*** The ArrayInsertAt function allows you to add an element to any point in a given array that you specify. Its arguments are the array, the position in which you want to insert your value, and the value that you want to add.
- ***ArrayDeleteAt:*** The ArrayDeleteAt function allows you to remove an element at any specified point in the array. Its arguments are the array, the position you want to delete your value from, and the value that you want to add.
- ***ArraySum:*** The ArraySum function returns the sum of all elements from the array. It accepts a single argument, the array.
- ***ArrayAvg:*** The ArrayAvg function will return the average of all elements in the array. Its single parameter is the array.
- ***ArrayMax:*** The ArrayMax function will return the largest value in the array. It accepts a single parameter – the array. It will work only on arrays that contain numeric data.
- ***ArrayMin:*** The ArrayMin function returns the smallest value in the array. Its sole parameter is the array. It will work only on arrays that contain numeric data.
- ***ArraySort:*** The ArraySort function will sort the elements in an array. It returns a true value if the array was sorted, false otherwise. Its parameters are the array, the sort type (numeric, text, or textnocase), and the sort order (asc or desc).
- ***ArrayClear:*** The ArrayClear variable will delete all data in an array.

This is not a complete list of functions, but it does contain some of the more common ones. A complete list is located in the Macromedia documentation, at http://livedocs.macromedia.com/coldfusion/6.1/htmldocs/functio3.htm#wp3473387.

Conclusion

That's all there is to say about arrays (in the length given in this article). You can read more about arrays in the Macromedia documentation at http://livedocs.macromedia.com/coldfusion/6.1/htmldocs/arrayst2.htm#wp1096146. Because of the built-in functions, arrays are often good for processing numbers and spreadsheet-related functions. Next month I'll talk about structures, the other complex variable.

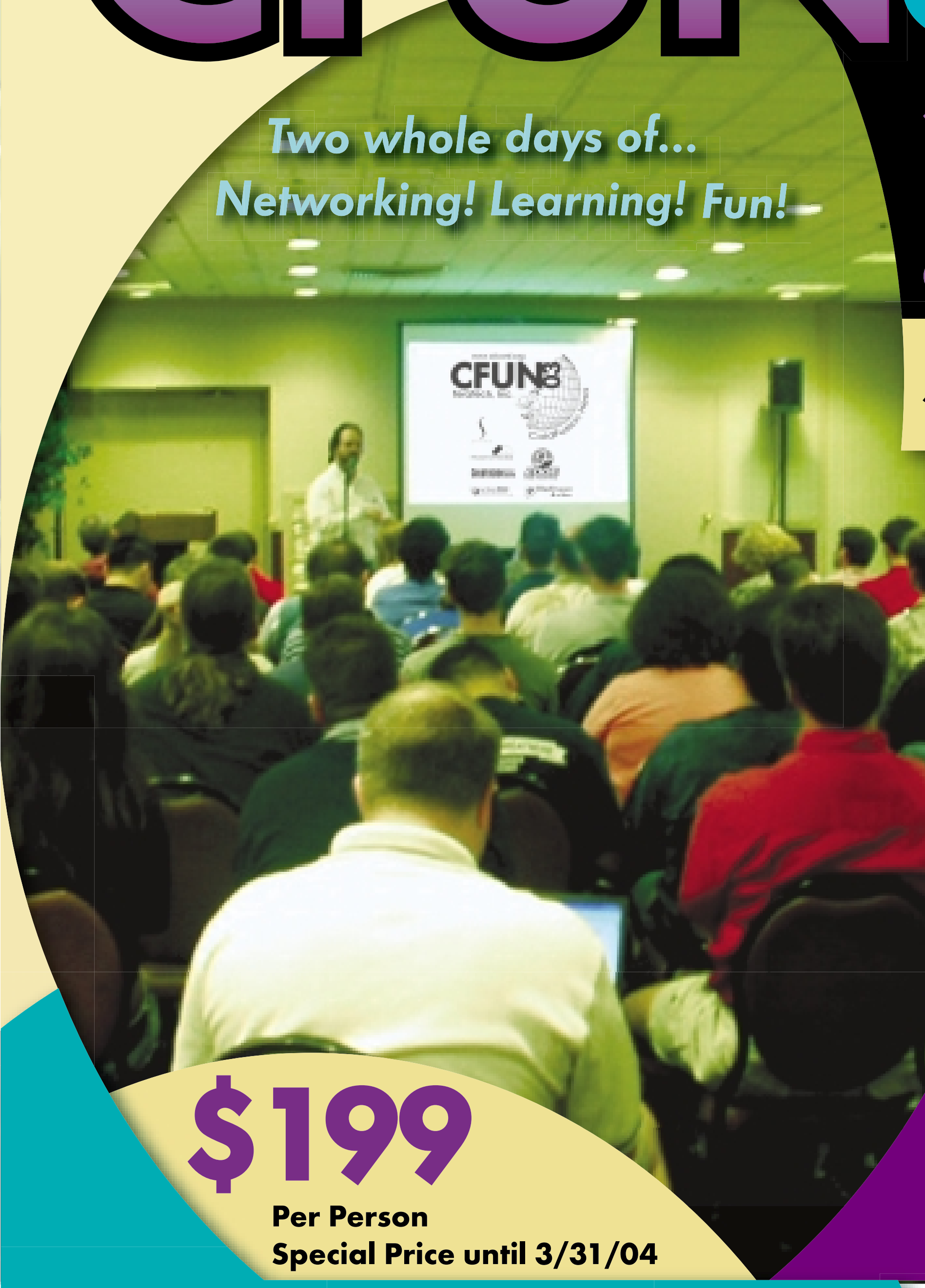
About the Author

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

CFUN 04

301.424.3903
info@teratech.com



Two whole days of...
Networking! Learning! Fun!

Sixth Annual ColdFusion Conference

June 26th & 27th, 2004

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

Macromedia Flex Now Available

(*San Francisco*) – Macromedia Flex is a new presentation server and application framework that enables enterprise development teams to put more effective interfaces on critical business applications. The Flex server offers a standards-based, declarative programming methodology for delivering rich user experiences via the ubiquitous Macromedia Flash Player. These experiences, known as rich Internet applications, combine the rich user interface of desktop software with the reach and ease of deployment of the Web. Partners and developers have already found strategic uses for Flex-based rich Internet applications. “Everyone knows the limitations of HTML for application user interfaces, but until now, there hasn’t been an alternative that works for enterprise development teams,” said David Mendels, senior vice president, Macromedia. “Flex is going to create a renaissance in enterprise application development.”

The Flex presentation server is designed to help development teams deliver rich Internet applications in scenarios where traditional page-based HTML applications are inadequate. Examples include rich visual data dashboards, online product selection and configuration tools, and customer self-service applications. These applications are multi-step processes that involve visualization techniques, immediate user feedback, and local processing that are difficult if not impossible to deliver in HTML. The Flex application framework offers an extensible and customizable class library of pre-built components, effects, behaviors, and layout managers for creating more effective experiences for these uses.

The release of Flex represents a key milestone in the emergence of an important new application architecture. This new approach blends the flexibility of services-oriented data access with the superior reach and effectiveness of a cross-platform rich client. The result is applications that are easier to build and maintain, use less bandwidth, deliver more functionality, and run on all leading server and desktop operating systems.

The Flex framework supports open tooling, allowing developers to use their text editor or IDE of choice to create applications that run on the Flex presentation server. Later this year, Macromedia plans to release a new Flex development tool code-named “Brady.” Built on Dreamweaver MX 2004, “Brady” will offer visual layout, code editing, debugging, and data connectivity tools for creating Flex applications.

The Macromedia Flex presentation server runs on leading Java application servers, including IBM WebSphere, BEA WebLogic, Macromedia JRun, and Apache Tomcat. A native .NET version is in beta and is expected to ship later this year. The trial version, which converts to a nonexpiring Developer Edition after 60 days, is available from www.macromedia.com/go/flex. Flex pricing starts at $12,000 for two CPUs and includes annual maintenance. Discounts are available in some regions for government and educational organizations. Flex is offered exclusively through the Macromedia Volume Licensing Program. See www.macromedia.com/go/mvlp for more information. OEM/ISV pricing information is also available at www.macromedia.com/go/flexoem. “Brady” is scheduled for release later this year, and pricing information is not yet available.

For more information, to get a trial version of Flex, or to see rich Internet applications built with Flex, visit www.macromedia.com/go/flex.

Macromedia RoboHelp X5 Adds Microsoft Word 2003 Support

(*San Francisco*) – Macromedia has announced that Macromedia RoboHelp X5 has added support for Microsoft Word 2003, which is part of the Microsoft Office 2003 software family.

RoboHelp X5 customers will now be able to utilize Word 2003 as their authoring environment, import Word 2003 documents into help projects, and generate printed documentation using Word 2003.

“RoboHelp has pioneered many of the innovative technologies that have advanced the help authoring industry,” said Anthony Olivier, senior vice president and general manager, Macromedia. “This integration fits in with our strategy to ensure people can use their authoring tools of choice.”

Macromedia RoboHelp provides the foundation to millions of online help and user assistance systems currently in use around the world. RoboHelp allows developers and technical writers to easily create professional help systems and documentation for desktop and Web-based applications, including .NET and rich Internet applications. RoboHelp X5 is the latest version of the industry standard help authoring tool used by developers and technical writers to create professional help systems and documentation for desktop and Web-based applications. With RoboHelp X5, users can leverage powerful new features including support for XML, PDF import/export, content management, distributed workforces, team authoring capabilities, and JavaHelp 2.0.

For more information on RoboHelp, visit www.macromedia.com/go/robohelp/.

Performance and Stability Boosts for Dreamweaver and Studio Released

(*San Francisco*) – Macromedia has announced the availability of an update for Macromedia Dreamweaver MX 2004 that offers substantial performance and stability improvements over the prior shipping version. With this updater, Dreamweaver MX 2004 increases end-user productivity by enhancing key workflows and improving the performance of the application. The updater is available at no charge at www.macromedia.com/go/updates.

With the release of the Dreamweaver updater, all Macromedia Studio MX 2004 products have been optimized for better quality and performance. New Studio MX 2004 purchases will include all of the updated products. Existing users can download the free individual product updaters online.

Dreamweaver MX 2004 now runs significantly faster, with performance gains of up to 50% on Windows and up to 70% on Mac OS X.

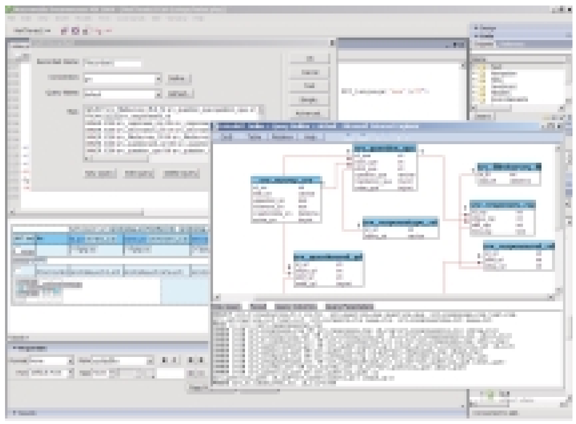
Dreamweaver Website Development

MX Kollection
Professional Dreamweaver Extensions
macromedia DREAMWEAVER 2004

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

macromedia COLDFUSION MX

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™