

COLDFUSION Developer's Journal

ColdFusionJournal.com

August 2002 Volume: 4 Issue: 8

FULL CONFERENCE PROGRAM

WEB SERVICES EDGE
INTERNATIONAL WEB SERVICES
CONFERENCE & EXPO
MILWAUKEE CONVENTION CENTER, SAN JOSE, CA

INTEGRATING APPLICATIONS
CONNECTING ENTERPRISES

JDJ EDGE
CONFERENCE: October 1-3, 2002
EXPO: October 2-3, 2002

CALL TODAY!
800-333-3333
SAVE \$200

INSIDE PAGE 34

Editorial
CF at the Center of the Universe
Robert Diamond page 5

CF Community
Tales from the List
Simon Horwith page 48

CF User Groups
page 46

CFDJ News
Flash Communication Server MX – the newest new thing...
page 50

SYS-CON MEDIA

COLDFUSION PERFORMANCE AND BEYOND

by Jason Clark
page 6

<BF>on<CF>: User Defined Functions – Round 2 <i>A whole new way to create UDFs in ColdFusion</i>	12
ColdFusion&Java: A Cold Cup O’Joe <i>Applets – They’re often the only solution</i>	16
Feature: Beef Up Your Apps with Enterprise JavaBeans <i>The world of n-tier application development</i>	20
CFMX&XML: Transforming XML <i>processing capabilities in a language you can learn easily</i>	26
Feature: Extending the Usefulness of ColdFusion <i>Client and Session Variables These variables may present problems, but they are fixable. Here’s how...</i>	30
Foundations: Understanding the MVC Design Pattern <i>Tackle the big jobs with this slick architecture</i>	36
Journeyman ColdFusion: New Possibilities for Session/Client Variable Handling in CFMX <i>It’s more than just J2EE sessions</i>	38
ColdFusion&Java: Stimulants and Science Fiction <i>Marrying Java with ColdFusion</i>	42

NEW ATLANTA
www.newatlanta.com

RACKSHACK
www.rackshack.net

AUGUST CFDJ 5

COLDFUSION Performance and Beyond

I'm sure you've heard this at some point in your ColdFusion career: "ColdFusion is just a scripting language. It can't handle load." Typically, such statements stem from one of three causes:

- Poor development practices
- Poor application environment design
- Poor choice of platform for the job

ColdFusion can and does handle load, but it takes proper development practice and a thorough examination of the application from end to end.

Most application server platforms have their cynics, those who don't believe in the server's ability to perform under load. And ColdFusion has more than its fair share of skeptics. It's designed to be a rapid development platform, and it's successful at being just that. Unfortunately, the ability to develop Web applications rapidly comes with a substantial risk of people making mistakes because the task at hand is so simple. Ensuring that your application performs well on all fronts requires looking at your application from different angles.

Simultaneous Requests

When a user requests a template, the Web server hands off the page to ColdFusion to process the CFML template. ColdFusion is multi-threaded, so it can process multiple requests simultaneously. The CF Administrator allows you to adjust the number of simultaneous requests that ColdFusion can process with the Simultaneous Requests setting. Let's say you have it set to 5. This essentially means that ColdFusion can spawn five simultaneous worker threads to process requests. Let's go deeper, as tuning simultaneous requests should be at the top of your list.

One of the main goals of a well-made application is to execute any page in the application as quickly as possible. The faster the page executes, the more requests per second your server can deliver. By enabling debugging, you can view the template execution times, but it can throw off your template execution times. Your best bet is to use `GetTickCount()` to track how fast your page or parts of your page are doing. To track a page or code snippet, simply set a start variable equal to `GetTickCount()` at the top of your page or code snippet, then set an end variable equal to `GetTickCount()` at the end of the page/snippet. Subtract the difference to get the number of milliseconds the page took to execute.

One of the most popular methods used to modularize ColdFusion code is custom tags. While these tags provide a way to reuse code, they can also cause performance bottlenecks. When ColdFusion calls a template with a custom tag in it, it essentially parses the calling template up to the point where the tag is called. At this point ColdFusion goes to the custom tag directories to look for the tag being called, finds it, sets up the environment for it, and finally executes it.

The overhead required for this entire procedure can be substantial when there are numerous simultaneous requests. To help reduce this overhead in ColdFusion 4-5x, write your custom tags as `CFINCLUDES` whenever possible. Although the code isn't quite as modular, performance is more important in certain scenarios. If you're using ColdFusion MX, write your custom tag logic into functions within ColdFusion Components, invoke the CFC in your application.cfm, and reuse it as required.

A fresh installation of ColdFusion by default allows only a handful of simultaneous requests. Depending on your application, this default setting is probably too low. While there's no magic formula to determine what the setting should be, proper load testing and a thorough knowledge of the applications residing on the server make setting simultaneous requests easy. In general, a reasonably uncomplicated ColdFusion application that is database intensive will allow you to increase the simultaneous requests substantially. What you need to do is establish a load test using the various load-testing tools available to simulate your expected load. It doesn't hurt to add an extra 20% to compensate for growth.

Some of the free load-testing tools can handle this task well, such as Microsoft's Web application stress testing tool (<http://webtool.rte.microsoft.com/>). Once your load test is running, monitor your processor usage, ColdFusion Requests Running, Queued Requests, and Average request time. You want to process as many simultaneous requests as possible while keeping the average request time at an acceptable level. Acceptable levels vary, depending on your application parameters – fewer than 300ms is usually acceptable. Don't be afraid to use your processors – that's why they're there.

Caching

Caching can provide a very large performance gain for your application. ColdFusion offers different ways to cache various parts of your application. The first and simplest to implement is query caching. If the query you're performing is called numerous times and doesn't change on each request (where statement is consistent), then cache the query using the `CachedWithin` attribute of `CFQUERY`.

Can ColdFusion really handle a substantial amount of load?

The most common question developers ask is how to uncache the cached queries. You can do one of two things. First, rerun the same query with the `CachedWithin` attribute set to "0,0,0,0". This will flush the cache for that specific query. Ensure that the syntax for the query is exactly the same as the cached query since ColdFusion caches queries based on the SQL syntax.

The second, equally easy way to flush the cache is to use `<CFOBJECTCACHE ACTION="CLEAR">`. Note that this tag requires ColdFusion 5+.

Template caching can also save substantial system resources. The first, and often overlooked, type of template caching is Trusted Cache in the ColdFusion Administrator. Enabling this feature allows ColdFusion to pull templates from memory instead of going to disk to check date and time stamps. One caveat: if you make template changes, you have to restart the server on ColdFusion 5 or less; on ColdFusion MX, uncheck Trusted Cache, load the template in your browser, and then recheck the Trusted Cache box. Many other template-caching tags available in Macromedia's Developers Exchange (<http://devex.macromedia.com/developer/gallery/index.cfm>) are worth a look, such as CF.SuperCache developed by Macromedia Consulting.

// In some cases this locking hint will provide large performance gains under load – worth the risk

Database and Query Tuning

Poorly written queries and a lack of indexing are among the leading causes of performance problems in dynamic Web applications. Writing SQL queries is easy, perhaps too easy. When writing your queries, make sure that for every query you write, you've examined any possible indexes, and when executing your query, look at its execution path to see what the database server is doing.

Microsoft's SQL Server provides some of the best tools in the industry to examine queries and ensure that each is as efficient as possible. Some extra hints, however, can be passed in queries to tell the database server what to do when executing your query.

Each database server is different, but here's a trick for SQL Server. Generally, when you issue an update statement to SQL Server, it initiates an exclusive lock on what you're updating. By default, SQL performs page locking, which equals about 8KB of data, so in most cases the lock is locking more than one record or row of data. Once locked, any selects performed against that data are queued until the exclusive lock is released. To get around this, you can provide a hint in your SQL statement to ignore these exclusive locks. Here's an example of a select statement using the NOLOCK hint:

```
select myfield from mytable WITH (NOLOCK)
where myid = 1
```

The NOLOCK command is the same as setting the isolation level for a query in a stored procedure to READ UNCOMMITTED. There can be issues when reading uncommitted data, but in some cases this locking hint will provide large performance gains under load – worth the risk.

An important database-related tag in ColdFusion is CFQUERYPARAM. This tag allows you to use bind variables with the various enterprise-level database platforms available, as well as provide security for your queries. Ben Forta wrote an in-depth article on the use of CFQUERYPARAM ("Faster and Safer Database Queries," *CFDJ*, Vol. 4, issue 2) that's worthy of your attention.

Paging data in Web applications is a common practice to distribute multiple rows of data coming back from the database to the user. On each page *n* rows of data with next and previous buttons are displayed. Depending on how the paging logic is written, the problem with this practice is that the query is written to return all of the rows but the output displays only *n* rows. To increase performance, return as few rows as possible. With SQL Server you can specify the number of rows returned using the TOP command or the ROWCOUNT command in stored procedures. For example:

TOP Command

```
Select top 20 myfield from mytable where
myid = 1 order by myname desc
```

ROWCOUNT Command


```
Set rowcount 20
```

```
Select myfield from mytable where myid = 1
order by myname desc
```

```
Set rowcount 0
```

MACROMEDIA

www.macromedia.com/go/cfmxad



When you return the first page of data, 20 rows of data are displayed. On the next page perform your paging logic to determine the number of rows to return as the user navigates each page of data. MySQL has a similar command called LIMIT that is issued at the bottom of a query. Other database platforms have commands that are similar.

Tuning the User Experience

Many developers overlook the process of what happens after ColdFusion has delivered the template back to the Web server, ready for the user to download. In most cases the resulting HTML is well over 50KB of data. HTTP Compression was introduced as part of the HTTP 1.1 specification to help relieve the problem of increased bandwidth consumption. Most browsers released since early 1999 are capable of receiving compressed content.

When a user requests information from a Web server, the browser sends header information to the Web server. In many modern browsers, part of the header will contain "Accept-Encoding:" and a list of the supported encoding types. Most, if not all, browsers will support the gzip encoding standard. If the Web server can do so, it will then deliver compressed content to that user. After the content is received, the user's browser simply decodes it and displays the page. The user isn't aware of receiving compressed content, but will notice increased performance.

For example, a ColdFusion template that generated 160,542 bytes was compressed to 31,132 bytes – a reduction of over 129,410 bytes. Not only does the user benefit, but your bandwidth consumption could be cut by as much as 50%.

Various software packages are available to compress content, both dynamic and static. Apache 1.x has a free software module, mod_gzip, available at www.remotecom munications.com/apache/mod_gzip/. IIS has HTTP Compression built in, but it currently doesn't work with ColdFusion templates. For IIS, commercial solutions are available from BPVN Technologies (www.pipeboost.com) and HyperSpace (www.ehyperspace.com/).


Load Balancing

So you've written your application, and tested and tuned it for maximum performance, but you're still hitting the limits of your hardware and operating system. Load balancing often scares people away due to the price of some of the commercial load-balancing systems available. ColdFusion Enterprise includes a software clustering solution that works fairly well, but if you're purchasing it just for the clustering solution, you may want to consider building your own hardware-based solution instead.

// **Most load-balancing solutions won't share session variables across a load-balanced cluster"**

The first step in load balancing is to ensure that your application will run in a load-balanced environment. If your application is using session management, you're more than likely going to need to rewrite that part of your application to use client-side management. Most load-balancing solutions won't share session variables across a load-balanced cluster, and although some solutions will, I recommend that you use client-side management. If you use cookies to store information for your application, ensure that all of them are domain-level cookies by using the domain attribute with CFCOOKIE.

What does it cost? For commercial hardware load-balancing solutions, it can range from \$5,000US to \$30,000US. But there are solutions that cost far less and are up to the task. Although these are considered to be "hardware" load-balancing solutions, they're really PCs with operating systems. The Linux Virtual Server project (www.linuxvirtualserver.org/) is a free load-balancing program that runs on the Linux operating system. The LVS software is very robust and can do most, if not all in some cases, of what the commercial systems can do. All that's required is a machine running the Linux operating system and at least one network card to make your own hardware-based load balancer. Having used the LVS solution on ColdFusion-based Web sites that see 2 million page views a day, I can vouch that it can certainly handle load.

ColdFusion has demonstrated that it can handle a substantial amount of load. There's no arguing that fact. It all boils down to writing code that performs well and spending the necessary time to test and tune your application. 

About the Author

Jason Clark, cofounder and CTO of FuseTalk Inc., has more than 10 years of experience in programming specifically the IT sector in general.

@ J A S O N @ F U S E T A L K . C O M

MACROMEDIA

www.macromedia.com/go/cfdjdevcon2002

User Defined Functions – Round 2

BY
BEN
FORTA



A whole new way to create UDFs in ColdFusion

The introduction of user-defined functions (UDFs for short) was the most requested and anticipated event in the ColdFusion 5 era – developers desperately wanted to be able to extend CFML, using not just tags, but functions as well. I first introduced UDFs a year or so ago (*CFDJ*, Vol. 3, issue 5), but now it's time to revisit the subject. Why? Because ColdFusion MX provides us with a whole new way to write UDFs, this time using tags.

Why UDFs?

The ColdFusion Markup Language is made up of two types of instructions:

- **Tags:** Like <CFQUERY> and <CFMAIL>
- **Functions:** Like Now() and StructNew()

Ever since ColdFusion version 2, developers have had ways to write their own tags, and many have done so. After all, writing your own tags allows you to write maintainable and reusable code.

Prior to CF5 we had no way to write our own functions. For example, ColdFusion provides a whole range of list-manipulation functions – ListFirst(), ListGetAt(), ListSort(), and so on – but there was no function to get the smallest or greatest value in a list – ListMin() and ListMax(). To get these values you'd have to loop through the list, doing the comparisons yourself. There were two ways to do this: (1) inline, right in the middle of your code; or (2) by creating a custom tag.

Neither option was ideal. The former wasn't reusable; each time we wanted to obtain these values we had to copy and paste the block of code. The latter was inefficient (custom tags execute more slowly than functions do) and clumsy (because

custom tags have no mechanism by which to return data). The right solution is a user-defined function, which became possible in CF5.

CF5 UDFs

UDFs in CF5 were written in <CFSCRIPT>, ColdFusion's scripting interface. In fact, in CF5 UDF support was the only feature that actually required <CFSCRIPT>. Anything else <CFSCRIPT> could do could also be done in straight CFML.

Here's a simple example of a <CFSCRIPT>-based UDF (this is the example I used a year ago). Suppose you need to use yesterday's or today's date repeatedly within your code. The CFML DateAdd() function can be used to add or subtract a day. To refer to yesterday you could use DateAdd("d", -1, Now()); to refer to tomorrow, DateAdd("d", 1, Now()). These function calls have to be repeated every time they're needed.

Or you could create a couple of UDFs. CFML already has a function, Now(), that returns today's date (and time). Why not Yesterday() and Tomorrow() functions to complement it?

Here's the CF5 UDF code that creates these two new functions:

```
<CFSCRIPT>
// Get yesterday's date
function Yesterday()
{
    return DateAdd("d", -1,
Now());
}

// Get tomorrow's date
function Tomorrow()
{
    return DateAdd("d", 1,
Now());
}
</CFSCRIPT>
```

As you can see, the UDFs are created using CFML scripting between <CFSCRIPT> and </CFSCRIPT> tags. This <CFSCRIPT> block defines two functions, Yesterday and Tomorrow. Each function is preceded by the keyword *function* so <CFSCRIPT> knows we're about to define a function.

These functions are very simple. All they do is return the result of a DateAdd() function: one adds 1 to Now(), the other subtracts 1 from Now(). The value returned must be preceded by the keyword *return*, and whatever follows return is returned by the function to the caller. You can return functions (as we did here), variables, or any other valid CFML expression.

Once defined, these user-defined functions can be used just like any other CFML functions. As long as the UDFs appear on the same CFM page, or are included in your code via <CFINCLUDE>, they can be used like this:

```
<CFOUTPUT>
Today is #DateFormat(Now())#<BR>
Yesterday was
#DateFormat(Yesterday())#<BR>
Tomorrow will be
#DateFormat(Tomorrow())#<BR>
</CFOUTPUT>
```

As you can see, the new Yesterday() and Tomorrow() functions are used just like the built-in Now() function, and all three functions are passed to DateFormat(). Once defined, your own UDFs can be used like any other ColdFusion functions – there's no difference whatsoever.

UDFs Revisited

The UDFs and syntax reviewed thus far are CF5 based (although they'll work in CFMX as well, obviously). But CFMX now provides a whole

new (additional) way to create UDFs addressing some important limitations of <CFSCRIPT>-based UDFs.

- For many ColdFusion developers <CFSCRIPT> was foreign. CFML is far more familiar and comfortable.
- <CFSCRIPT> supports a subset of CFML – functions but not tags. Thus <CFSCRIPT>-based UDFs can't access tags (for example, you can't execute a <CFQUERY> in a <CFSCRIPT>-based UDF).
- <CFSCRIPT> UDF syntax provides limited parameter validation (no support for data types), and doesn't provide a mechanism to simply work with optional parameters.

So CFMX introduced tag-based UDFs – the ability to create user-defined functions using CFML tags. The tags used are:

- **<CFFUNCTION>:** To define the function
- **<CFARGUMENT>:** To define any arguments (parameters)
- **<CFRETURN>:** To define return codes

Do these tags look at all familiar? They should – they're the same ones used to create ColdFusion Components (as explained in my two previous columns). Here are the same two functions, Yesterday() and Tomorrow(), but this time created using tags:

```
<!--- Get yesterday's date --->
<CFFUNCTION NAME="Yesterday"
    RETURNTYPE="date">
    <CFRETURN DateAdd("d", -1,
Now())>
</CFFUNCTION>
```

```
<!--- Get tomorrow's date --->
<CFFUNCTION NAME="Tomorrow"
    RETURNTYPE="date">
    <CFRETURN DateAdd("d", 1, Now())>
</CFFUNCTION>
```

They're pretty much self-explanatory. Each function is created using a <CFFUNCTION> tag and ends with a matching </CFFUNCTION> tag. The code between the tags makes up the function itself. These particular functions have no processing, just a return value specified using <CFRETURN> (as in the <CFSCRIPT> versions).

So how would you use these functions? Just as you would the <CFSCRIPT> versions:

```
<CFOUTPUT>
Today is
#DateFormat(Now())#<BR>
Yesterday was
#DateFormat(Yesterday())#<BR>
Tomorrow will be
#DateFormat(Tomorrow())#<BR>
</CFOUTPUT>
```

It's worth noting that there's one difference between the <CFSCRIPT>- and tag-based versions of these functions: the latter define a RETURN- TYPE. This optional attribute is used to define the type of data returned by a UDF. This value is used in validating return values as well as where functions can and can't be used. If RETURN- TYPE isn't specified, any type will be allowed, just as in <CFSCRIPT>-based functions. As a rule, you should always define return types.

Accepting Parameters

The functions created above are atypical in that they accept no param-

CFDYNAMICS
www.cfdynamics.com

A Cold Cup O'Joe Part 8 of 8



BY
GUY
RISH

Applets and ColdFusion

Java applets are a much-maligned technology in the Internet world. Originally a major focus of the Java platform, they've retreated from their former place in the spotlight.

Problems with browsers, corporate conflicts, and being outflanked by faster and (arguably) lighter-weight tools like Macromedia's Flash have diminished the popularity of applets considerably. Despite this, they continue to be used for a variety of purposes. What's more, they're often the only solution since most other technologies have focused on limited platforms while Java purports to be usable everywhere.

Overview

For several versions of ColdFusion, applets have enhanced the client side of a number of ColdFusion applications. While CFML has a tag, CFAPPLET, for integrating with applets, some applets have actually prompted the creation of specialized tags just for their functionality. Tags like CFGRID and CFTREE have been in the vocabulary of ColdFusion developers for some years now.

Like Java servlets, and unlike the other Java technologies discussed in this series, no special configuration needs to be made in the Java Settings page for these to work. Everything happens in the browser. The only time the ColdFusion Administrator is needed is to register an applet for use with the CFAPPLET tag.

Setup

In this article all examples assume that the accompanying files are installed in a directory called "banners" off the local Web server's root directory.

The Applet

This article shows a number of different ways to integrate Java applets in a ColdFusion application. To stay within the scope of the article, I'll use a simple but useful banner applet to display a single clickable image or rotate through a set of clickable images.

Parameter	Description
bgcolor	Background color of applet display space; parameter OPTIONAL with default of white.
target	HTML named target where link will be loaded. OPTIONAL – default of _self.
delay	Number of seconds to delay between displaying each banner. OPTIONAL – 5-second default.
count	Number of banner/link entries. OPTIONAL – default value of 1.
item0 - itemN	" " delimited value with two fields: URL to banner image and URL to which banner links; at least one item REQUIRED – numbering starts at 0.

TABLE 1 ClickBanner applet parameters

For convenience, the applet, ClickBanner, is contained in a single Java source (producing two class files because of an inner class). The class runs an additional thread if more than one banner item is specified. ClickBanner takes a number of parameters, as seen in Table 1.

Applets in HTML

There are two tags in HTML for handling Java applets: the deprecated APPLET tag and the newer OBJECT tag. Despite being deprecated, the APPLET tag is still broadly used. In fact, both Sun's latest applet examples and ColdFusion's own CFAPPLET tag (discussed later) use it.

APPLET Tag

The original embedded-content tag, APPLET is pretty simple to use, as can be seen in the snippet below (from Listing 1, AppletExample1.html):

```
<applet code="ClickBanner.class"
codebase="http://localhost/banners/" width="468"
height="60">
<param name="item0" value="http://localhost/banners/megatokyo.gif|http://www.megatokyo.com">
</applet>
```

There are actually a broader number of attributes for this tag, as shown in Table 2. (This list doesn't include the attributes added to all HTML elements in 4.x for consistency, such as style, alt.)

OBJECT Tag

One problem with using the OBJECT tag is that it necessitates use of the Java plug-in. Although a discussion on the use of the OBJECT tag with Java applets is beyond

Attribute	Description
archive	Comma-separated list of URIs containing other classes or resources to be preloaded; relative URIs are interpreted by applet's codebase attribute.
code	Name of applet class file.
codebase	Specifies base URI for applet, which if elided is current document's URI.
height	Initial height of applet's display.
name	Applet's instance name, used for scripting.
object	Specifies resource containing serialized applet.
width	Initial width of applet's display.

TABLE 2 APPLET attributes

the scope of this article, extensive documentation is available at <http://java.sun.com/products/plugin/>. There are even tools for automatically migrating your APPLET tag directives to OBJECT tag directives.

Listing 2 is an HTML document, AppletExample2.html, that contains an example, but it works only if the Java plug-in is installed.

Applets in ColdFusion

Using ColdFusion's applet interface, CFAPPLET, is an excellent way to work with applets that have a fixed set of parameters. There are a couple of prerequisites:

- Applets must be registered in the ColdFusion Administrator.
- CFAPPLET must be used subordinate to CFFORM.

CF Administrator's Java Applets Page

Using an applet with the CFAPPLET tag requires that it first be registered through the ColdFusion Administrator. The main Java applets page (see Figure 1) can be seen on the Administrator's default server menu display under the Extensions heading (Server > Extensions > Java Applets).

Depending on your configuration (and whether you're using ColdFusion 4.5x or 5.x), you may see other applets already registered. The most common one is the COPYTEXT applet entry (used with ColdFusion's own applet-backed tags). It's here that you register the ClickBanner applet:

1. Click the "Register New Applet" button.
2. Complete the Add/Registered Applet page (shown in Figure 2) and submit the form with the values in Table 3.

CFAPPLET

The CFML wrapper on the APPLET tag, CFAPPLET, requires preparation in the CF Administrator and must be placed subordinate to a CFFORM tag. Aside from those basic requirements, using the CFAPPLET tag has

Field	Value	
Applet Name	BANNER	
Code	ClickBanner.class	
Code Base	http://localhost/banners	
Archive		
Method		
Height	60	
Width	488	
VSpace		
HSpace		
Align	Left	
Not Supported Message	Unable to display – applet support required.	
Applet Parameters	Name	Value
	bgcolor	FFFFFF
	target	–self
	delay	10
	count	1
	item	http://localhost/banners/empty.bmp http://localhost/banners/ClickBanner.html

TABLE 3 ClickBanner applet configuration settings

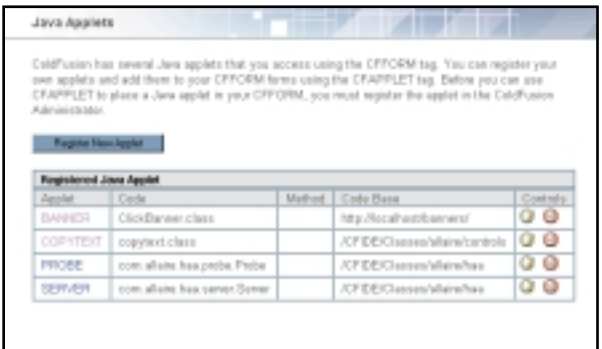


FIGURE 1 CF Administrator's Java applets page

only two required attributes (the complete list appears in Table 4): appletSource, the name of the registered applet; and name, the identifier given to the applet instance. This is mostly because so many defaults can be gotten from the registered data supplied to the Administrator.

Since the majority of the settings are looked up in the CF server registry, calling a registered applet with CFAPPLET is short work, as exemplified in this snippet from AppletExample3.cfm (see Listing 3):

```
<cfform
action="http://localhost/banners/AppletExample3.cfm" method="POST">
<cfapplet name="banner" appletsource="banner"
item0="http://localhost/banners/megatokyo.gif|http://www.megatokyo.com">
</cfform>
```

These few lines give us everything we've come to expect from the previous examples.

Applet-Backed Custom Tag

The two approaches looked at so far place the developer fairly close to the applet itself. HTML's APPLET tag

Attribute	Description
align	Controls placement of applet: <ul style="list-style-type: none">• Left• Right• Bottom• Top• TextTop• Middle• AbsMiddle• Baseline• AbsBottom OPTIONAL
appletSource	Name of registered applet
height	Optional height of applet
hspace	Amount of white space to add to left and right of applet. OPTIONAL
name	Form variable name of applet instance
notSupported	Display text if applets aren't supported in browser. OPTIONAL
param_1 to param_n	Param is only a placeholder name. OPTIONAL
vspace	Amount of white space to add to top and bottom of applet. OPTIONAL
width	Width of applet. OPTIONAL

TABLE 4 CFAPPLET attributes

ABOUT THE AUTHOR

Guy Rish is an independent consultant specializing in Web technologies and object-oriented design. He holds instructor certifications from Rational Software and Macromedia, and has taught in both corporate and academic settings. Guy recently contributed to books from Sybex and New Riders on ColdFusion MX, Flash MX, and Dreamweaver MX.

COLD FUSION Developer's Journal

Think in terms of logical entities
WHAT'S THE ADVANTAGE OF CFCs?
Find out in *CFDJ* September

Using data within the component
HOW IS IT DONE?
Find out in *CFDJ* September

Tips for a Trouble-Free Migration
HOW CAN YOU AVOID COMMON PITFALLS?
Find out in *CFDJ* September

Optimizing CFIF Tasks
HOW DO THEY MEASURE UP?
Find out in *CFDJ* September

Using a database and CF's graphing capability
WHAT CAN THIS DO FOR YOU?
Find out in *CFDJ* September

AddRegistered Java Applet

Click arrow on left to return without submitting changes.

Applet Name:

Code:

Code Base:

Archive:

Method:

Height: Width:

VSpace: HSpace:

Align:

Not Support of Message:

Click arrow on left to return without submitting changes.

Applet Parameters

Parameter Name	Value
bgcolor	0xffff
target	_self
relay	13
count	1
send	http://localhost:8080/status/ten

FIGURE 2 CF Administrator's Java applet registration page

and CFML's CFAPPLET tag require you to know a thing or two about the applet being used. This is where wrapping the applet in a custom tag can be very powerful. (This was actually done early on for CFML additions with CFGRID, CFTREE, and the like. These UI-oriented tags are really just thin wrapping over some pretty robust Java applets.) Doing so provides a number of important benefits: defaulting content, derived and housekeeping attributes, and applet lifecycle shielding.

While CFAPPLET allows for this through the applet registry, it doesn't allow you to script the logic. Using a custom tag allows a reactive default to be set where conditions can be checked.

Certain things that can foul up a page through a minor typo can often be avoided this way. Specifically, in the case of the ClickBanner applet,

Attribute	Description
destination	URL to link to when banner image is clicked. REQUIRED.
image	URL for image file. REQUIRED.

TABLE 6 CF_BANNERITEM attributes

Attribute	Description
bgcolor	Default background color on which banner images will be displayed. OPTIONAL – default white.
delay	Number of seconds to wait between displaying each banner image. OPTIONAL – 10-second default.
height	Height of banner's display area.
name	Name of applet instance, useful for scripting. OPTIONAL.
width	Width of banner's display area.

Attribute	Value
CF_CLICKBANNER	0
CF_CLICKBANNER	1
CF_CLICKBANNER	2
CF_CLICKBANNER	3
CF_CLICKBANNER	4
CF_CLICKBANNER	5
CF_CLICKBANNER	6
CF_CLICKBANNER	7
CF_CLICKBANNER	8
CF_CLICKBANNER	9
CF_CLICKBANNER	10
CF_CLICKBANNER	11
CF_CLICKBANNER	12
CF_CLICKBANNER	13
CF_CLICKBANNER	14
CF_CLICKBANNER	15
CF_CLICKBANNER	16
CF_CLICKBANNER	17
CF_CLICKBANNER	18
CF_CLICKBANNER	19
CF_CLICKBANNER	20
CF_CLICKBANNER	21
CF_CLICKBANNER	22
CF_CLICKBANNER	23
CF_CLICKBANNER	24
CF_CLICKBANNER	25
CF_CLICKBANNER	26
CF_CLICKBANNER	27
CF_CLICKBANNER	28
CF_CLICKBANNER	29
CF_CLICKBANNER	30
CF_CLICKBANNER	31
CF_CLICKBANNER	32
CF_CLICKBANNER	33
CF_CLICKBANNER	34
CF_CLICKBANNER	35
CF_CLICKBANNER	36
CF_CLICKBANNER	37
CF_CLICKBANNER	38
CF_CLICKBANNER	39
CF_CLICKBANNER	40
CF_CLICKBANNER	41
CF_CLICKBANNER	42
CF_CLICKBANNER	43
CF_CLICKBANNER	44
CF_CLICKBANNER	45
CF_CLICKBANNER	46
CF_CLICKBANNER	47
CF_CLICKBANNER	48
CF_CLICKBANNER	49
CF_CLICKBANNER	50
CF_CLICKBANNER	51
CF_CLICKBANNER	52
CF_CLICKBANNER	53
CF_CLICKBANNER	54
CF_CLICKBANNER	55
CF_CLICKBANNER	56
CF_CLICKBANNER	57
CF_CLICKBANNER	58
CF_CLICKBANNER	59
CF_CLICKBANNER	60
CF_CLICKBANNER	61
CF_CLICKBANNER	62
CF_CLICKBANNER	63
CF_CLICKBANNER	64
CF_CLICKBANNER	65
CF_CLICKBANNER	66
CF_CLICKBANNER	67
CF_CLICKBANNER	68
CF_CLICKBANNER	69
CF_CLICKBANNER	70
CF_CLICKBANNER	71
CF_CLICKBANNER	72
CF_CLICKBANNER	73
CF_CLICKBANNER	74
CF_CLICKBANNER	75
CF_CLICKBANNER	76
CF_CLICKBANNER	77
CF_CLICKBANNER	78
CF_CLICKBANNER	79
CF_CLICKBANNER	80
CF_CLICKBANNER	81
CF_CLICKBANNER	82
CF_CLICKBANNER	83
CF_CLICKBANNER	84
CF_CLICKBANNER	85
CF_CLICKBANNER	86
CF_CLICKBANNER	87
CF_CLICKBANNER	88
CF_CLICKBANNER	89
CF_CLICKBANNER	90
CF_CLICKBANNER	91
CF_CLICKBANNER	92
CF_CLICKBANNER	93
CF_CLICKBANNER	94
CF_CLICKBANNER	95
CF_CLICKBANNER	96
CF_CLICKBANNER	97
CF_CLICKBANNER	98
CF_CLICKBANNER	99

where a count parameter is set to tell the applet how many image/link banner combinations to pick up, this is a very useful feature. The ClickBanner custom tag counts these and generates the count parameter itself.

Once an interface has been established via a custom tag, minor changes to the applet can be hidden. Applet parameter name changes, changes to default values, even new required parameters can often be masked by a custom tag.

Following in the footsteps of ColdFusion's own designers, I've created a custom tag to wrap the ClickBanner applet. This custom tag is really a set with one subordinate to the other. CF_CLICKBANNER is the parent tag and sets up the applet; CF_BANNERITEM is the subordinate tag identifying the individual banner images and links. It won't take long to see that these two tags simplify the coding. CF_CLICKBANNER has only two required attributes (see Table 5 for the complete list), as does CF_BANNERITEM (see Table 6).


A simple example of this tag set can be seen in the following snippet (from Listing 4, `AppletExample4.cfm`):

```
<cf_clickbanner width="468"
height="60">
<cf_banneritem
image="http://localhost/ban-
ners/megatokyo.gif"
destination="http://www.mega-
tokyo.com">
</cf_clickbanner>
```

Despite the maligning and declining popularity, Java applets still have numerous uses. They can be very useful when concerns about the target platform would make alternate solutions difficult to implement. While

some tools are becoming more technologically competitive, their penetration is not nearly as great (Java applets run everywhere Flash does, but Flash doesn't run everywhere an applet does).

Given the variety of different implementations available in CFML

for incorporating an applet, they can in many cases be considered a slam-dunk option. Like any technology, when used judiciously it can be a great benefit. 

@GUYRISH30@YAHOO.COM

```
<html>
<head>
  <title>Applet Example 1</title>
</head>

<body>
  <applet code="ClickBanner.class" width="468" height="60">
    <param name="item0"
value="http://localhost/banners/megatokyo.gif|http://www.me
gatokyo.com">
  </applet>

</body>
</html>
```

```
<html>
<head>
  <title>Applet Example 2</title>
</head>

<body>
  <!-- "CONVERTED_APPLET"-->
<!-- HTML CONVERTER -->
<OBJECT
  classid="clsid:CAFEEFAC-0014-0000-0000-ABCDEFEDCBA"
  WIDTH = "468" HEIGHT = "60"
  codebase="http://java.sun.com/products/plugin/autodl/jinit-
  stall-1_4_0-win.cab#Version=1,4,0,0">
    <PARAM NAME = CODE VALUE = "ClickBanner.class" >

    <PARAM NAME="type" VALUE="application/x-java-
    applet;jpi-version=1.4">
    <PARAM NAME="scriptable" VALUE="false">
    <PARAM NAME = "count" VALUE = "2">
    <PARAM NAME = "item0" VALUE = "http://localhost/ban-
    ners/megatokyo.gif|http://www.megatokyo.com">
    <PARAM NAME = "item1" VALUE = "http://localhost/ban-
    ners/osdn.gif|http://www.osdn.com">
```

```
<COMMENT>
<EMBED
    type="application/x-java-applet;jpi-ver-
sion=1.4"
    CODE = "ClickBanner.class"
    WIDTH = "468"
    HEIGHT = "60"
    count = "2"
        item0 =
"http://localhost/banners/megatokyo.gif|http://www.mega-
tokyo.com"
        item1 =
"http://localhost/banners/osdn.gif|http://www.osdn.com"
    scriptable=false

pluginspage="http://java.sun.com/products/plugin/index.html
#download">
<NOEMBED>
```

```
</NOEMBED>
</EMBED>
</COMMENT>
</OBJECT>

<!--
<APPLET CODE = "ClickBanner.class" WIDTH = "468" HEIGHT =
"60">
<PARAM NAME = "count" VALUE = "2">
<PARAM NAME = "item0" VALUE
="http://localhost/banners/megatokyo.gif|http://www.mega-
tokyo.com">
<PARAM NAME = "item1" VALUE
="http://localhost/banners/osdn.gif|http://www.osdn.com">
```

</APPLET>
-->

```
<!--"END_CONVERTED_APPLET"-->
```

```
</body>
</html>
```

```
<html>
<head>
<title>Applet Example 3</title>
</head>

<body>
<cform
action="http://localhost/banners/AppletExample3.cfm"
method="POST">
  <cfapplet name="banner" appletsource="banner"
item0="http://localhost/banners/megatokyo.gif|http://www.megatokyo.com">
  </cfform>
</body>
</html>
```

```
<html>
<head>
<title>Applet Example 4</title>
</head>

<body>
<cf_clickbanner height="60" width="468">
  <cf_banneritem image="http://localhost/banners/mega-
tokyo.gif" destination="http://www.megatokyo.com"/>
</cf_clickbanner>
</body>
</html>
```

[illegible]

ColdFusion MX has opened up an entirely new world for us CF developers, and along with its new J2EE underpinnings comes an entirely new development construct for us to use in our application development: Enterprise JavaBeans.

EJBs aren't really a bunch of beans, but rather a specification. The EJB specification defines a methodology for deploying robust and distributed components. What this can do for you, in a nutshell, is increase the number of layers in your application. And this is a good thing.

Prerequisites

Before I get started, some words of caution: first, EJB isn't available within the ColdFusion MX standard application server. EJB runs in a separate "container."

Beef Up Your Apps with Enterprise JavaBeans



The world of *n*-tier application development

This container handles a bunch of services for the enterprise beans (these are what the actual components are called), exactly the way that ColdFusion handles a number of services for your Web applications, such as session management and transaction handling. This means that to use EJB you have to have CFMX installed on top of a J2EE server that includes such an EJB container – for example, Macromedia's JRun, Sun's iPlanet, and IBM's WebSphere (check www.macromedia.com for CFMX compatibilities).

Second, EJB isn't necessary for every application. In fact, most applications will be better served by other component

methodologies, like JavaBeans (yes, this is confusing, but JavaBeans isn't the same as EJB) or ColdFusion Components (CFCs).

The Old CF Way

Until now, we CF developers have developed our applications using what is called three-tier application development. (A little history: two-tier applications were the old client/server apps – one tier being the thick-client tier, the other the database tier. The Web brought a much better method for accessing applications, but because of HTML/browser limitations, having a thick client was pretty much out of the question. Thus three-tier development was born.)

Three-tier development has, obviously, three tiers: the user interface, the application logic (usually in an application server like ColdFusion), and the data layer (see Figure 1). Three tiers offer a lot of benefits over two tiers. For starters, the client tier displays only the data, saving processing needs on the client machine, as well as allowing for the use of cell phones or PDAs. Also, by breaking up the application and data tiers, both can be clustered or load balanced, allowing applications to scale as more users jump on.

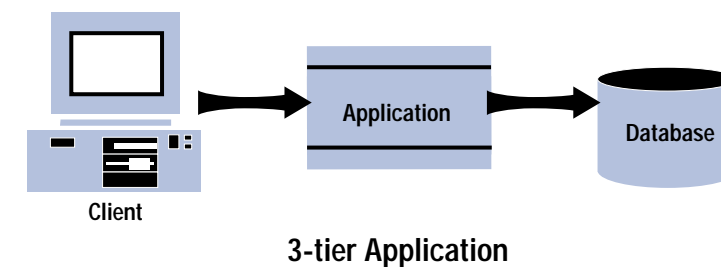


FIGURE 1 Three-tier development: An improvement over the old client/server apps

n-Tier Application Development

Enter *n*-tier development. Yes, this is the same *n* that brought us next *n* interfaces and to the “*n*th” degree. With *n*-tier development you increase the division of the application even more, from four layers on up. This includes adding services (like the many offered by J2EE or Java Message Service [JMS] and Java Naming and Directory Interface [JNDI]) and components (see Figure 2). The most important aspect of *n*-tier development is the use of these components to further separate pure business logic from program flow.

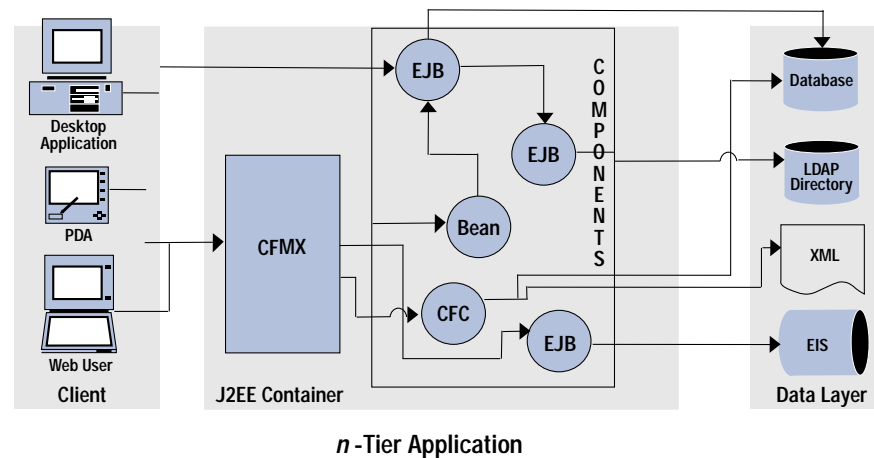


FIGURE 2 *n*-Tier development adds components and services, making apps more scalable and modular

In Figure 2 ColdFusion is controlling the application and serves as the interface between the client and the components and datasources. In *n*-tier development the components (in the form of JavaBean components, COM or CORBA objects, CFCs, or EJB components) handle all of the business logic, leaving .cfm pages (or servlets or JSP pages) to generate the UI, handle requests from clients, and send the appropriate responses.

The *n*-tier development approach offers several benefits:

- It's highly modular, allowing for easy updates and component swapping.
- It's scalable. Each layer can be improved or clustered individually.
- Clear separation of UI and program logic alone has many benefits:
 - Allows designers and programmers to focus on core competencies
 - Allows you to change underlying code without changing the UI
- Components can be accessed from external non-CF/Web applications.

This last point is a key component of EJB. Which leads us to the next section: What is EJB and when should it be used?

Defining Enterprise JavaBeans

Here are the facts: EJB is an industry standard. As such, a number of different J2EE application servers provide services for EJB. EJB components can't be developed in any programming language other than Java, though some have similar approaches to the same problem.

Enterprise beans can be accessed from any Java application – this includes ColdFusion, JSP, servlets, applets, and even rich Java desktop applications. They can be accessed locally or remotely – a huge point to remember. Most other component methodologies, such as (nonenterprise) JavaBeans, only allow you to access components locally from within the application server. Additionally, you can use ColdFusion (or other services) to deploy your enterprise beans as XML-based Web service components, allowing just about any application to use them, no matter what programming language is used.

You could create an enterprise bean, for example, to handle your shipping calculations. This complex business logic can then be accessed from applications throughout your enterprise: your e-commerce engine, ERP system, and Java-based sales reporting application. If you ever need to change your shipping mechanisms – to take advantage of UPS or FedEx exposing their calculation logic remotely to your company, for example – you only need to change the programming within the enterprise bean. None of the other applications require changes.

Advantages of Using EJB

As you can see, an enterprise bean can be a powerful component, handling many of the transactions and processes

that take place within your applications. EJB components offer many benefits – for example:

- They're deployable, which allows them to be accessed remotely.
- They're container-managed, saving development time when programming services and offering a number of additional benefits, such as session and persistence management.
- EJB allows you to cluster your components so you can have multiple servers handling only one component, further breaking up your applications and ensuring availability.
- EJB 2.0 (the latest EJB specification version) allows for asynchronous and synchronous messaging (see “Message-Driven Beans” section).

Part of the initial vision behind the EJB specification was to create a “component marketplace” where developers could find plug-and-play functionality for their applications, like a big-boned tag gallery. Truth be told, this marketplace hasn't grown as fast as predicted, but some good components are out there. Take a look at www.componentsource.com/java and <http://industry.java.sun.com/solutions>.

Anatomy of an Enterprise Bean

When you first receive an enterprise bean, either from the Java programmers in your organization or downloaded from a component broker, it's a simple JAR file, as in Java ARchive. You can examine the contents if you like, using WinZip. What you'll find are the following items:

- **Home interfaces:** Clients work with the home interface to generate bean objects.

CTIA

www.ctiashow.com

- **Remote interfaces:** It's the remote interface that allows any Java application to access the bean, even from across a network.
- **Local interfaces:** This allows clients to access the bean locally instead of remotely (the default method for accessing beans), thus saving overhead.
- **Enterprise bean classes:** These are the actual "beans" that are created to perform the necessary functions.
- **Deployment descriptor:** This is an XML file that tells the EJB container important information about how to handle this specific enterprise bean.
- **Possibly other JAR files or proprietary Java .class files:** These will contain programming utilities that are used in the enterprise bean.

Often Java IDEs or application server providers will provide wizards to ease the creation of these items. For example, Macromedia provides a wizard that creates the interfaces and the deployment descriptor and can even deploy the bean directly to JRun.

JMS is the universal application programming interface for utilizing message services in J2EE"

Types of Enterprise Beans

The different types of beans and what they're used for is a deep and ongoing study, certainly deeper than I can possibly go into here. That said, a brief introduction to session, entity, and message-driven beans is in order.

Session Beans

Simply put, session beans provide business logic. The example I stated previously for calculating shipping would be done using a session bean. Session beans are called such because they generally exist for a single "session" as invoked by a client. These beans are stored in memory and the EJB container gets rid of them when the client times out (much like session variables in CF). Session beans have two subtypes: *stateful* and *stateless*. The difference is about what you'd expect: a stateful session bean maintains state with the client. You'd use such a bean if you wanted to store information like an e-commerce shopping cart that relates directly to a client and must be maintained until the client is finished.

Stateless session beans generally perform functions that don't require this. The shipping bean example would probably be stateless; it would be created for the time required to calculate the shipping price and no more. In other words, stateful beans have a memory, stateless beans do not.

Entity Beans

Entity beans, unlike session beans, are *persistent*. Whereas session beans exist for a session, entity beans are permanently stored. In general, each entity bean

instance represents a single row from a database. Entity beans are directly mapped to your database, so changes made to the bean will eventually be committed to the database.

Why, you might ask, wouldn't I just want to work directly with the database? Actually, entity beans handle data extremely well. Because your database information is in memory, you can access the information faster, plus entity beans will (1) manage transactions, (2) allow concurrent access to information, (3) allow for transaction callbacks, and (4) cache data between transactions. Keep in mind that this is enterprise-level middleware – there's nothing exciting here, but as your applications grow you'll need these features more and more. EJB allows you to add these middleware services without having to code them yourself.

Message-Driven Beans

The most complicated to understand of all beans is the message-driven bean (MDB), new to EJB 2.0. This is because it requires an understanding of messaging and JMS. Calling on components remotely has some inherent problems: the client must wait while the server is processing, the server must be available (of course), and the component has to deal with a bunch of requests coming in from all directions.

Messaging is a way to eliminate these burdens. This message-oriented middleware (MOM) sits between the remote application and the local application and acts as a middleman, controlling flow, queuing requests, and returning any results or confirmations as they're processed. JMS is the universal application programming interface for utilizing message services in J2EE. It's through this API that messaging is done in Java.

MDBs are simply enterprise beans that can respond to requests sent through JMS. These beans don't actually return values or exceptions, but simply act as a mediator between JMS and other components in your *n*-tier application.

Example: The following demonstrates the use of an enterprise bean in a CF page. The bean itself (graciously provided by Kristian Cibulskis, one of my coauthors on an upcoming book, *Reality ColdFusion: J2EE Integration*) is called shippingCalc. Included in this bean is a method, getShippingCost(), that will calculate the shipping cost based on the information passed to it: sender's zip code, recipient's zip code, weight, and shipping method.

First we must deploy the enterprise bean to the J2EE server. In this example I'm using the full version of JRun 4 Server underneath ColdFusion MX. Because JRun 4 has an autodeployment mechanism built in, I simply copied my class files (and directories) into the CFMX root directory in a folder called /shippingCalc. I then added the following entry to the application.xml file in the <ColdFusion MX root dir>/META-INF/ directory:

```
<module>
  <ejb>shippingCalc</ejb>
</module>
```


The bean is now available for use in my CF applications.

Take a look at the CFML code for testBean.cfm in Listing 1, and follow the steps I took to consume this bean in ColdFusion:

1. Create an instance of the Java object javax.naming.InitialContext. This provides the methods to access the catalog of beans available on your J2EE application server. (If you're really interested, this is part of the JNDI API.) This is done using the <CFOBJECT> tag.
2. Next, obtain a reference to the bean's home interface. As mentioned earlier, enterprise beans have a number of interfaces. The home interface is used to generate new bean objects. This is done by setting a new variable, home, to the value returned by calling the method initContext.lookup("ShippingCalc"). In Java (and most other object-oriented programming languages), objects generally have a number of methods inherent within them, and are called by listing the object name followed by a dot, then the method name, then parentheses (possibly containing a comma-separated list of arguments, like a CF function). In this example we're calling the lookup() method of the initContext object by passing the String "ShippingCalc", the name of the bean we're looking for.
3. Now that you've found the bean, create an actual instance of the bean itself by calling the create() method. Set the bean to the shippingBean variable,

which will be the bean reference to access any methods within it.

4. For the last step calculate the shipping by calling the getShippingCost() method of the bean (using the shippingBean object you just created) and the necessary arguments. Note the use of the JavaCast() CF function. Java, unlike ColdFusion, has strict datatypes. When passing variables from CF to Java, it's often necessary to "cast" them to the correct datatype to avoid any ambiguity that would otherwise occur in the transition. Here I cast to int and String values.

That's it! We now have the price to ship a 2.2 pound package from Portland, Oregon, to Minneapolis via Pony Express: \$39.92. In your application, of course, you'd most likely want to call this tag using dynamic information (for example, the customer zip code and the total weight of items in the user's shopping cart). But you get the idea. Welcome to the world of J2EE and *n*-tier application development. 

About the Author

Drew Falkman, author of the JRun Web Application Construction Kit (Macromedia Press), is currently lead author on another book focusing on ColdFusion and J2EE integration. Drew has been building Web applications since 1995 using ColdFusion (CF5 certified) and, more recently, JRun (and other J2EE application servers).

 DREW@DREWFALKMAN.COM

Listing 1: testBean.cfm

```
<!--
Document Name: testBean.cfm
Create Date: 05/09/2002
Author: Kristian Cibulskis
Changes: Drew Falkman - changed to .cfm
from .cfc component
-->

<!-- Create the InitialContext -->
<CFOBJECT
  action=create
  name=initContext
  type="JAVA"
  class="javax.naming.InitialContext">

<!-- Obtain a reference to the home interface -->
<cfset home =
  initContext.lookup("ShippingCalc">

<!-- Create an instance of the session bean -->
<cfset shippingBean = home.create()>

<cfoutput>
#shippingBean.getShippingCost(
  JavaCast("int",97211),
  JavaCast("int",55345),
  JavaCast("int",2.2),
  JavaCast("string","Pony Express"))#
</cfoutput>
```

CODE LISTING

The code listing for this article is also located at

www.sys-con.com/coldfusion/sourcec.cfm

PACIFIC
ONLINE
www.paconline.net

Transforming XML Part 3

BY
DAVID
GASSNER



Powerful XML processing capabilities in a language you can learn easily

There once was a mild-mannered XML packet with greater ambitions. One day I'll be presentable, it thought! I'll dress up as HTML – and as WML, and maybe, just maybe, I can turn into WDDX!

Transformation looked scary, though, with lots of downloads, and installation, and really complicated syntax. But then along came ColdFusion MX, and the XML packet found that transformation wasn't such a big deal after all.

This is the third and final article in a series about ColdFusion MX and its new XML development tools. In previous *CFDJ* articles I focused on parsing XML and creating XML files (Vol. 4, issues 6 and 7, respectively). In this last article I'll describe how CFMX supports Extensible Stylesheet Language Transformations as a native technology.

XSLT is one of three major XML technologies that have driven adoption of XML as a data exchange format. (The others are DOM [Document Object Model] and SAX [Simple API for XML]). XSLT has been implemented for many platforms. There are XSLT processors built as COM objects (Microsoft's XML Core Services), and as libraries for C++, Perl, and Java developers (Apache's Xalan). XSLT processing is also built into the most recent browsers (Internet Explorer 6.0 and Netscape 6.2).

ColdFusion developers were able to use XSLT prior to the release of CFMX, but we had to locate and install the processors and learn the transformation syntax. CFMX now supports native XSLT with an embedded copy of Apache's Xalan processor (the Java version), and the whole thing just got much easier.

The XMLTransform() Function

An XSLT file is an XML file written in the XSLT language. To transform, you pass two packets into an XSLT processor. The first is the XML data packet; the second, the XSLT transformation packet. The processor performs the transformation and returns the resulting transformed text (see Figure 1).

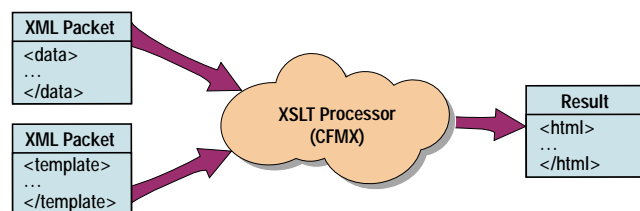


Figure 1 XSLT at work

CFMX executes transformations with the new XMLTransform() function. To perform a transformation, first read the contents of the XML and XSLT packets into ColdFusion text variables. Then pass the text variables into XMLTransform(), which returns the transformed text:

```
<CFSet myResult=XMLTransform(myXML, myXSLT)>
```

That's all there is to learn on the ColdFusion side. XMLTransform() passes the contents of the two packets into the XSLT processor and you get back the result. Everything you really need to learn is in the XSLT language itself.

How XSLT Works

XSLT provides a simple, extensible markup syntax for transforming XML into a variety of text formats. It can create formatted output as HTML or nonmarkup text, or transform from one flavor of XML to another. It's similar to CFML in that it's both a markup language and a command language.

XSLT is an XML-based language, and must follow the rules of well-formed XML. The root element is always the same:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
... transformation templates ...
</xsl:stylesheet>
```

The version number is always 1.0. The namespace URI indicates that the stylesheet follows the rules of the W3C's XSLT Recommendation, which was published in November 1999. The "xsl" namespace prefix is arbitrary, but all XSLT documents use it by convention. All XSLT command elements also have the "xsl" prefix. All other elements in the file are part of the resulting transformation.

Within the <xsl:stylesheet> element there is at least one <xsl:template> element that matches one or more parts of the XML data document – in the following example, the document root:

```
<xsl:template match="/">
... transformation rules ...
</xsl:template>
```

The match attribute uses XPath syntax, an XML expression language that follows many of the rules of directory and file addressing ("/" means the root of the document, "." means the value of the current element, and so on). XPath also supports full querying syntax that lets you find specific parts of an XML document. (I mentioned XPath in the article on reading XML (*CFDJ*, vol. 4, issue 6), as it's also used by ColdFusion's XMLSearch() function.)

What's XSLT Good For?

Minimally, XSLT can be used to perform XML formatting functions that aren't available in other parts of the CFMX XML API. For instance, in Part 2 of this series I mentioned that an XML document created using the Object method (using the XMLNew() and XMLElemNew() functions) comes out in a single string without any line feeds or indentation. It's well formed and parseable by an XML processor, but hard for the human eye to read.

To fix this, I created a user-defined function (see Listing 1) that merges the XML document with an XSLT stylesheet. This UDF can be used to format and indent any well-formed XML packet, including WDDX. The stylesheet contains a single <xsl:template> that matches the document root:

```
<xsl:template match="/">
<xsl:copy-of select="."/>
</xsl:template>
```

The <xsl:copy-of> element copies the entire content of the current object (the document root) to the transformation output. By itself, this <xsl:template> doesn't do much – it just returns the XML document in its original form. The real work is done by the <xsl:output> element that precedes the <xsl:template>:

```
<xsl:output indent="yes"
xalan:indent-amount="2"/>
```

The "indent" attribute indicates that the XML should be indented. By itself it doesn't actually indent, but it does create line feeds in the appropriate places. To indent I use the "indent-amount" attribute, which isn't part of the XSLT specification, but rather a unique feature of the

Xalan processor. Note the use of the "xalan" namespace prefix, which in turn is declared in the stylesheet's root element:

```
xmlns:xalan="http://xml.apache.org/xslt"
```

The <xsl:output> element also has an "encoding" attribute that can control the encoding format of the text.

Transforming XML to HTML

XML can be transformed into HTML with an XSLT stylesheet. For this example I'll use the Macromedia Designer/Developer Resource Feed, a list of available DesDev Center articles and columns in a predictable XML format.

To include the formatted contents of the XML file in a Web page, read the XML file with <CFHttp> and the XSLT file with <CFFile>:

```
<CFHttp url="http://
www.macromedia.com/desdev/
resources/macromedia_
resources.xml"/>
<CFFile action="read" variable="xslt"
file="#ExpandPath('.')#\MMResources.xml">
```

Now perform the transformation and output the results:

```
<CFSet result=XMLTransform
(cfhttp.FileContent, xslt)>
<cfoutput>#result#</cfoutput>
```

This simple code is all you need in your ColdFusion page (see Listing 2). The real work is done by the XSLT stylesheet (see Listing 3). There are two <xsl:template> elements: one for the root element ("macromedia_resources") and one for the data element ("resource"). The root <xsl:template> creates an HTML table skeleton and uses the <xsl:apply-templates> element to retrieve all resources with a "type" attribute whose value is the word Article:

```
<table border="1">
<tr><th>Article</th><th>Author</th></tr>
<xsl:apply-templates
select="resource[@type='Article']"/>
</table>
```

The second <xsl:template> matches each "resource" element. It creates an HTML table row and populates its cells with values from the element's children:

```
<xsl:template match="resource">
<tr>
<td><a
href="{url}"><xsl:value-of
select="title"/></a></td>
<td><xsl:value-of
select="author"/></td>
</tr>
</xsl:template>
```

The first <td> contains an <a> tag, with its "href" attribute populated from the current resource element's "url" child element. The curly braces indicate use of an XPath expression to retrieve a value from the XML data. The <xsl:value-of> elements also retrieve data with XPath expressions. The result is an HTML representation of the XML data. See the results in Figure 2.

Transforming XML to HTML

At some point you may want to transform an XML document into another flavor of XML. This is what XSLT was originally designed for, and it's really good at it!

The XSLT file in Listing 4 is designed to transform the DesDev Resource Feed into a WDDX packet representing an array of structures. The same two XSLT templates are used: one for the root and one for the data element. This time the root <xsl:template> creates the skeleton of a WDDX Array packet:



Figure 2 XML to HTML transformation

This transformation produces a WDDX packet that can be pub-

XSLT provides powerful XML processing capabilities in a lan-

@DGASSNER@NAPANET.NET

ABOUT THE AUTHOR
David Gassner is a Macromedia-certified instructor and the author of an upcoming course on XML development for ColdFusion developers. He teaches classes in ColdFusion, HTML, Java, and XML as well as other Web development skills.

CFXHOSTING
www.cfxhosting.com

ColdFusion offers developers several different scopes for variables, each with specific characteristics and uses. Application variables are global, shared by all users of an application. Server variables are similar, but if your server hosts several applications, these variables are shared among all the users of all the applications.

Request-scope variables are very handy – especially in the application.cfm template – to set variables that are available to all ColdFusion templates invoked with one HTTP request, including within custom tags that are otherwise shielded from the other variable scopes. Locally scoped variables are general purpose and the most widely used. However, client and session variables are especially interesting. They're most often used to provide users with their own extended attributes for an application. "Personalization" of a Web site greatly enhances the user experience, and client and session variables are ideally suited to this purpose. But both time out, and are tied to a particular client machine. This article demonstrates an easy way to make client variables available to a user on any PC used to access the application.

Session Variables

All Web application servers that I'm aware of (e.g., JSP, ASP, and ColdFusion app servers) support session variables. It's important to understand that a session refers to one user's activity at a site. As long as the server continues to receive HTTP requests from a given client within some configurable time frame, the session variables persist. If no activity is detected within that time frame, the session is deemed to have ended and all session variables expire. After a session variable is set on one page of a ColdFusion application, it becomes available to that user on every other page of that application as long as each HTTP request occurs within the configured timeout period.



Extending

the Usefulness of

ColdFusion Client and Session Variables

These variables may present problems, but they are fixable. Here's how . . .

A user called away from his or her desk could potentially lose information that might have been difficult or time-consuming to find and/or enter. This would not endear that Web site to me, and I'd need a very good reason to return. Personally, I keep a shopping cart full of purchases just standing at the checkout at Amazon.com, and anytime I feel flush I can make a purchase, wherever I am. If this data was simply stored in a session variable, that wouldn't be possible.

One strength of session variables is their support of complex data types. Session variables are stored within the default session structure and you can nest your own complex data types within this structure, permitting very powerful data constructs. ColdFusion 5 offers the CFDUMP tag, which greatly aids in debugging templates that use complex data types. If you're still working with an older release, you can download the CF_OBJECTDUMP custom tag from <http://devex.macromedia.com/developer/gallery>. Becoming comfortable with the notation required when manipulating complex data types can take some effort, and both tags offer clear, graphical views of your data, including arrays, structures, queries, and simple data types.

Since session variables are one of the three types of shared-scope variables stored in your server's memory (application and server variables are the other two), best practices insist that all references to them be enclosed inside a cflock tag to prevent corruption from concurrent updates. Such corruption can lead to performance problems, memory leaks, and server instability (see ColdFusion Locking Best Practices, TechNote 20370, at www.macromedia.com). Due to the overhead of locking variables as well as the hassle of coding the cflocks, I usually try to find another way to accomplish what I need to do.

If your application uses shared-scope variables, it's a good idea to enable "Full Checking" on a development server by going into the ColdFusion Administrator under Server Settings/Locking. In the interests of performance, I don't recommend using this option on your production server. When enabled, the CF compiler will check that all references to shared-scope variables are properly locked. For instance, it will catch this hard-to-find error:

To make session variables available to your CF application, simply include the session management attribute in your cfapplication tag:

```
<cfapplication name="myApp" sessionmanagement="Yes" sessiontimeout="#CreateTimeSpan(0,0,5,0)#">
```

Although session variables may be used in many ways, one of the most useful is in conjunction with a login process. For sites requiring some level of security, session variables are an ideal way to time out a user's session and force a relogin. I don't recommend using them for something like a shopping cart unless they're coupled with a more permanent data storage mechanism.

```
<cflock scope="session" type="readonly" timeout="5">
  <cfset Variables.session = session>
</cflock>
```

In this case the Variables.session["sessionid"] is nothing more than a second pointer to the same session variable and must be locked. The cfset statement didn't create a local copy of the session structure. You must use the Duplicate() function to create a

new structure, as opposed to a new reference to the same structure. Be careful with the StructCopy() function – it creates a copy of an existing structure by copying the simple data types in the source structure and creating references to the complex data types in the structure, something midway between a cfset and a Duplicate(). An example of the Duplicate() function would be:

```
<cflock scope="session"
type="readonly" timeout="5">
  <cfset Variables.session =
    Duplicate(session)>
</cflock>
```

Interestingly, the Full Checking setting won't catch this error:

```
<cfoutput>#Variables.session.sessionid#</cfoutput>
```

Since this is simply an alternate syntax for the previous statement, there appears to be a bug in the CF compiler. Also note that the Full Checking setting is incompatible with NAMED locks, as opposed to SCOPED locks. I found that the setting didn't work with the ColdFusion/Flash Component Kit I tested awhile back and has caused problems with early releases of Spectra

or any application using named locks. Any use of a named lock will throw an error with Full Checking enabled. Still, it's a useful debugging tool, if not 100% reliable or wholly compatible with existing tools and apps. Scoped locks, introduced in ColdFusion 4.5, are generally preferred over named locks, according to the Best Practices TechNote mentioned earlier, but named locks still have a limited purpose when you work with file operations and custom tags that aren't thread safe. To quote from the CF 5.0 help documents:

Providing the name attribute allows for synchronizing access to resources from different parts of an application. Lock names are global to a ColdFusion server. They're shared between applications and user sessions, but not across clustered servers.

Be sure to check the footnotes of the technote for additional clarification on locking. Going back to my suggestion to use a session variable in conjunction with a login process, I like to set a session variable with a name like isLoggedIn upon successful login. Then I can easily test whether a user is logged in or not by simply coding:

```
<cfif
isDefined("session.isLoggedIn")>...
```

Although the Full Checking option inside CF Administrator (discussed below) doesn't object if you fail to place a cflock around this statement, I believe this is another bug. If session.isLoggedIn isn't defined, I know that either the user hasn't logged in yet or the session has timed out. I don't need to test the value of the variable, only whether it exists. You can see a sample of this in the Application.cfm template in Listing 1.

Client Variables

Client variables are unique to ColdFusion and offer several terrific advantages over session variables. Since they're not shared-scope, they don't require locking. Your ColdFusion server, of course, is configured to store client variables in a real database like SQL Server, Oracle, or Sybase; you wouldn't consider deploying a production application in which you store your client variables in the server's registry or MS Access. This permits the database to handle all locking issues

for you. A database is also quick and works well in a clustered environment without additional programming. Further, client variables persist from session to session. A configurable timeout is associated with client variables, but it's generally measured in days or weeks as opposed to the usual timeout period of 15–20 minutes for session variables.

The beauty of client variables is that they're managed by ColdFusion and require only that the client management attribute of the cfapplication tag be set to yes. Behind the scenes, ColdFusion assigns a unique CFID/CFTOKEN to your client machine/browser the first time you access a CF application, and sets these values into a cookie stored on your hard drive with an expiration date far in the future. Using this cookie, ColdFusion then stores your client variables in the specified datastore and handles all updates and retrievals automatically. Interestingly, if stored in a database, ColdFusion manages this by serializing the client variable key/value pairs for storage and deserializing them as needed. As long as your users don't delete their cookies, ColdFusion will know who you are the next time you visit and will retrieve your client variables automatically. You can see samples of client variables in the registry under HKEY_LOCAL_MACHINE\SOFTWARE\Allair e\ColdFusion\CurrentVersion\Clients (see Figure 1) and in a database (see Figure 2).

Note that when stored in the registry, client variables are essentially stored as name/value pairs. When they're stored in a database, they're stored in two tables named CDATA and CGLOBAL. The former is the one that stores client variables set by the developer; they're stored in a string format in the data column. This string may be viewed as a ColdFusion list, delimited with # signs. Note also that embedded # signs are escaped. The cfid column actually contains both the cfid and cftoken, and the app column contains the application name. These two columns uniquely identify a row in the CDATA table. You can improve the performance of your application by manually creating a composite primary key on these columns; similarly, you can create a primary key on the cfid column of the CGLOBAL table. Also, if your application doesn't use the special system-managed client.hitcount, client.lastvis-

it, and client.timecreated variables, you can disable global client variable updates in the client variables section of the ColdFusion Administrator for another small performance boost. This setting may be customized for each individual client variable store.

One limitation of client variables is that they support only simple data types. You can't store a structure, query, or array in a client variable, although you can store a character, Boolean, string, number, or list. In practice, I haven't found this to be a drawback. In fact, if you first serialize your complex data type, you can store it in a client variable. I'm not sure why you'd want to, but...

Another limitation of client variables, easily surmounted, is their reliance on the CFID/CFTOKEN stored in the cookie. Users who delete their cookies are essentially deleting their client variables. ColdFusion server will assign a new CFID/CFTOKEN to these users and their client variables will be orphaned and not retrievable. Another aspect of this same issue is accessing the Web application from a second client. Since the client variables are tied to the cookie, which resides on the first

client machine, these variables are again not available. They will, however, be purged automatically by ColdFusion after the client timeout period – set in CF Administrator – has passed without a visit from your user.

Serialization and Portable Client Variables

Okay, what is this serialization business? Simply put, it's a method of taking a collection of variables or a single, complex variable and turning it into a string. ColdFusion includes support for WDDX, a type of XML, and provides a set of functions (both ColdFusion and JavaScript) and tags for serialization and deserialization. I'll use the simple CF tags to serialize and deserialize our collection of client variables so I can store them in such a way that I can access them as needed.

While developing these techniques, I worked up the beginnings of a simple e-commerce application that permits visitors to browse and customize their experience (see Figure 3) by setting their preferences. If they want to subscribe to an e-mail newsletter, make a purchase, or save their settings, they must

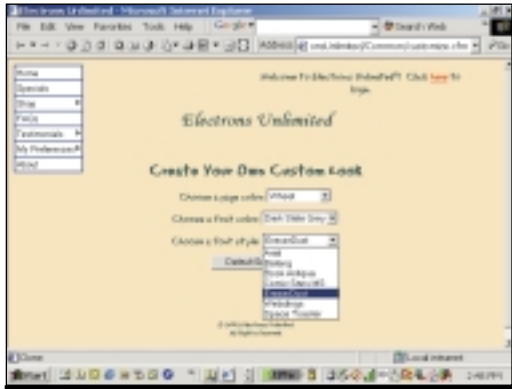


Figure 3 Simple app permitting visitors to browse/customize their experience

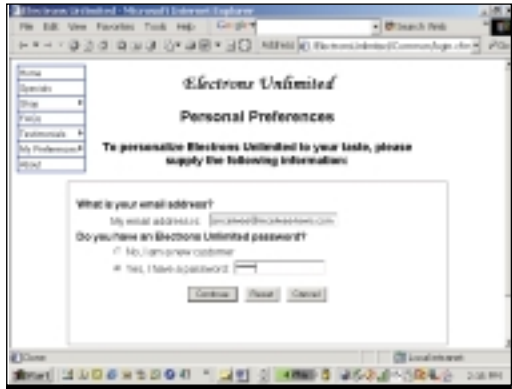


Figure 4 Visitors must create a valid account.

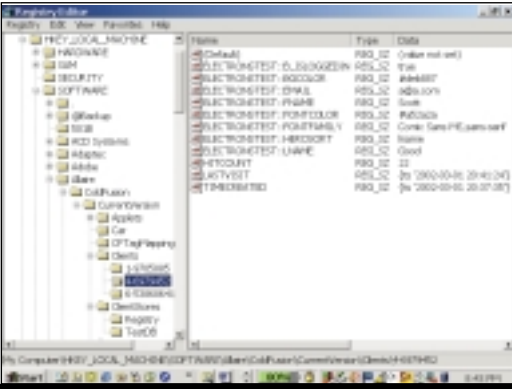


Figure 1 Client variables in the registry

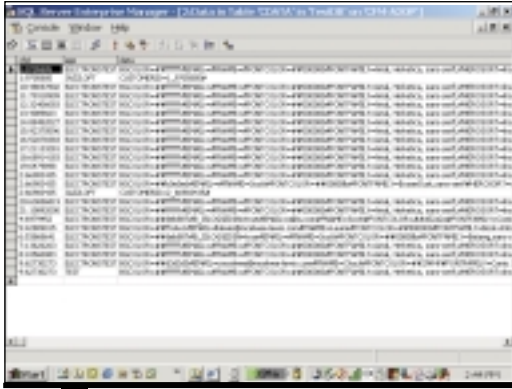


Figure 2 Client variables in a database

E-ZONE MEDIA
www.fusetalk.com

Understanding the Model-View-Controller Design Pattern

A sophisticated architecture that can help you enormously

BY
HAL
HELMS



With the advent of ColdFusion MX, CF programmers have the ability to bundle data and functions into a single unit for the first time.

Macromedia calls these constructs ColdFusion Components, or CFCs; other languages call them objects. This isn't to say that ColdFusion MX is now an object-oriented language. OO purists will rightly point out that CFCs don't have all the required features of first-class objects (including, perhaps most notably, polymorphism). Ben Forta's article on CFCs at www.macromedia.com makes this point under the heading "CFCs Are Objects, Kind Of."

Granting this point, though, doesn't diminish the great capability CFCs provide ColdFusion programmers and the new opportunities they open to us. CFCs can give our code greater power, flexibility, and reusability – all the benefits that object technologists have recognized for many years. If they aren't full card-carrying members of the Object Club, they're still very useful, and far less complex to use than their big brothers. Borrowing from a famous ad campaign, we might say that "CFCs are objects for the rest of us."

To gain advantage from this new capability requires new thinking. A shift in perception is required, in which the programmer moves from thinking of the application as composed of low-level details such as datatypes, recordsets, and functions to a more abstract view in which objects, in an almost lifelike way, handle their own data and know how to perform appropriate actions. Instead of viewing an application as data that's manipulated, the programmer begins to think of an application as composed of vari-

ous objects that respond to different requests.

Object theorists refer to this as encapsulation. Any required action for an application is assigned to an appropriate object.

Here the two methods are contrasted:

Method A: Traditional data manipulation

```
<cfquery
datasource="#request.dsn#"
name="UserInfo">
SELECT
    firstName,
    lastName,
    userID,
    userPermissions
FROM
    USER
WHERE
    userID = '#URL.drilldown#'
</cfquery>
<cfoutput>
Hello there,
#UserInfo.firstName#
#UserInfo.lastName#
</cfoutput>
```

Method B: Using encapsulated CFCs

```
<cfset currentUser =
CreateObject( "component",
"User" ) />
<cfset currentUser.getInfo(
url.drilldown ) >
<cfoutput>
Hello,
#currentUser.getFirstName()#
#currentUser.getLastName()#
</cfoutput>
```

In last month's column I said that CFCs combine structures with user-defined functions. Note, though, that here I'm not simply outputting two keys in the currentUser structure, currentUser.firstName and currentUser.lastName. That code would look like this – and would work:

```
#currentUser.firstName#
#currentUser.lastName#
```

But encapsulation asks us not to think of an object – or a CFC – as its constituent data elements, but as a self-contained entity. To get information from an object, I make a request of the object itself and therefore call the methods getFirstName() and getLastName(). It's up to the object to determine how this information is stored and in what way it will respond to these requests.

Although such a strict encapsulation may seem foreign to programmers at first, the benefits of such a plan soon win over most developers. When that change of perception does occur, programmers often succumb to "object fever" and the entire activity of programming seems radically new and exciting. In that fervor there's a tendency to ask objects to do too much, or rather, to ask a single object to do too much.

For example, a developer may realize that common user management functions (create, read, update, delete) can be wrapped into a single user object. Concentrating on what this object will be responsible for, our newly minted object programmer decides to include methods for dis-

playing a form to gather new user information: User.newUserForm(). Submitting this form will call User.new(). Editing a user is as simple as calling User.editUserForm(), followed by User.update().

All goes well with our new CFC developer – let's call him "Bob" – until more than one view of a new user form is required. Perhaps one user form will be displayed to a user registering him- or herself, while a different one, with added fields, is made available to a supervisor.

Bob thinks about making changes to the form, altering the display based on who called it, but he's well aware that excessive conditional code makes for fragile code, and elects instead to have separate methods: User.newUserAsUser() and User.newUserAsSupervisor().

It doesn't stop there, of course, and soon a request comes in to allow a new user to be created by sending a formatted e-mail, which the program should read. Bob adds User.newUserFromEmail() and is wondering, "Can something this ugly really be right?" The final blow comes when another developer working on a different application wants to use Bob's user CFC. Naturally all the forms for creating a new user are quite different, and Bob finds himself looking at a method like this:

```
User.newUserAsSupervisorForHRApplication()
```

One very good solution to Bob's problem is the use of a design pattern known as Model-View-Controller, or MVC. Design patterns are time-tested architectural solutions to common problems, and Bob's problem is a common one. MVC was developed some 20 years ago at Xerox's famed Palo Alto Research Center (PARC).

In the MVC design pattern the modeling of the external world and the visual feedback to the user are explicitly separated and handled by three types of objects, each specialized for its task.

Objects/CFCs belonging to the model are responsible for handling the business logic and "back-end" work. Typically, complex algorithms and databases belong to the model. View objects are responsible for displaying information to and retrieving information from the user. In the Web world this is normally done using HTML and graphics, with Flash and Java used when greater capabilities are desired. Controller objects interpret mouse and keyboard inputs from the user, making requests of the model and/or the view as appropriate.

Applying this to Bob's situation, we see that there are several different views – different users and different applications – while the model remains stable. After reading an article entitled (ahem) "Understanding the Model-View-Controller Design Pattern" in his favorite magazine, Bob decides to create a user CFC that will handle only the essentials of user management. The user CFC must be responsible for creating a new user, for example, but the interface into that CFC is something wholly separate, something belonging to the view. It will then be the job of a controller to manage interaction between model and view.

Bob decides that he will create a user CFC that acts as a model. To the user CFC, creating a new user means saving user information to a database, assigning a CSR to the user, and sending an e-mail to that CSR.

Everything else – what the entry form looks like, the permissions needed to see that form, etc. – is not the user's concern. With this new perspective, Bob creates a user model CFC that looks like this:

CFC: USER

Methods

```
new( firstName, lastName, address, city, state,
zip, email, [csrID], [userPermissions] )
edit( userID, firstName, lastName, address,
city, state, zip, email, [csrID],
[userPermissions] )
getInfo( userID )
delete( userID )
```

Bob realizes this is just a start toward a full MVC implementation, but it's a good start. At some point in the future, if the company decides to change the method of storing user information – from SQL Server to Oracle, say – Bob will only need to change the implementation of his user CFC. All the applications that use that model will work without change.

MVC is a sophisticated architecture that not all applications will need. Still, an understanding of it can help you enormously in tackling large, complex – and profitable – jobs. Fusebox 3 makes a very good platform on which to implement MVC. You can find out more on this at www.techspedition.com.



HAL@FUSEBOX.ORG

ABOUT THE AUTHOR

Hal Helms
(www.halhelms.com)
is a Team Macromedia member who provides both onsite and remote training in ColdFusion and Fusebox.

Pick 3 or More and
SAVE BIG!

Get a **SPECIAL LOW PRICE** when you
subscribe to 3 or more of our magazines



Wireless Business & Technology • Java Developer's Journal
WebSphere Developer's Journal • XML-Journal
Web Services Journal • ColdFusion Developer's Journal
WebLogic Developer's Journal • PowerBuilder Developer's Journal

**SYS-CON
MEDIA**

www.sys-con.com/suboffer.cfm

RECEIVE YOUR
DIGITAL EDITION
ACCESS CODE
INSTANTLY
WITH YOUR PAID
SUBSCRIPTION

New Possibilities for Session/Client Variable Handling in CFMX

BY
CHARLES
AREHART



It's more than just J2EE sessions

Tucked away among the press and praise about ColdFusion MX are some new options that could benefit every CF developer.

If you've seen them mentioned, they may not have seemed important at first glance, but I hope you'll give them consideration.

In this article I'll uncover some possibilities for session and client variable handling in CFMX – actually three new features you can choose to enable. I'll not only explain what they are, but why they're important and some other things to be aware of.

Are you thinking, "I already know about J2EE session variables"? That's not what this is all about. It is indeed one of the three new features, but the other two aren't dependent on that one and are of value to any CF developer. One of them doesn't even require any change to your code to obtain substantial benefits. The features are detailed under the following headings:

- The New `URLSessionFormat()` Function
- Using a UUID for CFTOKEN
- Using J2EE Session Variables

Before proceeding, I should point out that, despite the name of the first new feature, both it and the second feature are related not just

to session variable handling, but to supporting client variables as well. A few challenges that people have faced in working with both kinds of variables may indeed now be solved. You may even learn a new thing or two, as there are many misunderstandings about how these processes work. Let's start by setting the stage for the first feature.

Handling Session and Client Variables Without Cookies

The first new feature is a real hidden gem that hasn't been played up much at all. To understand its value, some background may be necessary. If you've needed to support session (or client) variables, you know that these features rely primarily on the browser supporting cookies to keep track of the user's identity (using what CF calls the CFID and CFTOKEN values it generates for each visitor).

Problems arise for browsers that don't support (or are by mandate not allowed to accept) cookies when they visit your site. In that case you must manually append the CFID and CFTOKEN to the URL used in any HTML you send to the browser that passes the user back to your server, such as `<A HREF>` and `<FORM ACTION>`. (The `<CFLOCATION>` tag also has an option called `ADDTOKEN="yes"` that accomplishes the same thing.)

If you haven't been paying attention to this issue, you may be suffering when users who don't support cookies visit your site. They never keep session variables from page to page, for instance. That can wreak havoc on a login process. And they also can't keep client variables from visit to visit.

The converse is that you shouldn't *always* append these values to a URL because that leaves your site open to several potential problems. If someone passed a bookmark of a URL with a given CFID/CFTOKEN pair shown, the user receiving that bookmarked URL would also now use the first user's CFID/CFTOKEN pair. Have you ever heard of two people seeming to share a session? This is one way it happens.

Another challenge is that someone can just guess at CFID/CFTOKEN values when displayed this way on a URL (more on another way to solve that problem in a moment).

So the optimal way to handle this (pre-CF MX) is to somehow test whether cookies are supported for the user running the template and then, only if they're not, append the CFID/CFTOKEN value. The code to do that isn't difficult, but getting it right and then placing that CFID logic around every instance of `<A HREF>`, `<FORM ACTION>`, or `<CFLOCATION>` (deciding whether to use `ADDTOKEN="yes"`) could be challenging.

New `URLSessionFormat()` Function

Enter the new `URLSessionFormat` function. With this simple function you can now let CFMX worry about whether to append the CFID/CFTOKEN pair (and/or the `JSESSIONID` if using J2EE sessions, as discussed in a moment). Here's a simple example:

```
<cfoutput>
<a href="#URLSessionFormat(
  ("MyPage.cfm?name=bob")#">
  some link</a>
</cfoutput>
```

CFMX Locking Best Practices

Though not technically a change in the way session variables are handled, it's worth pointing out that another new change in CFMX concerns locking access to shared scope variables, such as the session, application, and server scopes.

Rather than my taking time to explain the changes, I recommend that you look at the Macromedia TechNote "ColdFusion MX: Best Practices for Locking Shared Scope Variables" at www.macromedia.com/v1/handlers/index.cfm?ID=23021.

Web Services Java XML NET Wireless

web services
conference & expo

JDJ
conference & expo

XML
conference & expo

wireless
conference & expo

The Largest Web Services, Java, XML, .NET and Wireless Conference and Expo



OWNED BY
SYS-CON
MEDIA
PRODUCED BY
SYS-CON
EVENTS



CONFERENCE: OCTOBER 1-3, 2002
EXPO: OCTOBER 2-3, 2002

SAN JOSE MCENERY CONVENTION CENTER, SAN JOSE, CA

Focus on Web Services

Those companies that get an early jump on Web services to integrate applications for the enterprise will be the winners in today's challenging landscape. Take three days to jump start your Web services education!

Focus on Java

Hear from the leading minds in Java how this essential technology offers robust solutions to *i*-technology professionals and senior IT strategy decision-makers.

Focus on XML

Here's your chance to hear up-to-the-minute developments in standards, interoperability, content management, and today's new quest for more efficient and cost-effective Internet and intranet-enabled business process integration.

Reaching Beyond the Enterprise

The Conference...

Organized into five comprehensive tracks:

Web Services, Java, XML, Wireless, and IT Strategy

- Over 60 Information-packed Sessions, Case Studies and tutorials
- Thought leaders and innovators presenting the topics you most want to hear

Free...

FREE Web Services Tutorial to the First 100 to Register.

Attend a FREE Web Services Tutorial on October 3.

Register by August 15th to reserve your seat!

Who Should Attend...

- Developers, Programmers, Engineers
- *i*-Technology Professionals
- Senior Business Management
- C Level Executives
- Analysts, Consultants

The Largest Expo...

Over 75 exhibitors will deliver their best product demonstrations, stand ready to hear your challenges, and help you find solutions. Don't miss this important opportunity to expand your options.

PLATINUM SPONSOR



GOLD SPONSORS



SILVER SPONSOR



CORPORATE SPONSOR



MEDIA SPONSORS



If CFMX detects that the browser executing this template isn't passing the needed cookie (CFID/CFTOKEN for CF session and client variables, JSessionID for J2EE session variables), then it will turn that output HTML into the following:

```
<a href="MyPage.cfm?name= bob&
cfid=xxxx&cftoken=xxxxxxxx">some
link</a>
```

On the other hand, if CFMX detects that the needed cookies are being passed in by the browser, it leaves the identifiers off. Very cool! Note that it's smart enough to realize if the URL already has something in the query string (as it does above), in which case it uses an ampersand (&) to append the identifiers. Ta-da! (In that example I'm not yet showing what it looks like if J2EE sessions are being used.)

There is one gotcha: if you also need to use a URLEncodedFormat function for some part of the query string that has embedded spaces or special characters, it could become pretty cumbersome to use both functions in a single line of code, with embedded functions and strings within those functions. The following looks ugly, but it will work:

```
<cfoutput>
<a href="#URLSessionFormat
("MyPage.cfm?name=#URLEncoded
Format("billy bob")#)"#>some
link</a>
</cfoutput>
```

One other observation: I've noticed that when using J2EE sessions, it's possible for the resulting URL to appear in the format `MyPage.cfm;JSESSIONID=8030949691025145133137?name=bob`. Note that in this case the JSessionID is appended to the filename separated by a semicolon, rather than to the query string. Still, when that's happened, the URL has functioned as expected.

It may be worth pointing out that, technically, this function's name isn't completely accurate. It's needed not just for session variable handling but client variable handling as well. In other words, if you use client variables but not session variables, this is still the way to guarantee that the

CFID/CFTOKEN pair needed for client variable support as well is sent to browsers that don't allow cookies. (Indeed, if client variables are in use in that previous example showing the JSessionID, then the CFID and CFTOKEN would be appended to the query string as they were above.)

You also shouldn't dismiss browsers that don't support cookies as being antiques not worthy of your concern: many organizations force users to disable cookie support in their browsers. This solution helps you support them as well.

Using a UUID for CFTOKEN

I mentioned previously that one of the challenges of CFID and CFTOKEN pairs is that if the value is displayed on the URL, it's just very easy to try to guess. The CFID/CFTOKEN values are just a few simple numbers. By trying different numbers, a user may be able to impersonate or "spoof" another user's session (or client) variables. (And this really isn't a concern just if you pass them on the URL. Anyone familiar with the process can simply type a CFID/CFTOKEN pair on any URL running a CF template and, though the odds may be slim, possibly guess an active pair.)

So another new feature of CFMX, which has nothing to do with J2EE session variables, is that you can ask the server to generate more elaborate UUIDs (Universal Unique Identifiers) for the CFTOKEN. This is enabled in the Administrator, on the "server settings" page, by checking the option "Use UUID for cftoken". (The fact that this is not on the "memory variables" page reinforces the point that the CFTOKEN is used for both client and session variables.) You'll need to restart the CFMX server for this change to take effect. You needn't change any code to benefit from this new feature.

After enabling them, you may see that CFTOKEN values look more like this, as an example: `15ce46ab4e29af0a-AF695847-F92F-344A-133252991FB6C3B5`. (You can see it yourself with `<cfoutput>#cftoken#</cfoutput>`.) Definitely a lot harder to randomly guess an active value! It's a feature that probably should be enabled by all sites, just for the added protection. The only risk is if you

have any code that for some reason relies on the CFTOKEN being the simpler 8-digit number (or are storing the CFTOKEN in a database column that needs to be widened).

A side note: The ability to use a UUID for CFTOKEN isn't really new in CFMX. It's just easier to enable. In CF4.5 and 5 it requires a manual registry change. See the Macromedia TechNote at www.macromedia.com/v1/Handlers/index.cfm?ID=22427&Method=Full for more information.

Using J2EE Session Variables

I mentioned in the first section that CF now supports "J2EE sessions." What are they? And why would you care? Well, as a coder, it's possible that they'd only add benefit and, again, there's little reason not to use them. It's another feature set in the CFMX Administrator, in the "memory variables" page, checking the "Use J2EE session variables" option and restarting the server.

J2EE sessions work by sending to the client a cookie not with CFID and CFTOKEN but JSessionID. (Again, if CFAPPLICATION has CLIENTMANAGEMENT="yes", then the CFID/CFTOKEN pair is still sent, to support client variables only.)

There's more to the difference between the CFID/CFTOKEN pair and JSessionID than the name. First, the JSessionID value is a more elaborate combination of characters (including a UUID). As mentioned previously, the default CFID/CFTOKEN pair values are simply a few numbers each. That may make them possible to guess. Then again, you've just learned that you can change the CFTOKEN to use a UUID, so that benefit may seem diminished.

But there are still more differences, and they can be very important to some. First, and coolest of all, is that J2EE sessions work the way most developers have long wished CF session variables would: when the user closes his or her browser, the session is terminated as well. Hallelujah!

How does the mechanism work that allows the session to terminate when the browser is closed? Maybe you've already guessed: the JSessionID that's used for J2EE sessions is set as a nonpersistent (or "per-session" or

"temporary") cookie. That means simply that the cookie value isn't stored persistently in the browser user's hard drive. It's held only in the browser's memory. When the browser (all its instances) is closed, the JSessionID is lost. On a subsequent visit in a new browser window, the user is given a new JSessionID.

Technically, the session will live on in CFMX's memory until it times out. But with the user no longer holding the JSessionID for it, it's effectively "terminated" as far as he or she is concerned.

This also points out another benefit of using J2EE sessions for those organizations that aren't allowed to use persistent cookies (such as the CFID and CFTOKEN cookie values set by CFMX and previous versions). These organizations can use J2EE sessions much more easily than they could CF-based sessions. Of course, there are ways in all releases of CF that they could force the CFID and CFTOKEN to be nonpersistent, as outlined in the Macromedia TechNote at www.macromedia.com/v1/Handlers/index.cfm?ID=21079&Method=Full. With J2EE sessions, they needn't bother with that.

One final benefit of using J2EE sessions, which may or may not impress most CF developers, is that using them allows sharing of session variables with JSP and servlet programs also run under CFMX. That could be valuable, if you start exploring that capability. For more about sharing session and application variables between CF and JSP/servlet pages, see Chapter 32 of the CFMX manual, *Developing ColdFusion MX Applications with CFML*, available online at <http://live.docs.macromedia.com>.

Summary

Again, there's more to what's new in session variable support in CFMX than just J2EE sessions. Those first two items are valuable to all CFMX developers and apply to client variables as well (and the second one can apply even to users of CF4.5 and 5). The features add new dimensions in security, flexibility, and capability. Check them out!

@CAREHART@SYSTEMANAGE.COM

ABOUT THE AUTHOR

Charlie Arehart is a certified Macromedia trainer/developer and CTO of SysteManage. He contributes to several CF resources, is a frequent speaker at user groups throughout the country, and provides training, coaching, and consultation services. He was recently named to Team Macromedia.

HOSTMYSITE.COM

www.hostmysite.com

Stimulants and Science Fiction

Marrying Java with ColdFusion

BY
CURTIS
SCHLAK

I'm writing my Congressional representative. I'm going to ask him if he can introduce a bill on the floor to declare May 29 as "Curtis Is Really Happy Day."

One of his assistants will ask about my motivations, of course. I'll just forward the e-mail I received the morning of May 29, 2002, announcing the commercial release of ColdFusion MX. My representative's assistant will peg me as some weirdo and forward my contact information to the FBI where they can just add it to the fat file sitting on AD Skinner's desk. It's worth it, though.

If you don't already know, CFMX is built on Java 2, Enterprise Edition (J2EE) technology. This means that the voodoo that the ColdFusion Application Server used to perform on ColdFusion templates during the compilation process is no longer practiced; ColdFusion now compiles ColdFusion templates to Java bytecode that can be run with CFMX or Macromedia ColdFusion MX for IBM WebSphere.

In the middle of my celebration of this amazing development, I realized that the entire subject matter for this article had changed. I had to change the way I approached the topic and the examples in it. It would be pretty silly to read an article about Java and XML when that functionality is built into CFMX.

I don't expect that everyone reading this has gone out and purchased CFMX. I still develop in ColdFusion 4.0 for a client of mine who refuses to invest in upgrading. They also run MS WinNT 4.0 on a DEC Alpha machine with MS SQL Server 7.0, so I don't see their purchase of CFMX occurring anytime soon. Even the deep-pocket clients I develop for don't have plans for purchasing a CFMX upgrade until Q4.

With these factors in mind, it only seems reasonable to develop the same miniapplication for both the ColdFusion 5.0 and ColdFusion

MX architectures, highlighting the power of CFMX along the way. If you don't have CFMX and would like to complete the parallel development, you can download the CFMX trial version from the Macromedia site.

The Quandary

ColdFusion is great at retrieving data and formatting it for display; however, until CFMX it has only been a scripting language. Version 5 introduced user-defined functions that altered the top-to-bottom processing of ColdFusion templates. CFMX introduces Cold Fusion Components, which seems like a tag-based way of creating classes, a quantum leap for the language.

Now that developers can create CFCs, the question is, Why write external Java classes to supplement the functionality of ColdFusion when ColdFusion MX has so many features? The answer is quite simple, really. Java has a rich set of classes and interfaces that can make a developer's work easier. During the course of this article, we'll leverage a small portion of this library to analyze a set of simple data with some complex relationships. We'll implement it as a ColdFusion extension (CFX), as an object instantiated through the use of `<CFOBJECT>`, and as a servlet integrated into the CFMX architecture.

The Product

This article will analyze a two-player, online version of Go. For those of you who play Go, you know how intense and complex the game is. If you're unfamiliar with the game, there are only nine rules:

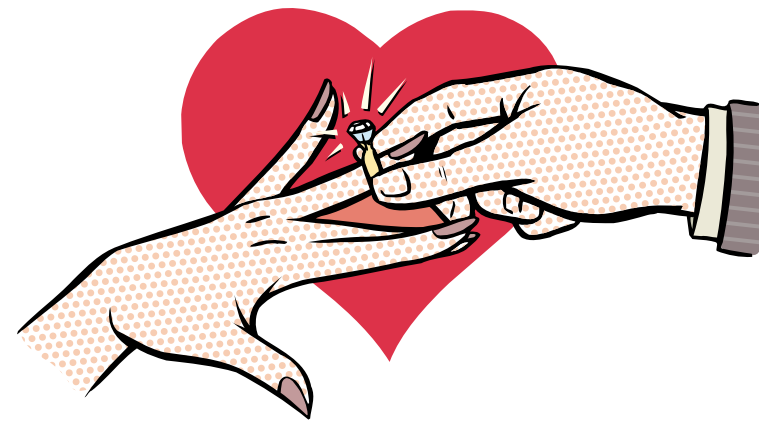
1. The board starts out empty.
2. A move consists of playing a piece on an empty intersection.

3. A piece or a solidly connected group of pieces is captured when completely surrounded by the opponent's pieces.
4. Captured pieces are removed from the board.
5. The previous board position may not be re-created.
6. A player may not place a stone such that it would cause the immediate capture of that stone or the group to which it belongs.
7. A player may pass on any turn.
8. The game ends when both players pass in sequence.
9. The player that captures the largest portion of the board wins.

Without using objects, a programmer most probably will resort to recursion to analyze the pieces on the board. This type of programming can be expensive in terms of resources and time. Creating the appropriate objects to represent the game board and groups of pieces can allow for easy analysis of the board.

This architecture will implement the analysis portion of those rules using Java classes to analyze each move's validity and update the structure that represents the board. We'll use ColdFusion to create the HTML to present the information and to retrieve it from WDDX packets saved to the server.

The serialized game data structures will be stored in two standard XML templates: `games.wddx` and `template.wddx`. The WDDX schema was chosen because it's the only XML schema that ColdFusion 5 supports natively. CFMX, built on Java technology that has had XML parsing support for a long time, can use XML packets with any schema, including WDDX. Hence, for interoperability between the two server versions, WDDX is our choice.



Stored in `games.wddx` is an array of structures serialized as a WDDX packet. The structure holds the create date of the game, the name of the game, and the file in which the individual game information has been serialized. Those individual game files are stored in the data directory. The choice to use WDDX in this instance is merely to circumnavigate the need for a database.

Stored in `template.wddx` is the initial structure of a game, where both players have captured none of their respective opponent's pieces and the Go board is empty. To allow for a clearer analysis of the data in this example code, the board is stored as a query serialized into a WDDX packet with 19 columns and 19 rows. There are more economical ways to store the data. This is not production-quality data storage, but example-quality data storage.

The Code

The ColdFusion code remains the same through all the different implementations with the exception of the template that calls the validation algorithms. Memory variables, such as `SERVER-`, `APPLICATION-`, and `SESSION-` scoped variables, aren't present in this application; however, it's important to point out that ColdFusion MX allows the use of J2EE session identifiers in addition to the `CFID` and `CFTOKEN` values to which we've all become accustomed. If the "Use J2EE sessions" option is turned on in the CFMX Administrator, then servlets and JavaServer Pages can share information that's stored in the `APPLICATION` and `SESSION` scopes. While we won't see this in the code, I'll point out possible uses at the appropriate times.

The code must check the version of the ColdFusion server that's running this game because it's supported only by ColdFusion 5.0 and higher. The CFX implementation would probably run on ColdFusion 4.5, but it hasn't been tested. This version test is performed as the first step in the `Application.cfm`. If a nonvalid version of ColdFusion is found to be running, the user is relocated to `error.htm`, a simple page that informs users to upgrade their version of ColdFusion. The `Application.cfm` also sets up the variables that contain the path to the file that contains a WDDX packet listing the current games, a variable that points to the current-move validation template, and a portion of code that intercepts the request to create a new game. That interception appends the new game to the game details structure and reserializes it for storage. There are also instructions for configuring each of the validation templates and their resources.

The default template, `index.cfm`, reads the data from `games.wddx` and displays it, as well as presents a text field to the player to create a new game. Both the form and the links surrounding the game titles point to `board.cfm`.

The actual game is played from `board.cfm`, where a graphical representation of the board is drawn. Each image of the playing field contains an intersection; if the intersection is empty (there's no piece on it), then it's a hyperlink to a validation page, where the interesting portion of the code exists.

`OnRequestEnd.cfm` uses a `CFDUMP` on the `REQUEST` scope if the parameter `request.debug.showStructures` is present and evaluates to a value that's considered `TRUE` by ColdFusion. That dump of the con-

tents of the `REQUEST` scope shows all the pertinent information gathered by the data retrieval processes.

There are four validation templates: `validateMoveCFX.cfm`, `validateMoveCFOBJECT5.cfm`, `validateMoveCFOBJECTMX.cfm`, and `validateMoveServlet.cfm`. The servlet will be used to highlight one of the new features of CFMX, using servlets by including them directly into ColdFusion. Each of the templates is named with its respective mode of move validation.

Two basic Java classes are used to model the Go board: `Board` and `Group`, which are stored in the subdirectory `com/grayiris/go` and belong to that package. The `Group` object holds information about how many pieces are in the group and how many empty spaces there are around the group. If the number of those empty spaces reaches zero, then that group is captured and will be removed from the board. The number of pieces in that group will be added to the current player's prisoner total.

Board models all the states that the board has gone through in the course of a game and which empty spaces on the board are adjacent to which `Group` object. It also contains `Group` objects that represent the groups of pieces on the Go board for the current state of the board. `BoardFactory` contains a static method `createBoard()` that will initialize a board object from a complex structure passed to it. All that's left is to integrate these into the ColdFusion application.

CFX Implementation (CF5 & CFMX)

If you've never used a CFX tag, it's like using a custom tag written in ColdFusion Markup Language except that the syntax is slightly different. When invoking a custom tag written in CFML, you type `<cf_filename>` and ColdFusion looks around for the file and runs it in its own process. CFXs are similar, but are written in Java or C++. You must register the CFX in the ColdFusion Administrator on the "CFX Tags" page. One of the major advantages of writing CFXs in Java is thread safety; since each Java CFX class has its own associated `ClassLoader` that loads it and any dependent classes, per-request information is safe to store in the CFX.

The C++ version isn't done this way, and the developer must make the code thread-safe, adding another level of tedium. But if you have a performance-sensitive application and need the fastest code possible, the C++ extension is really the way to go. For this article, however, all information in this section is applicable only to CFXs written in Java.

CFXs have a query attribute through which a reference to a ColdFusion query object is passed. That query and its associated meta-information, such as row count and column names, can be retrieved through the `getQuery()` method of the Query Interface of the ColdFusion extension API. The query passed into `CFX_ValidateGoMove` is the query that's retrieved from the WDDX packet stored in the game file. It contains the history of the board from the first to the current move. The CFX instantiates a Board object and populates it with the information from the query. How the CFX is used is shown in Listing 1.

If the move succeeds, a variable with a name equal to the string passed into the CFX in the `invalidMoveBoolName` attribute will contain a 1. A WDDX packet will be created in the variable with the name that you passed into the `newBoardWDDXName`, which can be saved directly to the game file. If the move failed, then a variable with a name equal to the string passed into the CFX in the `invalidMoveBoolName` attribute will contain a 0 and a variable will be created with the name passed in through the `reportStringName` attribute. We pass that back as the error message.

CFOBJECT Implementation (CF5 & CFMX)

The difference in implementing the Board object with CFOBJECT in ColdFusion 5 and CFMX is the method of delivery of the information to the object created by CFOBJECT. CF 5 doesn't support automatic conversion of Date/Time objects and structures to Java objects. CFMX, now built in Java, has a greater capacity for sending and receiving information from ColdFusion with automatic conversion of structures, arrays of different element types, and queries. (The CFMX documentation states that a

ColdFusion Query object can be cast as a Java object implementing the Map interface. However, with the version I'm running [6,0,0,46617] I can't get this functionality to work.) This will allow us to pass the information to our object more easily in CFMX than in CF5. The latter implementation is the file `validateMoveCFOBJECT5.cfm`; the CFMX implementation is the file `validateMoveCFOBJECTMX.cfm`.

In CF5 the Board object is created and then populated with the information from our query by repeating a call to the `setBoardState()` function. In CFMX we can send the entire structure built from the game file to the BoardFactory and it will return a Board object. After these objects have been created and their information loaded, we call the `makeMove()` method, which returns a Boolean value, true if the move succeeded and false if it failed. From that information we fork the code, sending the reason for failure back to the browser, or retrieving the WDDX packet for the new game state with the `getWDDXBoard()` method, and save that string to a file.

Servlet Implementation (CFMX Only)

I won't lie to you: this implementation is the most exciting for me. It allows the seamless integration of JSPs and Java servlets with ColdFusion templates.

The ColdFusion template `validateMoveServlet.cfm` reads the information from the game file, just like the rest. It then includes the servlet, which can access that structure through the request scope with the `getParameter()` method found in the

request interface. The structure is then cast as a Map and everything works as before. A BoardFactory returns the appropriate Board object from the query stored in the structure. The same processing is completed and the servlet then sets variables in the request scope that ColdFusion can then access. Voilà!

A Marriage Made in Heaven

Java is an appropriate solution in many situations. ColdFusion has allowed us to harness that power for a few years. However, with the release of ColdFusion MX, the power of Java now propels ColdFusion. We can integrate servlets and JSPs directly into our applications without the overhead of creating HTTP requests and waiting for answers. We can now share APPLICATION- and SESSION-scoped variables with these previously untouchable objects. CFMX allows for tighter integration of data through automatic conversion of native ColdFusion data types to native Java types.

Resources

- *Java 2 Platform, Enterprise Edition*: <http://java.sun.com/j2ee/>
- *Macromedia ColdFusion MX for IBM WebSphere*: www.ibm.com/software/webservers/coldfusion/mx/
- *Macromedia MX - Free Product Evaluations*: www.macromedia.com/software/trial_download/
- *Go Rules*: www.usgo.org/

@ CURTIS_SCHLAK@YAHOO.COM

Listing 1

```
<CFX_ValidateGoMove
query="[the query that contains the board state]"
lastMove="[the number of the last move]"
row="[the row number of the current move]"
col="[the column number of the current move]"
reportStringName="[variable name that should contain reason for failure]"
newGameWDDXName="[variable name that holds new board as wddx]"
invalidMoveBoolName="[variable name that indicates if move is invalid]"
```

CODE LISTING

The code listing for this article is also located at

www.sys-con.com/coldfusion/sourceec.cfm

FREE TUTORIAL
JUNE 27TH WITH WEB SERVICES EDGE
2002 EAST REGISTRATION. LIMITED OFFER!

web services **EDGE**
world tour 2002

TO REGISTER: www.sys-con.com or Call 201 802-3069

\$195

REGISTRATION
FOR SYS-CON
SUBSCRIBERS
BEST EDUCATIONAL VALUE
FOR THE HOTTEST
TECHNOLOGY SKILLS!

Take Your Career to the Next Level!

SHARPEN YOUR
PROFESSIONAL SKILLS.

KEEP UP WITH THE
TECHNOLOGY
EVOLUTION!

"Presented an excellent overview of Web services. Great inside knowledge of both the new and old protocols. Great insight into the code piece."

— Rodrigo Frontecilla

"Very articulate on the Web services SOAP topic and well-prepared for many questions. I've learned a lot from this seminar and I appreciate this seminar for my job. Thank you!"

— Kenneth Unpingco, Southern Wine & Spirits of America

"I liked the overview of Web services and the use of specific tools to display ways to distribute Web services. Good for getting up to speed on the concepts."

— B. Ashton, Stopjetlag.com

Echoed over and over by
Web Services Edge World Tour
Attendees:

"Good balance of theory and demonstration."

"Excellent scope and depth for my background at this time. Use of examples was good."

"It was tailored toward my needs as a novice to SOAP Web services – and they explained everything."

WHO SHOULD ATTEND:

- Architects
- Developers
- Programmers
- IS/IT Managers
- C-Level Executives
- i-Technology Professionals

Learn How to Create, Test and Deploy Enterprise-Class Web Services Applications

TAUGHT BY THE INNOVATORS
AND THOUGHT LEADERS IN
WEB SERVICES

EXPERT PRACTITIONERS TAKING AN APPLIED APPROACH WILL PRESENT TOPICS INCLUDING BASIC TECHNOLOGIES SUCH AS SOAP, WSDL, UDDI AND XML, PLUS MORE ADVANCED ISSUES SUCH AS SECURITY, EXPOSING LEGACY SYSTEMS AND REMOTE REFERENCES.

RECEIVE **4**
FREE \$345 VALUE!
1-YEAR SUBSCRIPTIONS
1 2 3 4 OR 4



SEATING IS LIMITED. REGISTER NOW FOR THE CITY NEAREST YOU!
WWW.SYS-CON.COM

IF YOU
MISSED THESE...

BOSTON, MA (Boston Marriott Newton) **SOLD OUT!**
WASHINGTON, DC (Tysons Corner Marriott) **SOLD OUT!**
NEW YORK, NY (Doubletree Guest Suites) **SOLD OUT!**
SAN FRANCISCO, CA (Marriott San Francisco) **SOLD OUT!** **CLASSES ADDED**

BE SURE NOT TO
MISS THESE...

Each city will be sponsored by a leading Web services company

...COMING TO A CITY NEAR YOU

2002
NEW YORK AT WEBSERVICES EDGE 2002 **EAST**JUNE 27
SAN FRANCISCOAUGUST 6
SEATTLEAUGUST 27
AUSTINSEPTEMBER 10
LOS ANGELESSEPTEMBER 19
SAN JOSE AT WEBSERVICES EDGE 2002OCTOBER 1
CHICAGO**WEST**OCTOBER 17
ATLANTAOCTOBER 29
MINNEAPOLISNOVEMBER 7
NEW YORKNOVEMBER 18
SAN FRANCISCODECEMBER 3
BOSTONDECEMBER 12

2003
CHARLOTTEJANUARY 7
MIAMIJANUARY 14
DALLASFEBRUARY 4
BALTIMOREFEBRUARY 20
BOSTONMARCH 11

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE LOWEST REGISTRATION FEE.

TOPICS HAVE INCLUDED: Developing SOAP Web Services
Architecting J2EE Web Services

On May 13th, the San Francisco tutorial drew a record 601 registrations.

i-TECHNOLOGY
SYS-CON EDUCATION

REGISTRATION FOR EACH CITY CLOSES THREE BUSINESS DAYS BEFORE EACH TUTORIAL DATE. DON'T DELAY. SEATING IS LIMITED. NON-SUBSCRIBERS: REGISTER FOR \$245 AND RECEIVE THREE FREE ONE-YEAR SUBSCRIPTIONS TO *WEB SERVICES JOURNAL*, *JAVA DEVELOPER'S JOURNAL*, AND *XML JOURNAL*, PLUS YOUR CHOICE OF *BEA WEBLOGIC DEVELOPER'S JOURNAL* OR *WEBSERVICES DEVELOPER'S JOURNAL*, A \$345 VALUE!

TO REGISTER:
www.sys-con.com
or Call 201 802-3069

ABOUT THE
AUTHOR
While in the Army,
Curtis Schlak
integrated Microsoft
products with legacy
dBase logistics
software and wrote
his first Web
application. Formerly
a senior director of
development
(ColdFusion) at
KickFire, Inc., in
Saratoga, CA, he has
now started a new
company, Striatum
Solutions, in Houston.

Alabama
Birmingham, AL CFUG
www.bcfulg.org

Alabama
Huntsville, AL CFUG
www.nacfulg.com

Alaska
Anchorage, AK CFUG
www.akcfulg.org

Arizona
Phoenix, AZ CFUG
www.azcfulg.com

Arizona
Tucson, AZ CFUG
www.tucsoncfug.org

California
Bay Area CFUG
www.bacfulg.net

California, Fresno
Central California CFUG
www.cccfulg.com

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

California
Los Angeles CFUG
www.sccfulg.org

California
Orange County CFUG
www.sccfulg.org

California
Sacramento, CA CFUG
www.saccfulg.org

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

Southern California
Southern California CFUG
www.sccfulg.org

Northern Colorado
Northern Colorado CFUG
www.nccfulg.com

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

Delaware, Dover
Delaware CFUG
www.decfug.org

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

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

Florida
Orlando, FL CFUG
www.cforlando.com/

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

Florida
Tallahassee, FL CFUG
www.tcfug.com/

Florida
Tampa, FL CFUG
www.tbcfug.org

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

Atlanta
Georgia CFUG
www.cfugorama.com

Georgia
Columbus, GA CFUG
www.vcfug.org

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

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

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

Indiana
Indianapolis, IN CFUG
www.hoosierfusion.com

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

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

Kentucky
Louisville, KY CFUG
www.loulexcfug.com

Louisiana
New Orleans, LA CFUG
www.nocfulg.org

Maryland
Annapolis, MD CFUG
www.ancfulg.com

Maryland
Baltimore, MD CFUG
www.cfugorama.com

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

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

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

Massachusetts
Boston, MA CFUG
www.cfulgboston.org

Michigan, Dearborn
Mmaniacs CFUG
http://ciwebstudio.com/MMania/MMania.htm

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

Minnesota, Minneapolis
Twin Cities CFUG
www.colderfusion.com

Missouri
Kansas City, MO CFUG
www.kcfusion.org

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

Montana, Helena
Montana CFUG
http://montanacfulg.site-o-matic.com

Nebraska
Omaha, NE CFUG
www.necfulg.com

Nevada
Las Vegas, NV CFUG
www.sncfulg.com

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

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

New York
Albany, NY CFUG
www.anycfug.org

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

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

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

New York
Syracuse, NY CFUG
www.cfulgcn.org

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

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

North Carolina
Raleigh, NC CFUG
www.ccfug.org

Ohio, Columbus
Ohio Area CFUG
www.oacfulg.org

Ohio
Mid Ohio Valley MMUG
www.movcfug.org/

Oklahoma
Oklahoma City, OK CFUG
http://idgweb4.ouhsc.edu/cfulg/

Oklahoma
Tulsa, OK MMUG
www.tulsacfulg.org

Oregon
Eugene, OR CFUG
www.EugeneCFUG.org

Oregon
Portland, OR CFUG
www.pdxcfug.org

Pennsylvania, Harrisburg
Central Penn CFUG
www.centralpenncfug.org

Pennsylvania
Philadelphia, PA CFUG
www.pacfulg.org

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

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

Rhode Island
Providence, RI CFUG
www.ricfulg.com

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

Tennessee
Nashville, TN CFUG
www.ncfulg.org

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

Texas
Dallas, TX CFUG
www.dfwcfug.org/

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

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

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

Virginia
Hampton Roads CFUG
www.hrcfulg.org

Virginia
Northern Virginia CFUG
www.cfugorama.com

Virginia
Richmond, VA CFUG
http://richmond-cfulg.btgi.net

Virginia, Roanoke
Blue Ridge MMUG
www.brmug.com/

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

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

Wyoming, Jackson
Wyoming MMUG
www.wycfug.org

International

Australia-Melbourne
Australian CFUGs
www.cfulg.org.au/

Belgium, Brussels
Belgium CFUG
www.cfulg.be

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

Canada
Edmonton, AB CFUG
http://edmonton.cfulg.ca/

Canada
Kingston, ON CFUG
www.kcfug.org

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

Canada
Ottawa, ON CFUG
www.cfulgottawa.com

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

Canada
Toronto, ON CFUG
www.cfulgtoronto.org

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

Canada
Vancouver Island CFUG
www.cfulg-vancouverisland.com

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

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

France, Valbonne
France CFUG
www.cfulg.fr.st/

Germany, Frankfurt
Frankfurt CFUG
www.cfulg-frankfurt.de/

Ireland
Belfast, Ireland CFUG
www.cfulg.ie

Ireland
Cork, Ireland CFUG
http://viewmylist.com/Cork/

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

Malaysia, Kuala Lumpur
Malaysia CFUG
www.coldfusioner.com/

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

Pakistan, Lahore
Pakistan CFUG
www.cfulgpakistan.org

Saudi Arabia, Riyadh
CFUG Saudia
www.cfulgsaudia.org/

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

Switzerland
Martin Bürlmann
Switzerland CFUG
www.cfulg.ch

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

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

Turkey, Izmer
Turkey CFUG
www.cfr.net

Thailand
Bangkok, Thailand CFUG
http://thaicfulg.tei.or.th/

United Kingdom, London
UK CFUG
www.ukcfug.org

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

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

Subscribe Now

for Instant Access to

CF Advisor

CFAdvisor.com!

12 months
\$89
24 months
\$169

Now on CD!

THE MOST
COMPLETE LIBRARY
OF EXCLUSIVE
CF ADVISOR
ARTICLES!

3
YEARS
40
ISSUES
200
ARTICLES
ONE CD

Since May 1998, CF Advisor™ has provided the CF community with in-depth information that quickly pays dividends on the small subscription investment. For just \$89.00, you get 12 full months of uninterrupted access to CF Advisor™ – including:

- usable code
- skill-building articles
- tips
- news updates
- conference reports
- interviews with industry movers-and-shakers.

You get full access to our sizable archives covering such subjects as:

- custom tags
- functions
- variables
- framesets
- style sheets
- structures
- javascript usage
- debugging techniques
- programming practices and much, much more...

THE MOST COMPLETE LIBRARY OF
EXCLUSIVE CF ADVISOR
ARTICLES ON ONE CD!

"The Complete Works"

CD is edited by ColdFusion Developer's Journal Editor-in-Chief Robert Diamond and organized into 21 chapters containing more than 200 exclusive CF Advisor articles.

Easy-to-navigate HTML format!

E-Commerce
Interviews
Custom Tags
Fusebox
Editorials
Databases
News
CF & Java

CFBasics
Reviews
Scalability
Enterprise CF
Wireless
Verity
Source Code CF
Applications

Object-Oriented CF
WDDX
Upgrading CF
Programming Tips
CF Tips & Techniques
ProgrammingTechniques
Forms

Order Online and Save 10% or More!
WWW.JDJSTORE.com
OFFER SUBJECT TO CHANGE WITHOUT NOTICE



Subscribing now guarantees that you'll immediately receive access to everything you need for keeping up-to-date with ColdFusion, no matter where you are. All the content is available exclusively over the Net and is regularly updated throughout the month.



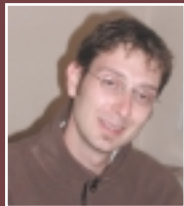
135 CHESTNUT RIDGE ROAD
MONTVALE, NJ 07645
WWW.SYS-CON.COM



CF Community

Tales from the List

Exploring CFCs



BY SIMON HORWITH

This month, as predicted, ColdFusion MX questions abound on the **CFDJList** as many developers attempted to create ColdFusion Components for the first time. While the majority of these posts turned out to be simple syntactical errors, **Ray Camden** did chime in from time to time offering links to CFC code samples for practice applications he's in the process of building. A tic-tac-toe component application can be found at www.camdenfamily.com/morpheus/cf_cfc.cfm. Ray also announced the soon-to-be-released cfc-zone.org Web site – an open source CFC site similar to cflib.org (for UDFs).

Another interesting CFC thread began when **Kevin Bridges** wrote stating that he's interested in using CFCs to build a suite of security Web services that connect to databases to authenticate users and their roles for applications. The idea is to be able to build Web sites in “free” languages such as PHP or ASP and access the Web services written in CFML in order to perform most security and authentication functionality. Several members replied and notified Kevin that this certainly can be done; both PHP and ASP can call Web services. I replied to the post and sparked a discussion on CFC best practices.


My reply concurred with the others that what Kevin is trying to do certainly can be done, but I pose the question whether it *should* be done. Object-oriented programming best practices and object modeling pattern theory tell us that an event should be associated directly with the object upon which it performs. Better than having a security object with a `verifyuser()` method called by all objects, an application should have a user object that has a `verify()` method. I also question this practice because it's entirely conceivable that at some point in the future a client may ask that the security framework change. Changing a “shared method” might solve one client's needs but could prove disastrous for others, or could result in code that's more difficult to maintain. It also doesn't make much sense to me to have security information from several unrelated applications stored in one database; that information should be kept local and private to an application. The thread definitely made apparent the need for those developers in the community who understand object modeling to help to enlighten others and to define best practices for CFC as well as Web services use.

In an unrelated post **David Sammons** reported a database error from inside a CFC that he couldn't

get to work properly. When executing a query against his Oracle database using Macromedia's JDBC driver, the database driver returned an error that his IN statement was missing a right parenthesis, which it clearly wasn't. The SQL syntax looked absolutely fine – in fact, it ran without error in other applications. **Ray Thompson**, Ray Camden, and I had David jump through dozens of hoops trying our suggestions – still no success. One possibility we considered was a difference between the ODBC and JDBC drivers, as this has been the culprit in several errors recently. No luck – the SQL when hard-coded in a `cfquery` tag worked fine.

Now we all know that there's no logical reason for a query not to work in a CFC if it does work in a CFML template, so the error must be coming from some other code. This began a short discussion between David and me about (more) CFC best practices. I reminded everyone on the list that components should be tested independent of the rest of an application. Object-oriented programming is as efficient and as widely adopted as it is due to the rock-solid (performing) and easy-to-maintain applications that result when you break an application into a modular format and test those modules as stand-alone apps so that when they're all pieced together, everything works like a well-oiled machine.

I received a ZIP file containing David's code this morning and just finished testing it. As suspected, the error messages he was receiving had nothing to do with the query itself, but with an unrelated error on the page that instantiated the component. One bug I did find was that the CFC was extending itself, a coding no-no. So, syntax errors and not testing components in a modularized manner are to blame for the reported error. As I write this, I've posted the results of my tests to **CFDJ**.

The volume of posts about ColdFusion Components has made it obvious that many developers out there are very excited about this new technology and are eager to implement it in their applications. CFCs are still relatively new, and best practices haven't been agreed on yet; in the meantime, developers are going to continue to send e-mails to the list in an effort to become fluent in CFC lore. That's one of the most exciting things about ColdFusion MX: with every `<CFCOMPONENT>` tag we write, we're exploring uncharted CFML territory. 

@ SHORWITH@FIGLEAF.COM

WWW
WWW.WWWWWW.COM

ABOUT THE
AUTHOR
Simon Horwith, a senior developer and Macromedia-certified ColdFusion instructor at Fig Leaf Software in Washington, DC, has been using ColdFusion since version 1.5. He is a contributing author to Professional ColdFusion 5.0 (WROX) as well as technical editor of The ColdFusion 5.0 Certification Study Guide (Syngress).

New Fusebox Book Released (Las Vegas) – Fusebox, a widely adopted architectural framework for building Web applications, has been used on some of the largest ColdFusion-based Web applications and Web sites. Interested programmers and managers can now learn more from a new book, *Discovering Fusebox 3*, written by Hal Helms and John Quarto-vonTivadar. The book, published by Techspedition, explains how to use the Fusebox application framework and is targeted toward the working programmer with



no previous knowledge of Fusebox. For more information, or to purchase the book online, go to www.techspedition.com.



TrueSpectra Integrates with Dreamweaver Platform (San Mateo, CA) – TrueSpectra, Inc., has released their Image Server extensions for Macromedia Dreamweaver MX,



Dreamweaver 4, and Dreamweaver UltraDev 4, providing Web developers and designers a faster, easier way to incorporate server-side rich image media into Web sites.

TrueSpectra Image Server allows developers to modify images on-the-fly using a

Flash Communication Server MX Now Available

(San Francisco) – Macromedia Flash Communication Server MX, the first server to unite communications and applications, is now available from Macromedia. The new server integrates support for streaming media, real-time collaboration, and multiway video, audio, and text messaging into a single solution.



For development, the communication server includes new extensions to the Macromedia Flash MX authoring environment. The extensions include new authoring, scripting, and debugging capabilities, as well as prebuilt components for login, multiway chat screens, white boards, and video conferencing.



www.macromedia.com

standard HTTP URL query string called the Image URL, eliminating many of the manual editing tasks in the image production workflow.



Developers can download the extensions for free at the TrueSpectra Web site. www.truespectra.com

e-Zone Media Releases FuseTalk 3.1



(Ottawa) – e-Zone Media Inc.

has released FuseTalk 3.1, a discussion forum solution for ColdFusion environments. Enhancements in FuseTalk 3.1 include a full Java-based graphing engine for reporting, improved survey reports, and



more verification and e-mail capabilities.

www.e-zonemedia.com

IMlogic Adds J.J. Allaire to Board of Directors

(Boston) – IMlogic Inc., a leading provider of enter-

prise applications for the instant messaging market, has announced the addition of J.J. Allaire to its board of directors.



Allaire, founder of Allaire Corporation (now Macromedia), created its flagship product, ColdFusion. He joins the board to help guide the company's growth in enterprise IM penetration and platform development.

www.imlogic.com

EVENTS Macromedia DevCon 2002

October 27–30
Lake Buena Vista, FL
www.macromedia.com/v1/conference/

CF Cruise 2003

Caribbean cruise –
departs from Tampa
www.cfconf.org/cf_cruise

CFDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
ACTIVE PDF	WWW.ACTIVEPDF.COM	949.582.9002	4
CFADVISOR	WWW.CFADVISOR.COM		47
CFDYNAMICS	WWW.CFDYNAMICS.COM	866.CFDYNAMICS	13
CFXHOSTING	WWW.CFXHOSTING.COM	866.CFXHOSTING.COM	29
CTIA	WWW.CTIAHOW.COM		23
ENGINDATA RESEARCH	WWW.ENGINDATA.COM	201.802.3082	51
E-ZONE MEDIA	WWW.FUSETALK.COM	866.477.7542	33
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.HOST	41
INTERMEDIA	WWW.INTERMEDIA.NET	800.379.7729	52
MACROMEDIA	WWW.MACROMEDIA.COM/GO/CFMXAD	877.460.8679	9
MACROMEDIA	WWW.MACROMEDIA.COM/GO/CFJDEVCON2002	877.460.8679	11
NEW ATLANTA	WWW.NEWATLANTA.COM		2
PACIFIC ONLINE	WWW.PACONLINE.NET	877.503.9870	25
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	15
RACKSHACK	WWW.RACKSHACK.NET	800.504.SURF	3
SYS-CON MEDIA	WWW.SYS-CON.COM/SUBOFFER.CFM	201.802.3069	37
TERATECH INC.	WWW.CFCONF.ORG/CF_UNDERGROUND4/	800.447.9120	49
WEB SERVICES EDGE	WWW.SYS-CON.COM	201.802.3069	45
WEB SERVICES EDGE WEST	WWW.SYS-CON.COM	201.802.3069	39

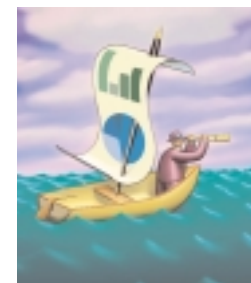
General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of *ColdFusion Developers Journal*. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or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

Chart Your Course to Success in IT...

...order your copy of Java Trends: 2003

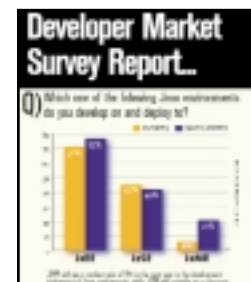


Don't go astray. In the vast sea of Internet technology, market conditions change constantly. Will Java remain the hot platform it is today? Will C# rapidly gain ground? What are the world's foremost Java developers aiming their sights toward? Which companies come to mind when planning their next project? How can their thinking direct your efforts?



EnginData Research has studied the IT industry extensively, spotting many key trends and alerting industry leaders along the way. In the first quarter of 2002, they embarked on their most challenging mission ever: the most in-depth study ever done of the Java development marketplace.

After months of analyzing and cross-referencing more than 10,500 comprehensive questionnaires, the results are now available.



Here's just a sample of the critical data points the Java report delivers...

- ✓ Current and projected usage of languages and platforms
- ✓ Multiple rankings of hundreds of software vendors and tools
- ✓ Types of applications being developed
- ✓ Databases being used
- ✓ Purchasing patterns
- ✓ Company size
- ✓ Development and purchasing timelines
- ✓ Perceptions and usage of Web services
- ✓ Preferred Java IDE
- ✓ J2EE, J2ME, J2SE usage comparisons

As an IT specialist, EnginData Research focuses exclusively on three leading drivers in IT today – Java, Web services, and wireless development. Coming soon, our Web services survey will deliver such critical knowledge as:

- ✓ Time frame for developing Web services – enabled apps
- ✓ Percentage of apps with Web services today
- ✓ Sourcing and hosting Web services
- ✓ Perceived leaders in Web services tools and solutions

Navigate your company through the challenging IT marketplace with the trusted knowledge and intelligence of EnginData Research. Order your copy of the 2002–2003 Java market study by calling Margie Downs at 201-802-3082, or visiting our Web site.



www.engindata.com

Intermedia
www.intermedia.net