CREATING AN IM BOT USING CF *PG 14*

COLDFUSION Developer's Journal

Implementing an SSO Solution Using ColdFusion

PG 22

Constructing a single sign-on solution for Web applications

PLUS...

One little box.
A whole lot of power.

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

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

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

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

› Make rapid J2EE development a reality.

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

› New mobile applications are a go.

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

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

The Ever-Changing Face of CFDJ

By Simon Horwith

It was exactly one year ago that *ColdFusion Developer's Journal* began focusing heavily on a different topic every month. The feedback that I hear from our readers at conferences and via email indicates that it has been a well-received change in format. While there is no plan to stop this practice, there are other ideas that I'd like to experiment with in order to try and give our readers the most useful content possible. The specialized focus issue format was introduced in order to make each month's issue more informative and to offer something for everyone, regardless of the level of expertise. The new format will be to have both focus and non-focus issues, but with certain common themes that are touched upon all year round.

The idea is that with deliberate focus, *CFDJ* will become the most valuable resource possible for CF developers. I've been spending a lot of time recently looking at and thinking about trends in technology and on the web. I've been talking with different folks at Macromedia about some of their future initiatives, including how their vision of the future is evolving and/or coming into realization. I've also been looking at the newest research from Microsoft, Sun, Adobe, and a couple of other companies. Last, but not least, I've been paying special attention to the trends, discussions, and opinions within the ColdFusion development community. *CFDJ* is going to place more emphasis on these trends and on deficiencies that I believe require special attention. *CFDJ* will strive to focus on topics that will make all of our readers more marketable and that will raise the quality of applications being developed by educating less about niche topics and more about topics that affect us all. In short, my goal is to make our readers more successful. This may be a tall order, but if nothing else it's a noble effort.

In order for this to work, I am going to rely on the community more heavily than ever to submit great articles and topic ideas. Please check my blogs at http://www.horwith.com or http://simon.coldfusionjournal.com from time to time to find out what topics I'm looking for authors to write about. I currently have several topics that are going to be woven into *CFDJ* in upcoming issues. Many of them deal with technologies and products other than ColdFusion, but with a focus on introducing how they work and how we can take advantage of them from our CF apps. Of particular interest is Flex and Rich Internet Applications in general. Macromedia has said that RIAs are the future for several years now, and Microsoft will is finally be jumping on the band-

"*CFDJ* will strive to focus on topics that will make all of our readers more marketable"

wagon in the near future. Flash player support is finally coming to mainstream mobile devices in the U.S. Soon, I believe that RIA development will not only be a realistic cost-effective option for ColdFusion developers to deliver but will also be a pleasure for us to develop.

Other area I plan to pay more focus is frameworks, methodologies, and best practices in general – whether they're for application development, architecture, management, planning, testing, or deployment. I am writing this editorial a few days prior to the TeraTech Fusebox and Frameworks conference, which I'm really looking forward to. Having always been fairly vocal about not being much of a fusebox fan in the past, this year will be my first time at the confer-

http://coldfusionjournal.com for SYS-CON TV MAX podcasts and other conference reports and news. In addition to examining frameworks, expect coverage of the Fusebox and Frameworks, CFUnderground, and MAX conferences in next month's issue.

Turning our attention to the now, we have several great articles this month. By the time this is published, the CFMX 7 Updater will have been released. This month, Macromedia's Dave Gruber introduces the new features in ColdFusion MX 7.0.1. Some of the other notable articles this month include the return of BF on CF (Ben Forta) after several months on the road, another installment of Hal Helms' popular etude series, and an article from Rob Gonda about a very hot

"RIA development will not only be a realistic cost-effective option for ColdFusion developers to deliver but will also be a pleasure for us to develop"

ence, where I'm giving two presentations ("Introduction to Fusebox," which I had to learn for the conference, and "Developing Applications Without Frameworks"). While at the conference I will approach the leading experts for each of the major frameworks that are available and ask that they develop an application and contribute an article describing the code for next month's issue. Each framework will implement the same application. So, it will be a frameworks extravaganza!

I thought it would be interesting for our readers to get a side-by side comparison and explanation of all of the major frameworks that are out there.

I have recently agreed to release the methodology I use for development, so expect to see more on that as well. Just over two weeks from now I will be at the MAX conference in Anaheim (and CFUnderground on the Saturday prior to MAX registration). Check the *CFDJ* web site at

topic – AJAX.

I hope to meet many of you at the Fusebox and Frameworks, CFUnderground, and/or MAX conferences in the next several weeks. If you attend any of these events, be sure to stop by and say hello. As always, I welcome feedback about *CFDJ* and what you think of the direction it takes under my guidance. If you have thoughts or advice, or want to contribute, don't hesitate to email me at simon@horwith.com.

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

simon@horwith.com

Using Database Views and Stored Procedures

Offloading some of the processing from ColdFusion to the database server

By Jeff Houser

Most of us are never going to work on a Web site that gets the amount of traffic that Google or Yahoo (or MySpace) gets. Unfortunately, you can get away with bad coding practices on a small site that only gets 100 unique visitors a month and no one will ever know. With such a low visitor count, you'll never have to deal with the "why isn't my site loading quick enough" problem.

Unfortunately, a successful Web site will most likely grow. With growth come more users and more load. When it comes time to examine the application you'll find that more often than not the ColdFusion server is the cause of the bottleneck, and that the database server is barely being hit. Maybe there's a way to offload some of the work that the ColdFusion server is doing to the database server? Yes, there is and this article is going show you how you can use SQL Server views and Stored Procedures to offload some of the processing from ColdFusion to the database server. (Note that if you're looking for tips on designing your database structure they can be found in the April '05 issue of this column).

Using a View to Pre-Join Tables

Suppose you had an application that you designed to keep track of your record collection. You might design a database to look like what you see in Figure 1. The bands table, albums table, and songs table contain the content. An intersection table, bands_albums, connects the bands and albums table. This is a many-to-many relationship: A band can have many albums, and some albums, such as compilations or soundtracks often have multiple bands on them. The albums_songs table is another intersection table connecting the songs with the album

it appears on. This is another many-to-many relationship, since an album can (and better) have more then one song. And in many cases, an album song will end up on a compilation or movie soundtrack.

Now suppose you want to write a query that retrieves all the songs by a particular band. In ColdFusion it might look something like the JH Code Segment 1 at the end of this article. You're joining five tables to get the information you need, and this can be considered a fairly complicated join. If we were to move this code into an SQL Server view, you'd achieve efficiency. A view is a virtual table that lets you represent data in alternate ways. The most common way to create them is using Enterprise Manager, but you can also use T-SQL. In Enterprise Manager, you can follow these steps:

1. Launch Enterprise Manager, open your database server, and select your database.
2. Click the views selection in your database.
3. Right-click in the content window and select "New View."
4. You can create your code either using the graphic interface or just writing the SQL in the SQL Window. I copied and pasted the SQL code from the ColdFusion code.

Since a view is like a table, each column must have a unique name. When doing joins with tables that share common field names, you'll have to make sure that you select one of the fields, or give the duplicate an alias. In this case, the BandID and SongID are both in two tables. Instead of selecting both in our view, I specified one table over the other.
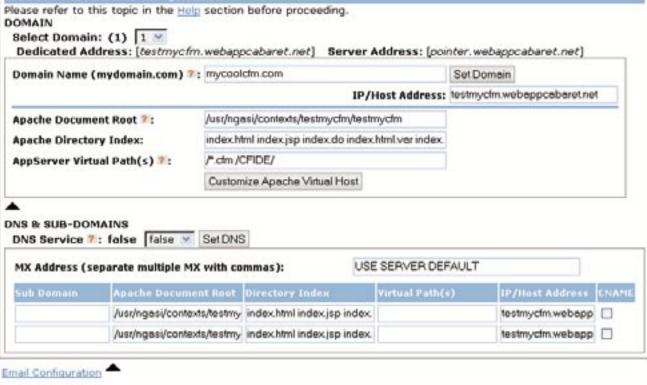
When writing ColdFusion code you can access the view just like a table, as shown in JH Code Segment 2. Notice that the view doesn't contain the query qualifier 'band = #variables.bandID#'. The view returns all the bands and their respective songs. SQL Server will optimize views using built-in optimization techniques. These techniques aren't applied to single queries. You can also index views the same way you index tables.

Advanced Logic in a Stored Procedure

Suppose that our record database is implemented using CFCs and a hint of object orientation. There's a band object that

Imagine a hosting company dedicated to meet the requirements for complex web sites and applications such as those developed with ColdFusion MX. At WebAppCabaret our standards based process and tools make deploying ColdFusion MX applications as easy as a point-and-click. We call it **Point-and-Deploy Hosting**. Our advanced NGASI Web Hosting management Control was designed for the hosting and management of ColdFusion web sites and applications thus cutting down on maintenance time.

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

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

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

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

WebAppCabaretsm   **http://www.webappcabaret.com/cdj.jsp   1.866.256.7973**

Point-and-Deploy J2EE Hosting
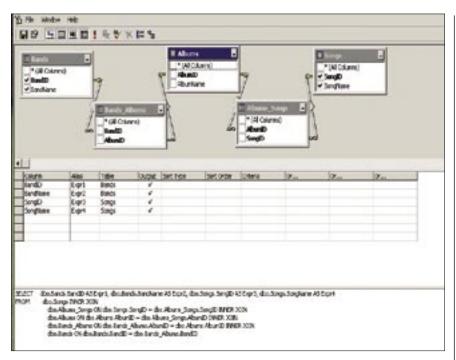
Figure 1: Record collection database diagram

contains an array of album objects that contains an array of song objects. When you init the band object, it runs a query to find out about the band's albums. In a loop it initializes the album objects, and each album initialization results in another query call to get the album information. In each album init function, a query is executed to get the song information. The song information is looped over to initialize the song objects. Each song makes yet-another query call. For a band with two 12-song albums, you'll end up with at least 27 different calls to the database. No wonder your application server is starting to smoke.

In many of my objects, I'll often implement an "initByData" method. Instead of a method that makes a database call and sets up the component, all data is manually passed into this method to set up the component. Using this method, I won't have to daisy chain calls to the database. So instead of 27 different database calls, I can use one query to get the band information, one query to get the band's album information, and one query to get the song information about each album. Our database calls went from 27 queries to three. Not bad.

Using a stored procedure, we can put

all three queries in a single stored procedure call, resulting in only one trip to the database. This will put more pressure on the database server, but will most likely give your ColdFusion application server a break. (It's been working hard, so it's earned it.) You can follow these steps to create a stored procedure:

1. Launch enterprise manager, open your database server, and select your database.
2. Click the Stored Procedure selection in your database.
3. Right-click in the content window and select "New Stored Procedure." Does this sound familiar yet?
4. Write your stored procedure. The finished code is in JH Code Segment 3.

The stored procedure code starts out with the "Create Procedure" command. Then comes the name of the stored procedure (GetBandInfo). Then a comma delimited list of all arguments to the stored procedure. In this case, we just have one argument, the bandID. Arguments are distinguished by using the @ symbol before the argument name. The keyword AS comes next and then the code that you want to run. This stored

procedure is a relatively simple one containing only selects. Stored procedures can also contain more advanced logic, including conditionals using if else, looping over recordsets through the use of cursors.

Stored procedures are precompiled objects. Unlike traditional queries (that you execute using the cfquery tag), the code doesn't have to be compiled first, which improves performance at runtime. You can execute a stored procedure using the cfquery tag using the 'exec' command, like JH Code Segment 4.

This method works, but only returns a single recordset to ColdFusion. It's undesirable in situations where you want to return multiple recordsets. The alternative is to use the cfstoredproc tag. Full documentation on the tag can be found at http://livedocs.macromedia. com/coldfusion/7/htmldocs/00000338. htm#2607555. Some attributes are similar to what you'd find in the cfquery tag, such as a datasource, a username, password, blockfactor, and result. Other attributes are specific to cfstoredproc, such as procedure for the procedure name.

Along with the cfstoredproc, there are two subtags that can be used. Cfprocparam is used to send information to the stored procedure, and full documentation can be found at http://livedocs. macromedia.com/coldfusion/7/html-
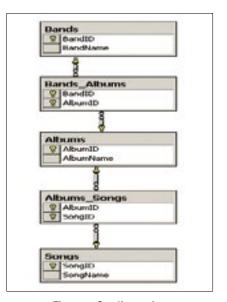
Figure 2: Creating a view

docs/00000313.htm#1102102. Cfprocresult is used to return information from the procedure. Full documentation for cfprocresult can be found at http://lived-ocs.macromedia.com/coldfusion/7/html-docs/00000314.htm#1102246. The code is in JH Code Segment 5.

For cfprocparam, the two parameters needed for this example are the value and the cfsqltype. The cfprocresult tag uses two parameters: name and resultset. The name is the variable that will contain the resultset after the stored procedure has executed. This is similar to the name attribute of the cfquery tag. The resultset should contain a numerical value. If the value is one, then the value will be given the first recordset returned from the stored procedure. If the value is two, then the variable is given the second record set.

Where To Go from Here

This article is intended to give you a peek into the power of using the database to do your dirty work instead of ColdFusion. There's plenty more to learn, so what should you look into next? You can start by investigating aggregate functions, which can be used to do calculations inside of a query. I talked a little about aggregate functions in the January '05 issue of this column. More info can be found at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_fa-fz_9yuk.asp. You might also look up user-defined functions in SQL Server. They're similar in concept to UDF in ColdFusion just implemented a bit differently. More info can be found at http://msdn.microsoft.com/msdnmag/issues/03/11/Data-Points/.

You might also look up how to create views and stored procedures using Transact SQL. With one of my current projects, we decided to create SQL files that contain the T-SQL to create the views and stored procedures. By storing these files in the repository, we're able to track database changes and even add some version control to the code.

About the Author

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

jeff@instantcoldfusion.com

JH Code Segment 1

```
<cfquery name="getSongs" datasource="MyDSN">
 select Bands.BandID, Bands.BandName, Songs.SongID, Songs.SongName
 from songs join albums_songs on (songs.songID = albums_songs.songID)
  join albums on (albums.albumID = albums_songs.AlbumID )
  join bands_albums on (bands_albums.albumID = Albums.albumID)
  join bands on (bands.bandID = bands_albums.bandID)
 where bandID = #variables.bandID#
</cfquery>
```

JH Code Segment 2

```
<cfquery name="getSongs" datasource="MyDSN">
 select BandID, BandName, SongID, SongName
 from SongsView
 where SongsView.bandID = #variables.bandID#
</cfquery>
```

JH Code Segment 3

```
CREATE PROCEDURE GetBandInfo

@BandID int

AS


select * from bands
where bands.bandID = @BandID
```

```
select albums.*
from albums, bands_albums
where bands_albums.albumID = albums.albumID and bands_albums.bandID =
@BandID

select albums.albumID, songs.*
from albums, bands_albums, albums_songs, songs
where bands_albums.albumID = albums.albumID and
 bands_albums.bandID = @BandID and
 albums_songs.albumID = albums.albumID and
 songs.songID = albums_songs.songID
GO
```

?? JH Code Segment 4

```
<cfquery name="getSongs" datasource="MyDSN">
exec  GetBandInfo #variables.bandID#
</cfquery>
```

?? JH Code Segment 5

```
<cfstoredproc procedure="GetBandInfo" datasource="MyDSN">
 <cfprocparam value="#variables.bandID#" cfsqltype="cf_sql_integer">
 <cfprocresult name="getBand" resultSet="1">
 <cfprocresult name="getAlbum" resultSet="2">
 <cfprocresult name="getSong" resultSet="3">
</cfstoredproc>
```

Building an IM Bot

It's easier than you think using ColdFusion

By Ben Forta
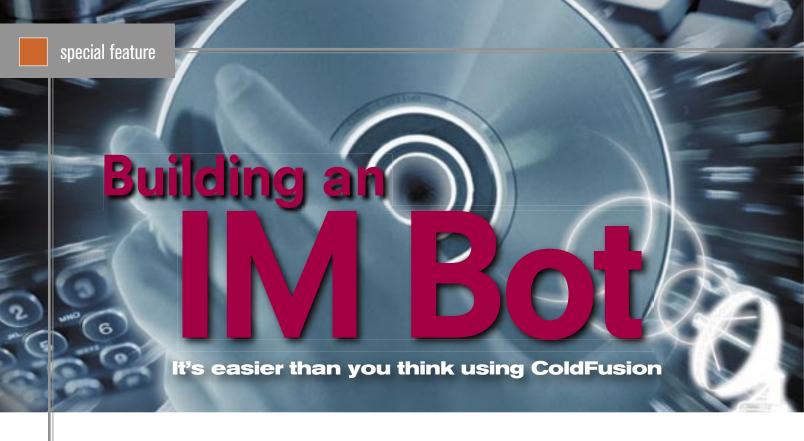
I recently brought a Google Talk bot that I put online at cfdocs@gmail.com. Google Talk users can add this user to their buddy list and then submit CFML tag and function lookups to it. (I've also brought Yahoo IM and AIM versions online as nickname cflivedocs, but more on those shortly). In this column I'll explain exactly what the IM bot is and what it does, and show you how to easily create a bot of your own.

The Big Idea

We've all used instant messaging (or IM), almost always for person-to-person communication; send a user a message and it'll pop up on their desktop allowing them to respond. An IM bot is a virtual IM user (*bot* is short for *robot*), an application that can receive and send messages programmatically. Bots are not a new idea, IRC bots have been around for years, in pre-web days we used to use e-mail bots to submit gopher and usenet searches via e-mail, and if you think about it, spiders (used by search engine to index web site content) are bots of sorts in that they programmatically simulate what it is that users would do with their web browsers.

So what would an IM bot be used for? Imagine that UPS or FedEx had bots online (named UPS or FedEx) that would allow you to simply type a tracking number in an instant message, instantly receiving a tracking response. Or an airline allowing you to type in a flight number to obtain gate information or flight status. Or your company employee directory allowing users to type in partial names to do lookups (the results of which could include an e-mail address which could be clicked on to send an e-mail). Or my ColdFusion LiveDocs example, mentioned previously. Of course, not all applications are suited to this type of interaction. But many are, and IM bots allow you to provide instant access to applications leveraging the immediacy and simplicity of now ubiquitous IM clients.

ColdFusion Event Gateways To The Rescue

Developing an Google Talk IM bot obviously requires that you have an application waiting to receive requests (and thus running at all times) that is logged in to the Google Talk network (and thus requiring a thorough understanding of the technical implementations used by Google Talk). Or you can simple use ColdFusion, which obviously will be always running (so as to respond to HTTP requests), and which already comes with the code needed to login and interact with Google Talk. And this requires a word or two (or more) about ColdFusion event gateways.

ColdFusion MX 7 Enterprise (and Developer's Edition) includes support for event gateways. The best way to understand event gateways is to understand what it is they allow.

ColdFusion has long been an incredibly simple way to build Web apps. That's why we all use ColdFusion, simplicity (and thus productivity). But ColdFusion is not exclusively tied to the Web. In fact, ColdFusion does not even talk to Web browsers (that is the Web server's job). ColdFusion simply executes scripts on the server in response to requests, requests which (thus far) have been typically HTTP originated.

So could ColdFusion respond to other requests? For example, data sent to a specific port, or changes in a folder, or inbound SMS and IM messages, or database table changes, or … ? The answer is yes, ColdFusion can respond to any and all of those, it just needs a way to know when those events occur.

Understanding ColdFusion Event Gateways

And that is what the event gateways are all about. Gateways are interfaces to other systems, ways for events to trigger ColdFusion processing. A gateway watching a folder on a server can trigger ColdFusion execution when folder contents change. A gateway connecting to an SMS provider can respond to inbound SMS messages (and send SMS messages as well). A gateway can be pinged by a database trigger so that a database event forces ColdFusion processing (imagine being able to automatically dynamically generate static HTML pages whenever back-end databases change). And a gateway connected to Google Talk can respond to (and send) Google Talk instant messages.

ColdFusion has a built in event gateway service (which can be stopped and started using the ColdFusion Administrator, and which is running by default). The event gateway service receives all gateway requests, queuing and routing them to the appropriate gateways as needed, and then executing them using a designated number of allowed threads (10 by default).

ColdFusion ships with an array of gateway types out of the box (additional ones are available from 3rd parties, and you may write these yourself too). Gateway types written in Java, and are registered in the ColdFusion Administrator. Gateway types are just that, types, they are not gateway instances. The folder watcher type has no code specific to your folders or what you'd want when an event is triggered. An SMS gateway type knows how to connect to SMS servers and exchange messages, but is not tied to a specific account or processing. Gateway types are generalized, they are the processing that is not unique to your application. You can kind of think of gateway types as being a bit like database drivers, they contain (and encapsulate) the code and processing needed to interact with some system or technology. And, as already stated, they must be registered in the ColdFusion Administrator before they can be used.

Gateway instances are also defined in the ColdFusion Administrator. A gateway instance is simply an entry that defines the name, the associated type, the name of an optional configuration file, and the name of the CFC file containing your processing. So, if you needed a gateway to respond to requests coming in over a socket, you'd create a gateway instance using the appropriate gateway type and specifying a CFC file that would actually receive those requests so as to be able to respond to them. A single gateway type can be used in as many gateways as needed (so you can have several different IM applications running, or multiple folders being watched each triggering different events). The CFC used is no different to any other CFC, except that specific method names are used (these are specified by the gateway type being used, each gateway type will expect specific CFC methods to be present). The CFC is straight CFML, and can contain any CFML code and processing. When a gateway event occurs (an SMS message arrives, a database trigger fires, data is received by a socket being watched, and so on), the ColdFusion gateway engine calls the appropriate CFC method, telling it everything it needs to know about the event. And your CFC code can then respond, using the same CFML that you know and love.

What You Need

Which brings us back to Google Talk IM bots. As should be clear by now, creating a Google Talk IM bot requires the following three items:

- An installed ColdFusion gateway type that is compatible with the Google Talk network. As Google Talk is built on top of Jabber which is XMPP based, the XMPP gateway that comes with ColdFusion fits the bill.
- A configuration file specifying the Google Talk account name and password to be used by the bot.
- A CFC file to actually process and respond to inbound IM messages.

Obviously, any IM bot needs an IM user account, so the first thing you'll need to do is obtain a Google Talk account. Google Talk uses GMail addresses and passwords, so obtain a new GMail account if needed, or use any unused account (don't use an account that you yourself will be using with Google Talk, the same account cannot be logged in twice of course, so the bot needs its own account).

Creating The Config File

Next you'll need to save that account information into a configuration file. Gateway config files are typically stored in **/gateway/config** under the ColdFusion root (**c:\cfusionmx7\ gateway\config** on a Windows standalone installation, **c:\jrun4\ servers\cfusion\cfusion-ear\cfusion-war\web-inf\cfusion\gateway\config** on the default ColdFusion instance on a Windows multi-server installation), although they could be stored elsewhere too. That folder may contain a sample XMPP configuration file named xmpp.cfg which you may copy and modify. Your configuration file should contain (at a minimum) the following entries (obviously setting userid and password to your GMail account information):

```
userid=account@gmail.com
password=password
resourceName=ColdFusion MX 7
secureprotocol=TSL
securerequirement=true
serverip=talk.google.com
serverport=5222
```

Creating The Bot CFC

Next you need the CFC that will respond to inbound requests. At a minimum you need two methods in your CFC, **onIncomingMessage** and **onAddBuddyRequest**.

"ColdFusion has long been an incredibly simple way to build Web apps"

All instant messages sent to your bot are routed to a method named **onIncomingMessage**. This method is the one that actually receives all inbound messages, and this is where you'd process user requests, generating responses if needed.

Here is a simple example:

```
<cffunction name="onIncomingMessage">
  <cfargument name="CFEvent"
            type="struct"
            required="yes">

  <!--- Get the message --->
  <cfset var message=CFEvent.data.message>
  <cfset var originatorID=CFEvent.originatorID>

  <!--- Result structure --->
  <cfset var retValue=structNew()>
  <cfset retValue.BuddyID= originatorID >
  <cfset retValue.Message="You said: #message#">

  <!--- send the return message back --->
  <cfreturn retValue>
</cffunction>
```

All methods invoked by gateways receive a single argument, a **CFEvent** structure (the contents of which vary based on the gateway type). In this example, two local variables are created containing values extracted from the CFEvent structure, **message** is the actual text sent by the user, and **originatorID** is the sender name.

Typically your bot would do some processing based on the contents of **message**, but this simple example just echoes the content back to the sender. A return structure is created, and two values are set, **BuddyID** is set to **originatorID** (so that the message is sent back to the sender), and **Message** contains a string echoing the received message.

And that is all that is needed. The return structure is returned by the <cfreturn> tag, and the echo will be sent back to the sender.

That's **onIncomingMessage**. Next comes **onAddBuddyRequest**. When you add a user to your own IM buddy list, the network sends a message to that user asking them to accept or deny the request. If accepted you'll be able to see when that user is online and will be able to send him or her messages, and if denied then you'll not be able to see online status, and you may not even be able to send messages. When a user adds your bot as a buddy, the server will ask it to accept or decline the request. Obviously, these requests must be handled programmatically (as there is no actual user), and so these are routed to a CFC method named **onAddBuddyRequest**. This method can always return an **accept** response, or can conditionally **accept** or **decline** based on some code (maybe a database lookup, or some password or code provided, and so on).

Here is an example:

```
<cffunction name="onAddBuddyRequest">
  <cfargument name="CFEvent"
            type="struct"
            required="true">
  <cfset var retValue = structNew()>

  <cfset retValue.Command="accept">
  <cfset retValue.BuddyID=CFEvent.data.sender>
  <cfset retValue.Reason="Welcome!">

  <cfreturn retValue>
</cffunction>
```

Once again, **onAddBuddyRequest** receives a **CFEvent** structure. This method must respond to the network either accepting or denying the request. Here a return structure is created. **Command** is set to **accept**, accepting the request (set it to **decline** to deny the request). **BuddyID** is the name of the user being accepted, and **Reason** is a welcome message that may be sent to the user (depending on the network and IM client being used). And like before, the return structure is then returned using the <cfreturn> tag.

Other methods are supported too. And best practices dictate that every method be present in your CFC, even if they are not all used. So every IM bot CFC you create should probably contain the following (even if the methods are all empty):

```
<cffunction name="onAddBuddyResponse">
  <cfargument name="CFEvent"
          type="struct"
          required="YES">
</cffunction>

<cffunction name="onBuddyStatus">
  <cfargument name="CFEvent"
          type="struct"
          required="YES">
</cffunction>

<cffunction name="onIMServerMessage">
  <cfargument name="CFEvent"
          type="struct"
          required="YES">
</cffunction>

<cffunction name="onAdminMessage">
  <cfargument name="CFEvent"
          type="struct"
          required="YES">
</cffunction>
```

The CFC can be saved anywhere on your server. The default path for gateway CFC files is **/gateway/cfc** under the ColdFusion root (**c:\cfusionmx7\gateway\cfc** on a Windows standalone installation, **c:\jrun4\servers\cfusion\cfu-sion-ear\cfusion-war\web-inf\cfusion\gateway\cfc** on the default ColdFusion instance on a Windows multi-server installation).

Defining The Gateway Instance

With the config file and CFC saved, you can now register the new gateway instance. Here are the steps needed:

1. Open the ColdFusion Administrator.
2. Go to the **Event Gateways, Settings** screen, and make sure that **Enable ColdFusion Event Gateway Services** checkbox is checked.
3. Next, go to the **Event Gateways, Gateway Instances** screen.
4. Use the form at the top of the screen to add your new gateway instance. Start by specifying a unique **Gateway ID**.
5. In the **Gateway Type** field select **XMPP – XMPP Gateway**.
6. In the **CFC Path** field specify the path to your gateway CFC file.
7. In the **Configuration File** field specify the path to your gateway configuration file.
8. To have your gateway start automatically upon ColdFusion server start, set **Startup Mode** to **Automatic**, otherwise set this field to **Manual**.
9. And finally, click the **Add Gateway Instance** button to save your new gateway instance.

To start the gateway, click the green **Start** button in the gateway **Action** column. The status will change to **Starting**, and then **Running**. And once running, your Google Talk IM bot will be ready to receive and respond to requests.

But what if the bot won't start? The **Debugging & Logging, Log Files** screen contains a file named **eventgateway.log** (this file will be created the first time event gateways are used). You can inspect this log file to determine what your gateway is doing and what errors it may have thrown.

And with that, your Google Talk IM bot can be online, and responding to requests.

What About Other IM Networks?

Thus far we have concentrated on Google Talk IM bots, but there are other IM networks out there. In fact, AOL IM (or AIM) and Yahoo IM (or YIM) are the most used public IM networks, with MSN and ICQ (now owned by AOL) being important, too. So what about those networks? What would it take to create an IM bot for those services?

ColdFusion ships with two IM gateway types, XMPP and Lotus Sametime. XMPP is the protocol used by Jabber, and thus by Google Talk (as explained previously), and by private Jabber servers. Lotus Sametime is IBM's real-time collaboration platform which includes instant messaging (and is generally used within organization for internal IM communication, as opposed to public IM communication). So, while ColdFusion can automatically communicate with Google Talk and other Jabber based services, it cannot talk to AIM, YIM, MSN, and ICQ without custom gateway types for these networks and their proprietary protocols.

So how could your bot connect to these services? You have several options available to you:

- The best option is JBuddy-CF, a set of gateway types created by Zion Software (http://www.zionsoftware.com/). JBuddy-CF is a commercial product which supports all of the major networks. It comes with an installation program which installs the required Java files into your ColdFusion

server, and even includes an optional .cfm file which uses the Administration API to register the new gateway types. Once installed, these can be used like any other gateway types. (And indeed, this is what I am using for the YIM and AIM interfaces of my CF LiveDocs gateway connected as user cflivedocs). This is not a free solution, but it is the simplest and cleanest option available thus far.

- The CFMX7 IM Event Gateways SourceForge project (http://sourceforge.net/projects/cfmximgateways/) is an ongoing initiative to develop IM gateway types for AIM, MSN, and YIM. This project is a work in progress, but if you don't mind getting involved with alpha code, this may be a viable option.
- You could install your own Jabber server (your own local IM server), and connect to it using the included XMPP gateway. Jabber add-ons can then be used to connect your IM server to the major IM networks (some servers and add-ons are freebies, others are commercial products). This configuration is not trivial, but it can be made to work very inexpensively.

Once installed and configured, the steps needed to build your bot are no different from those described in previously.

One important point to note is that different gateway types may require different CFC methods and may support different options and gateway helper functions (we'll get to those shortly). JBuddy-CF (built using the same underlying technology as the included XMPP gateway) provides the cleanest IM abstractions, using the exact same interfaces and helper functions for all networks. As such, if using JBuddy-CF, the identical CFC can be used for all networks (only the gateway types and config files will differ). This may, or may not, be the case when using the other two options described above.

Using Helper Functions

So now you know how to create bots and bring them online. But what if you wanted to manipulate bot settings, perhaps to set the status to **away** temporarily, or to programmatically obtain the number of messages responded to?

Developers creating the Java code for the underlying gateway type can include Java helper methods specific to that gateway type. Not all gateway types have helper methods, but many (including the XMPP gateway) do. To access the helper function you need to obtain a Java GetwayHelper object using a little known CFML function named **GetGatewayHelper()**. This function takes the id of a gateway as a parameter, and returns a Java object that can then be used within your CFML code.

To demonstrate this, the following code is used to set the IM status for my CFDocs Google Talk IM bot. To use this code with your own bot, simply change the **gateway_id**. To try out this code, just save it in a .cfm file (on the server running your bot) and execute it.

```
<!--- Gateway id --->
<cfset gateway_id="jabber cfdocs">

<!--- Status options
     Bit off a hack needed because the
     strings returned by getStatusAsString()
     are not the same as the ones passed to
     setStatus().
--->
<cfset status_options="ONLINE,AWAY,DND,NA,FREE_TO_CHAT">
<cfset status_display="ONLINE,AWAY,DO NOT DISTURB,NOT AVAILABLE,FREE TO
CHAT">

<!--- Get the GatewayHelper --->
<cfset helper=getGatewayHelper(gateway_id)>

<!--- If form post is present --->
<cfif IsDefined("FORM.status_new")>
  <cfset helper.setStatus(ListGetAt(status_options, status_new), "")>
</cfif>

<!--- Get current status --->
<cfset status=helper.getStatusAsString()>

<cfoutput>
<!--- Display current status --->
<h3>Status: #status#</h3>

<!--- Form to change status --->
<form action="#CGI.SCRIPT_NAME#" method="post">

<!--- Loop through status options --->
<cfloop index="o" from="1" to="#ListLen(status_options)#">
<input type="radio"
     name="status_new"
     value="#o#"
     <cfif ListGetAt(status_display, o) is status>checked</cfif>
     onChange="submit()">#ListGetAt(status_display, o)#<br>
</cfloop>

</form>

</cfoutput>
```

"Different gateway types may require different CFC methods and may support different options and gateway helper functions"

This is a pretty simple example. The gateway id is defined, as are lists containing the status values returned by and passed to the GatewayHelper methods. Next, **GetGatewayHelper()** is used to obtain a GatewayHelper object. The form in this little app is self-posting, and if a new status has been specified, **helper.setStatus()** is used to set the status (which will immediately be reflected in relevant buddy lists). The code then uses **helper.getStatusAsString()** to obtain the current status, and then displays a form used to change the status.

This is just a simple example, and there are lots of other **GatewayHelper** methods, too. Some are informational, like **numberOfMessagesReceived()** which returns the number of received inbound messages, and **isOnline()** which indicates whether or not the gateway is online and connected. Others are used to control buddy lists, including **getBuddyList()** which returns the buddy list, **getBuddyInfo()** which returns information about a buddy, and **addBuddy()** and **removeBuddy()** which do exactly what their names suggest. The full list of **GatewayHelper** methods for IM gateway types is included in the ColdFusion documentation (in the ColdFusion MX Event Gateway Reference section). Other gateway types may have other helper methods, consult the relevant gateway documentation to determine which methods are available to you.

Managing Session State

The last topic to discuss is session state management. Fortunately, there is not much to discuss. If ColdFusion session state management is enabled then you can use the SESSION scope in your bot code as you would in any other applications.

It is worth noting that although you may use SESSION just as you would in a web application, ColdFusion uses a different mechanism to identify sessions. In web applications ColdFusion uses a jsessionid or a cfid/cftoken pair (in cookies or URL parameters) to identify sessions, and these are passed back and forth with each request. There are no cookies or URL parameters to be passed back and forth when using IM, and so ColdFusion uses a combination of gateway id, gateway type, and the originator id (sender's buddy name) to identify sessions. And this is all handled internally by ColdFusion, you can just use sessions as you usually would, and it'll just work.

Summary

And there you have it, all you need to know to create your own IM bots, one example of the type of integration made possible but event gateways in ColdFusion MX 7 Enterprise. Have fun with this one, and if you do bring a public bot online, be sure to let me know about it.

About the Author

Ben Forta is Macromedia's Senior Technical Evangelist, and is the author of several bestselling books on ColdFusion and other topics. His web site is www.forta.com.
ben@forta.com

Polishing ColdFusion MX 7

By Dave Gruber

Merrimack (ColdFusion 7.0.1)

Since releasing ColdFusion MX 7, the Macromedia ColdFusion product team has been listening to feedback from ColdFusion MX 7 customers and using it to refine this important release, polishing what has already proven to be an incredible product. This month we'll release the ColdFusion MX 7 Updater, a free incremental release that updates platform support, rolls up all hot fixes and security fixes to date, updates embedded services, and adds incremental feature improvements.

It adds an important new platform for production use – Mac OS X. You can now download an easy-to-use turnkey installer built specifically for Mac OS X, making the installation experience fast and simple. It's been a long time in coming, and we're very excited to announce the Mac platform as a first-class production-use runtime for ColdFusion. Besides adding Mac OS X support with an install experience similar to Windows, we're also providing updated platform support for all of our major platforms including:

- Windows 2003 Service Pack 1
- - Red Hat Linux AS 4.0
- - SuSE Linux Enterprise Server 9
- - AIX 5L v5.3
- - IBM WebSphere Application Server (WAS) Network Deploy (ND)
- - Oracle Application Server 10g
- - Oracle 10g RDBMS (note that 10g Real Application Clusters (RAC) aren't supported in the Updater)
- - Some basic support for IPv6

Beyond updated platform support we've seen a number of opportunities to refine some of the new features in 7. We've added RTF output format support for cfreport besides the existing PDF, FlashPaper, and Excel formats.

This was actually something that we wanted to get into the original release but ran short of time. And we've made several refinements to the ColdFusion Report Builder like adding an "Element Properties" so you can view the specific details of an element, plus the ability to specify "Group By" order in the Query Builder, and we've added a "Band Properties" item to the right mouse button for bands and report components.

We've also refined cfdocument and cfdocumentsection by adding new "src" and "srcfile" attributes enabling the specification of either the url or path relative to the web root, or the absolute path of a file resident on the server. We also put in a "mimetype" attribute so you can specify the mime type of the source, including text/html (default), text/plain, text/xml, image/jpeg, image/png, or image/gif.

Also in the Updater are several refinements to Flash Forms, including the addition of Actionscript Event support (onfocus and onblur) in cfgrid and cftree, plus new support for the "onload" attribute, which means you can now run initialization ActionScript for the form. We've put in a new cftextarea attribute html="true|false," which lets you set the value as HTML or text when loading the form. Finally, the cfformitem tag type attribute now includes script as a valid value, which lets you create functions in Flash forms; this reduces the possibility of reaching the 32KB or 64KB limit errors.

These are just some of the feature refinements in this important, free Updater. And there are others, too. If you're already running ColdFusion MX 7, and want to update to 7.0.1, you can simply download the Updater and apply it to your server. If you haven't installed ColdFusion MX 7 yet, you can download full updated installers from our trial download site so you start with the best release of ColdFusion ever. Mac OS X kits are also available for download from the ColdFusion trial site.

ColdFusion MX 7 has proven to be an incredible product, and with this Updater to 7.0.1, it's even better! Many of the Updater refinements come directly from your feedback, so please keep sending in your comments and feedback. We're listening and ready to continue evolving ColdFusion, having already begun work on the next great major version of ColdFusion code named "Scorpio."

dgruber@macromedia.com

Implementing a Single Sign-On Solution Using ColdFusion

Constructing a single sign-on solution for Web applications

By Zachary Loch &
Steven Forehand

There is an evolution that takes place when organizations start to develop Web applications as part of their IT infrastructure. Initially, an application is written for a particular purpose, say a contact management system or an inventory control system.

These systems typically require a facility to log in with a username and password to control access. As time passes, several of these systems will be built across the organization, each in isolation of one another. The problem soon arises where users need to access several of the Web systems, all with different usernames and passwords. The next logical step is to create a Web portal to provide a single point of entry to the Web applications. The Web applications are then moved into the portal so that users can go to one place to access all of their systems. But what do you do with those applications that cannot be moved into your portal, or if you only have a handful of applications and are not ready for a portal? Are you doomed to leave your users stranded on isolated systems? In this article, we'll answer this question as well as explore East Carolina University's evolution, why some systems cannot be moved into a portal, how to provide a single sign-on solution for those systems, and speak more about the benefits of tackling the problem.

During the development and evolution of the East Carolina University's Onestop ColdFusion-based campus portal, we definitely encountered these issues. As addressed in our article the April 2005 edition of the *CFDJ*, one of our primary objects was to offer a software development environment managed by the "portal," meaning we wanted to free resources by allowing software development staff to focus on the task at hand rather than all the supporting software. We all know the amount of time spent by development staff creating a security structure, identifying the customer, creating repetitive page layouts, and display elements, etc. By using a development API, we were able to remove these time-consuming processes and allow software developers to simply focus on the software to be developed. If a process was not made available via the API, it was extensible enough to allow development staff to simply build upon the model and offer additional functionality. All was well in campus portal land.

However, while we all lived happily ever after for some time, two major events caused us to recognize and examine a major, missing element within the model. First, we were in the process of refining our portal software to become a marketable product (pssst… anyone need a portal?). Second, our campus had purchased a large, packaged software product from a well-known vendor to address the needs of the entire enterprise. Every pro-

cess was to be changed in nearly every area of campus business including Human Resources, Finance, and Student, and in all the custom-written legacy software as well as a large amount of the custom "Web" software.

While we knew ColdFusion and our campus portal would definitely be able to address the software development needs of the campus, how were we going to interface with the Web-based applications provided by the new enterprise system? Furthermore, if we are to market a portal product, shouldn't we offer a method of interface to Web-based systems external to our portal? Again, we knew we had a strong product for managing all these resources in a comprehensive, secure environment and would definitely be able to manage applications developed and written specifically for our portal, but how would we manage these external applications and systems? Could we just "pull" or consume these applications directly into our Web portal?

There are several reasons why you might not be able to pull all applications into your Web portal. One reason is that an application might be in a hosted environment. Or perhaps it is a system that was purchased and the code cannot be modified (as in our case), or the system is written in another language. No matter what the reason, it can be frustrating for users to enter separate usernames and passwords for each system in which they need to access. After all, isn't part of the purpose and power of a portal to offer improved and consolidated services to the customer? Of course, but how can we accomplish this goal? Let's look at how we can create a single sign-on solution so that users can stop writing down their passwords and putting them under their keyboards (IT security departments everywhere can breathe easier).

If we take a step back for a second and think about what needs to be accomplished, the problem is straightforward. We would like to present the user with a link from within one application, or portal, into another Web application so that when the link is clicked, a new window opens to the new application and the user is already logged in. If we think about this operation from the browser point of view, it's a simple process. The browser calls a page, which we'll refer to as login.cfm hereafter, and an HTML form is returned. The user then enters the username and password through the browser, hits submit, and the form is sent to the action page specified on the HTML form's action attribute. Usually some cookies are set (we'll discuss this later) and a page is returned to the user. To provide a single sign-on solution, we need to mimic the Web browser process above through code effective at creating a "soft browser." Next, we'll break down each part of the process and see how we can construct a full single sign-on solution with ColdFusion.

Before we dig into coding, some investigative work needs to be done. Let's grab our magnifying glass and don our Sherlock Holmes hat. We need to gather the following information: the URL of the login page, the HTML form element names for the username and password, the URL of the action attribute in the form of the login page, and the URL of the page that is called upon successful login. We can gather this information by having an actual user log into the system and by viewing the HTML source code (most browsers have this option) of each page. After

we have gathered the necessary information, we are ready to put our coding hat back on (you do have one…right?).

As we all know, ColdFusion provides some powerful tools to accommodate this process. We could start by sending a <CFHTTP> post request with the username and password to the action page, but most systems set the cookies for authentication up front (for example, CFID, CFTOKEN, JSESSIONID) when you visit the login page. Then, after successful logon, a session variable is set flagging the user as logged in. With that being the case, we need to follow the exact process of the browser. We need to issue a <CFHTTP> request to login.cfm.

```
<cfhttp method="get" url="http://testsso.com/login.cfm">
</cfhttp>
```

The server will then send back cookie information, which we need to store in an array or structure so that it can be sent for later HTTP (or HTTPS) requests. ColdFusion makes this extremely easy since the CFHTTP variable holds a nice data structure of all HTTP responses. You can access the structure with cookie information directly by performing the following: <cfset cookieStruct ="#cfhttp.responseheader['Set-Cookie']#">. The "Set-Cookie" structure creates the key names in numeric order (1,2,3) so we need to create a new structure with the key names equal to the cookie names and copy over the values as below:

```
<!--- pick out cookie information and store in temporary structure--->
<cfset tempCookieStruct ="#cfhttp.responseheader['Set-Cookie']#">

<!--- loop through cookie structure and reformat into new structure with cookie
name as key and cookie value as key value--->
<cfset cookieStruct=StructNew()>
<cfloop collection="#tempCookieStruct#" item="key">
    <cfset cookieKeyAndValue=ListGetAt(tempcookieStruct[key],1,';')>
    <cfset cookieKey=ListGetAt(cookieKeyAndValue,1,'=')>
    <cfset cookieValue=ListGetAt(cookieKeyAndValue,2,'=')>
    <cfset StructInsert(cookieStruct,cookieKey,cookieValue)>
</cfloop>
```

Now that we have created our session and have the necessary cookies in place, it's time to log the user in. Before we get to the code, let's discuss a few different ways to get the username and password. We could ask for all user accounts up front and enter them in a database manually, but if you have many users, that could take time and resources. One solution is when users log in and click the link to use single sign-on, we can check to see if they have a password in our database. If they don't, we make them key it in and store the credentials with the password encrypted. From then on, when users access single sign-on, we'll have their credentials locally stored in a database that we can pass on to the target system. The only drawback is that if the account expires, the user will have to reenter their credentials into our system. Depending on the target application, you might be able to automate some of that through code. Once we have obtained the credentials, we can log the user in by passing them

as <CFHTTPPARAM> attributes:

```
<!--- send the user credentials with previous cookies via HTTP post--->
<cfhttp method="post" url="http:// testsso.com/processlogin.cfm;jsessio
nid=#cookieStruct.JSESSIONID#">
    <cfloop collection="#cookieStruct#" item="key">
        <cfhttpparam type="cookie" name="#key#" value="#cookieStruct[
key]#">
    </cfloop>
    <cfhttpparam type="formfield" name="username" value="user1">
    <cfhttpparam type="formfield" name="password" value="mypwd">
</cfhttp>
```

It is important to remember to send the cookies during every http request since HTTP is stateless and that's how we are maintaining session. One thing you might have noticed – the URL has the following code attached to it: ';jsessionid=#cookieStruct.JSESSIONID#'. If your target application is running ColdFusion with J2EE session variables enabled or it's running any other J2EE server, you must pass the jsessionid as part of the URL. The fact that I'm accessing the jsessionid in the cookie structure by name means that I'm expecting it to be present. However, if you would like to write a single sign-on application that's generic enough to work with any system, you should first search for the jsessionid in the cookie structure. If it's there, append it to all HTTP URLs. One other gotcha: if you do append the jsessionid as part of the URL, IIS might think it's actually part of the URL and return a 404 Page Not Found error. If you have that problem, there is a ColdFusion Technote documenting the problem and resolution on Macromedia's Web site (www.macromedia.com/cfusion/knowledgebase/index.cfm?id=1c6b723). It's a simple fix and requires a small edit to the JRUN.INI file on the target system.

At this point the user is logged into the system and we have all the cookies stored in a structure. It's worth mentioning that we should check to see if any cookies are passed back from any of our HTTP requests and either add them to or update our cookie structure. My example assumes that no cookies are sent to the client during any other HTTP request other then the initial HTTP GET operation on the login.cfm page. From our earlier research we know what page the user is sent to after login. We can perform an HTTP GET on this page to verify the user is successfully logged into the system:

```
<!--- redirect user to main page of new application--->
 <cfhttp method="get" url="http://testsso.com/index.cfm;jsessionid=#coo
kieStruct.jsessionid#">
    <cfloop collection="#cookieStruct#" item="key">
        <cfhttpparam type="cookie" name="#key#" value="#cookieStruct[
key]#">
    </cfloop>
</cfhttp>
```

If this request doesn't work, the target system will most likely send us back to the main login page or an unsuccessful login page; however, if the page we requested is sent back to us, then we verified that the user is logged in. Usually we can check for some type of "signature" in the HTTP response to let us know which case we're dealing with. In my example I'm assuming everything is working fine (because it always does… right?).

Our end goal is to provide a new window to the user that displays the target application with the user already logged into the system. We currently have logged the user in and have a structure containing the authentication cookies. We now need to get the cookies into the browser and open a new window with the target application. There are two main paths we can travel to get this done and it all depends on our environment, namely, if both the target system and the system providing single sign-on access are both on the same domain. *If* both systems are on the same domain, we can use JavaScript to write cookies to the browser and open a new window. Most browsers maintain cookies when a new window is opened, which will in turn maintain our session. Since both of our systems are part of the same domain, the browser will continue to send the cookies to the target system after we make our hand off to the new browser window. We just need to make sure that the cookies are written before the window is opened. To do this we can use the following JavaScript:

```
<script language="javascript">
<cfloop collection="#cookieStruct#" item="key">
    document.cookie='#key#=#cookieStruct.jsessionid#;path=/';
</cfloop>
window.open('http://testsso.com/index.cfm;jsessionid=#cookieStruct.
jsessionid#','SelectWindow','width=800,height=300);
</script>
```

After the above code has been sent to the browser, our mission is done. The above code writes the necessary cookies to the browser and a new window will open that will effectively give up control to the target application.

Now, we must consider the case where our systems are *not* on the same domain. This presents a problem because cookies cannot be set and accessed across domains for security reasons. Imagine if you could read and write cookies from other domains (like online banking applications). Session cookies and security of those sessions would fall apart; the cookies would crumble (sorry, I had to). Therefore, we need to construct a way to write our cookies to the browser from the domain of the target application. We can do this by passing our cookies to an HTML file on the target server domain that will write the cookies to the browser for us and redirect the user to the target application. Here is an example of what the HTML file might look like (also in Listing 2):

```
<SCRIPT language="JavaScript1.1">
var queryString = document.location.search;

 // take out ? to get query string
queryString = queryString.substring( 1 );

// parse parameters based on &
var cookies= queryString.split( '&' );
for ( i=0; i<cookies.length; i++ ){
document.cookie=unescape(cookies[i]);
```

```
}

document.location.replace('http://testsso.com/index.cfm');
</SCRIPT>
```

The code above should be placed somewhere in the target application's domain. The code takes in query string parameters and writes them to the browser as cookies. It then opens a window to the target system. Since the cookies are in the browser, the session has been maintained and all is well. The only thing we need to do now is modify our original code to call this file passing the cookies instead of writing them to the browser. The format of the request is http://testsingle sign-on.com/setCookies.html?cookie=values;path=/&cookie=value;path=/. Therefore we need to take out the code that writes the cookies to the browser in our original example and add the following code:

```
<cfset cookieString="">
<cfloop collection="#cookieStruct#" item="key">
    <cfset cookieString=cookieString & "#key#%3d#cookieStruct[key]#;path%3d/&">
</cfloop>

<cfoutput>
<script language="javascript">
```

```
window.open('testsso.com/setCookies.html?#cookieString#','SelectWindow',
'width=800,height=300');
</script>
</cfoutput>
```

The first chunk of code formats the cookies so that they are ready to be passed via the URL. We need to strip out "=" and replace it with the HEX equivalent, namely, "%3d". The next chunk opens a new window that points to the HTML page that sets our cookies and we pass the cookies in the URL. The setCookies.html file then writes the cookies to the browser, and then redirects the user to the target application…game…set…match. The obvious drawback to this approach is that you must have permission or access to set the setCookies.html file on the target domain. That completes SSO 101. Next we'll look more at East Carolina University's specific implementation.

The possibilities of implementing an SSO solution are nearly endless. We have chosen to extend our implementation with the addition of a managed password "store" database. This database contains all the userID and password combinations for each user for each externally accessible system or application (don't worry, the data is encrypted). When a user logs into our portal, after all the authentication sequence is completed, the information related to that user in the password store database for SSO purposes is retrieved and stored in session. This method reduces the amount of database activity, provid-

ing better performance while also making this information available to internal functions of the portal as well as internal software applications.

In addition to a managed password store database, we have also developed standard XML configuration files for each system we access via SSO. Obviously, as this article states, there are bits of information you need from the external system to allow the SSO process to work. What are the names of the input fields for the userID and password? What is the URL of the external system? Our standard, proprietary XML configuration file defines all the elements needed for the external system to be accessed. The power of this feature is that it is extensible enough to offer access to multiple systems. Need access to another system? No problem. Simply create the XML configuration file. Much like the authentication information mentioned previously, all XML configuration files are read and stored in an application-scoped variable managed by the portal.

With the addition of the managed password store and the XML configuration files, we are dangerously close to a robust SSO solution. We are currently adding an administrative tool or "wizard" to step users through the process of creating the XML configuration file, allowing SSO access to external systems and applications. This "wizard" will not require any knowledge of XML, only that the user knows the URL and a few basic elements of the external system. Remember all that coffee and late nights sleeping on the floor at work making our office seem more like a hotel than a workplace (confused? see our previous article in *CFDJ*)? Here we go again…

As your organization progresses through the evolutionary life cycle of implementing Web-based applications, keep single sign-on and Web application portals in mind. You can save yourself much time and many headaches by starting your Web application portal while you still have only a few applications. Then, for those systems that cannot be moved into the portal, implement a single sign-on solution. Users can be freed from remembering multiple passwords for each system as well as the URLs. By combining a Web application portal with a single sign-on solution, any organization can provide a single point of access for all Web applications enterprise wide. This can increase security for the organization and provide convenience to users. After all, isn't convenience for our customers the ultimate goal? Enough said.

About the Author
Zachary Loch, a Macromedia Certified Advanced ColdFusion MX Developer, is project manager of application development at East Carolina University and also works on special data integration projects. He has 8 years of software development experience in a diverse set of industries including health care, insurance, education, and telecommunications.

Steven Forehand is the team manager for the New Technologies Development Group, a team of 12 talented application developers at East Carolina University located in Greenville, North Carolina. He has been using Macromedia ColdFusion since just prior to version 2 and has over 9 years of software development experience. Steven is Macromedia ColdFusion MX certified.

forehands@mail.ecu.edu

LISTING 1: sso.cfm

```
<!--- make http request to login.cfm--->
<cfhttp method="get" url="http://testsso.com/login.cfm">
</cfhttp>

<!-- pick out cookie information and store in temporary structure--->
<cfset tempCookieStruct ="#cfhttp.responseheader['Set-Cookie']#">

<!--- loop through cookie structure and reformat into new structure with cookie
name as key and cookie value as key value--->
<cfset cookieStruct=StructNew()>
<cfloop collection="#tempCookieStruct#" item="key">
    <cfset cookieKeyAndValue=ListGetAt(tempcookieStruct[key],1,';')>
    <cfset cookieKey=ListGetAt(cookieKeyAndValue,1,'=')>
    <cfset cookieValue=ListGetAt(cookieKeyAndValue,2,'=')>
    <cfset StructInsert(cookieStruct,cookieKey,cookieValue)>
</cfloop>
```
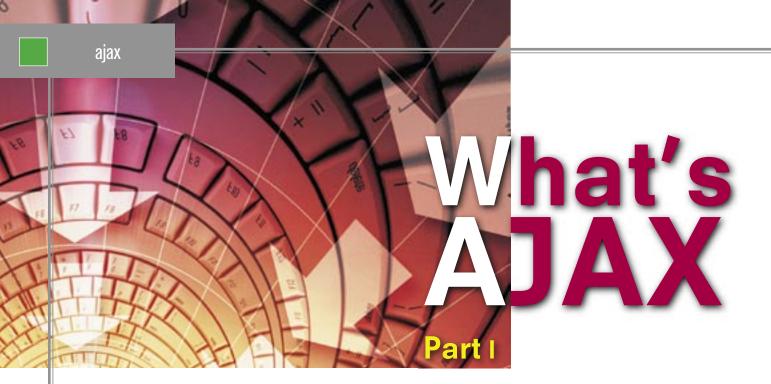
```
<!--- send the user credentials with previous cookies via HTTP post--->
<cfhttp method="post" url="http://testsso.com/processlogin.cfm;jsessioni
d=#cookieStruct.JSESSIONID#">
    <cfloop collection="#cookieStruct#" item="key">
        <cfhttpparam type="cookie" name="#key#" value="#cookieStruct[
key]#">
    </cfloop>
    <cfhttpparam type="formfield" name="username" value="user1">
    <cfhttpparam type="formfield" name="password" value="mypwd">
</cfhttp>

<!--- redirect user to main page of new application--->
 <cfhttp method="get" url="http://testsso.com/index.cfm;jsessionid=#cook
ieStruct.jsessionid#">
    <cfloop collection="#cookieStruct#" item="key">
        <cfhttpparam type="cookie" name="#key#" value="#cookieStruct[
key]#">
    </cfloop>
```

```
</cfhttp>

<cfoutput>
<script language="javascript">
<cfloop collection="#cookieStruct#" item="key">
    document.cookie='#key#=#cookieStruct.jsessionid#;path=/';
</cfloop>

window.open('http://testsso.com/index.cfm;jsessionid=#cookieStruct.
jsessionid#','SelectWindow','width=800,height=300,resizable=yes,status=
yes,menubar=yes,toolbar=yes,scrollbars=yes');
</script>
</cfoutput>
```

Listing 2: setCookies.html

```
<SCRIPT language="JavaScript1.1">
var queryString = document.location.search;

 // take out ? to get query string
queryString = queryString.substring( 1 );

// parse parameters based on &
var cookies= queryString.split( '&' );
for ( i=0; i<cookies.length; i++ ){
document.cookie=unescape(cookies[i]);
}

document.location.replace('http://150.216.184.31/lochz/portal/index.
cfm');
</SCRIPT>
```

Listing 3: modified sso.cfm

```
<cfset cookieString="">
<cfloop collection="#cookieStruct#" item="key">
    <cfset cookieString=cookieString & "#key#%3d#cookieStruct[key]#;pat
h%3d/&">
</cfloop>

<cfoutput>
<script language="javascript">
window.open('testsso.com/setCookies.html?#cookieString#','SelectWindow'
,'width=800,height=300');
</script>
</cfoutput>
```

What's AJAX

Part I

By Rob Gonda

So what is AJAX?

It's become very popular lately, even though it's not exactly new. It's been possible to use the concept behind AJAX since browsers introduced the XMLHttpRequest function in 1999.

AJAX isn't a technology, or a language, and there's no recipe to implement it; it's just a combination of various components to achieve something you otherwise couldn't: asynchronous http requests. However, since early 2005, when Google and Flickr popularized the concept, its use has grown rapidly.

The name AJAX is short for Asynchronous JavaScript and XML. It uses the JavaScript XMLHttpRequest function to create a tunnel from the client's browser to the server and transmit information back and forth without having to refresh the page. The data travels in XML format because it transmits complex data types over clear text.

AJAX uses XHTML for the data presentation of the view layer, DOM, short for Document Object Model, which dynamically manipulates the presentation, XML for data exchange, and XMLHttpRequest as the exchange engine that ties everything together.

Because of these requirements, AJAX works on I.E. 5.0+, Mozilla 1.0+, Firefox 1.0+, Netscape 7.0+, and Apple added it to Safari 1.2+.

Traditional HTML sends a request to the server, which processes it and either returns a static HTML page or dispatches the request to some scripting language such as ColdFusion, which creates and returns an HTML page for the browser to render. When this method has to retrieve new data from the server it has to repost and reload another HTML file. In many cases perhaps only a small portion of the returned HTML code varies and the shell itself remains the same resulting in huge overhead because the data has to be downloaded every time.

Some classic examples of applications that would benefit from AJAX are searching for information and displaying it back in a table, related select dropdowns, or checking if a user exists before submitting an entire form.

As we can see, AJAX offers many advantages over traditional HTML applications, but we shouldn't overestimate it. Because the data is JavaScript-driven, one of the main drawbacks is that search engines won't index any of the dynamically generated content. It's definitely not SEO-friendly.

People familiar with MVC will have a better grasp of the concept. Though details of MVC are outside of the scope of this article, the three defined components are Model, View, and Controller. The controller mediates between the data model and the view. It responds to events, which are usually actions by users, and changes the view or model as appropriate. The view consists of the HTML. JavaScript reacts to events triggered by the user and alters the existing rendered content with DOM. ColdFusion will be our model layer and can consist of one or more files.

Building an AJAX platform or engine from scratch can be a difficult and lengthy procedure. There are many AJAX engines available for download and you're welcome to use any of them. The only difference between implementations will be the data encoding, transmission, and decoding methods. The views and models of the MVC will be the same. My examples will be based on CFAJAX, a community-driven Open Source project. One of the problems with CFAJAX is its poor documentation. There is no manual or even a complete FAQ. So I will explain how to set it up step-by-step and work around its downside.

To use AJAX you'll need to really know JavaScript and DOM.

But by the end of this article you'll be able to set up AJAX, a ColdFusion model, make basic calls, and exchange simple data. As the applications grow and become more complex, the AJAX engine will remain the same, and the CF model will have more functions, but your JavaScript will have to manipulate more and more objects and that's where it's really at.

Enough chatter, let me show you how it works.

First, go to http://www.indiankey.com/cfajax/project.asp and download cfajax.1.2.zip. This file contains the core engine, some utilities, and some examples. Let's set up a folder in your Web root or whatever accessible folder you like called 'ajax.' Put the 'core' folder located in the cfajax zip file inside your 'ajax' folder. There are only two important files in this folder: 'engine.js' and 'cfajax.cfm.'

The engine.js contains the whole AJAX object that we'll use as our tunnel and cfajax.cfm has some basic functions that your ColdFusion model will have to include to decode the AJAX packet. Other files, not so core anymore, are 'util.js,' which contains a series of DOM functions to facilitate HTML manipulation when the server response is received, 'rico.js,' used for a built-in accordion example, and 'settings.js,' which aren't really settings, only the location of your ColdFusion model and an error-handler function. The only reason why this file exists is to hide the location of your ColdFusion file should someone open and view the source of your HTML file, which in my opinion is counter-productive because you have to edit this file and add a location variable every time you want to use AJAX with different models. The security should be built into the ColdFusion file.

Some people will claim that they can provide you with AJAX functionality without writing a single line of JavaScript; I disagree. Although you could use ColdFusion libraries to generate JavaScript, the results will be limited to the functionality provided. AJAX means asynchronous JavaScript and XML, not ColdFusion. So if you want to use AJAX effectively and provide solutions to new problems, you should learn JavaScript.

Now that we have our CFAJAX in place, let's create an index.cfm file and a model.cfm file in your ajax folder. For larger applications I would store the views in a 'views' folder and models in a 'models' folder, but I digress. Shall we concentrate on the basic example?

Open index.cfm and include the 'core/engine.js' file. Then create a 'getGreetings' and 'getGreetings_result' function. We'll call getGreetings onLoad for now, and the results function will just alert the response.

The 'engine.js' file creates an object called DWREngine; it means Direct Web Remoting. We will mainly be using a method called '_execute.' Execute takes four or more arguments. The first argument is the ColdFusion model location, then a queryString, a methodName, and a callback function, which is the JavaScript function that will be called when we get a response from the server. Execute also takes some optional arguments. It's not documented, but Execute will take any number of arguments, which will be parameters to be passed to the called method, and they have to be put between the third and fourth argument. I'll show you an example later.

Right now the file looks like this:

Our model.cfm will be extremely simple. All we need to do is include the 'core/cfajax.cfm' file and a getGreetings function:

```
model.cfm
<cfinclude template="core/cfajax.cfm">
<cffunction name="getGreetings" returntype="string">
    <cfreturn "Today is " & DateFormat(now(),'MMMM DD') & ", Greetings."
/>
</cffunction>
```

Well, that was easy; we just finished our first AJAX application.

Now let's create a small application that could actually benefit you. How many times have you created a registration form that can validate almost everything client-side by using JavaScript, but the username has to be validated upon submission, and if it already exists, you either have to abort and use JavaScript for the alert and send the client back, or cflocate and populate all the form fields by using the session. Wouldn't it be great if we could avoid all that and simple check the username as another JavaScript call? And not even think of creating an existing username array and have it local in the registration form?
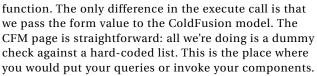
It 's not that much different from the file we have. Let's modify the index.cfm file a little. We'll create a basic user registration form with a single fieldname called user and we'll pass that value to the AJAX call. The tricky part here is that AJAX is asynchronous, which means that the function won't return the value you need. Instead the callback function will get called on a different thread. Not to worry, I'll demonstrate how this can be done.

```
model.cfm
<cfinclude template="core/cfajax.cfm">
<cffunction name="userExists" returntype="boolean">
    <cfargument name="event">
    <cfset var return = false />
    <cfif ListFindNoCase('rob,john,paul',event)><cfset return = true /></
cfif>
    <cfreturn return />
</cffunction>
```

Index.cfm now contains a user registration form. On submit, we'll call checkUser(). Note that we must actually call it with a return parameter or we'll always return false! No matter what happens, the form won't get submitted the traditional way. Note that I included 'utils.js' only to call the 'useLoadingMessage' function, which imitates the Gmail loading message while AJAX makes its round trip. 'Execute' sets a callback function, which checks for the ColdFusion return. If true, we'll display an error message, and if the user doesn't exist, we'll manually submit the form by using the JavaScript submit()

function. The only difference in the execute call is that we pass the form value to the ColdFusion model. The CFM page is straightforward: all we're doing is a dummy check against a hard-coded list. This is the place where you would put your queries or invoke your components.
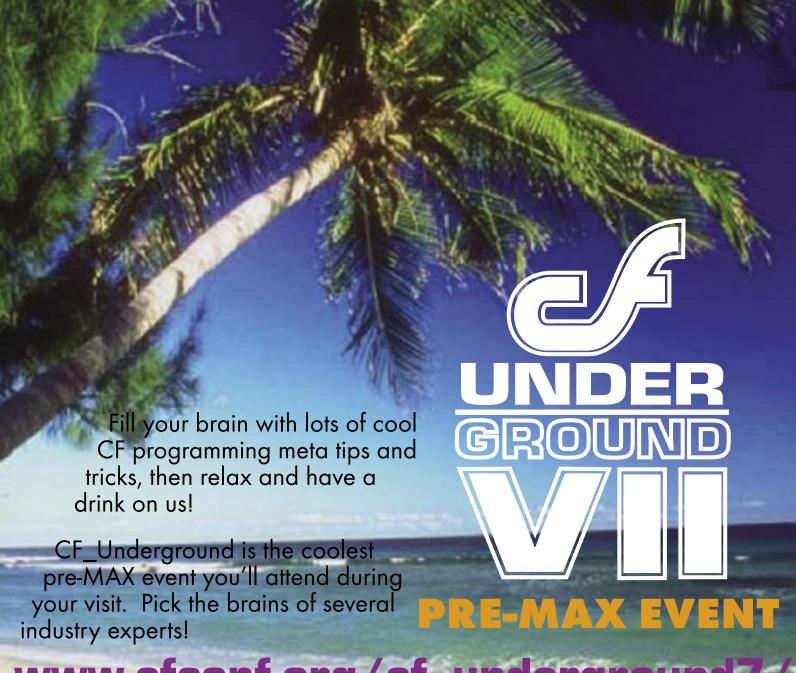
You should now be able to set up an AJAX engine, a simple HTML view, make a call, listen, process, and respond using ColdFusion, and process the response in your callback function. In the next part of this article, I'll explain how to pass complex objects by using WDDX serialization, populate tables, use innerHTML properties for advanced JavaScript view manipulations, and provide a full internal rotating banners application that tracks impressions and clicks. I will also cover cross-browser compatibility and offer some little tricks to make your AJAX application as widely compatible as possible.

rob@ichameleongroup.com

```
Index.cfm
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<HTML>
<head>
    <title>Greeting AJAX application</title>

    <script type='text/javascript'>_cfscriptLocation = "model.cfm";</script>
    <script type='text/javascript' src='core/engine.js'></script>

    <script type="text/javascript">

    function getGreetings()      {

    DWREngine._execute(_cfscriptLocation, null, 'getGreetings',  get-
Greetings_result);

    }


    function getGreetings_result (msg) {

    alert(msg);

    }

    </script>
</head>

<body onload="getGreetings();"></body>
</HTML>


Index.cfm
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<HTML>
<head>
    <title>User Check</title>

    <script type='text/javascript'>_cfscriptLocation = "model.cfm";</script>
    <script type='text/javascript' src='core/engine.js'></script>
    <script type='text/javascript' src='core/util.js'></script>

    <script type="text/javascript">

    function checkUser()          {
```

```
    DWRUtil.useLoadingMessage();

    var user = document.frm.user.value;

    DWREngine._execute(_cfscriptLocation, null, 'userExists', user,
checkUser_result);

    return false;

    }




    function checkUser_result (userExistsBool) {

    if (userExistsBool == 'true') {

    alert('Sorry, that user already exists');

    return false;

    }

    document.frm.submit();

    }

    </script>
</head>

<body>

<form name="frm" method="post" onsubmit="return checkUser();">
    <input type="Text" name="user">
    <input type="Submit" value="register">
</form>

</body>
</HTML>
```

Etudes: Arrays and Structures

Problem solving

By Hal Helms

In the first etude, we looked at arrays; in the second, we worked with structures. In this etude, let's work through some exercises that may use one or the other or both. Ready to dig in?

1. Your friendly, local veterinarian has asked you to come up with a way of storing information about the animals she treats. The only problem: no database. While you puzzle out a solution to this problem, look over the animals the vet has to deal with:

Animals
Dogs
Akbar, a 4-year old Rottweiler
Rex, a 3-year old Great Dane
Nyvek, a 6-year old Husky
Sage, a 1-year old German Shepherd

Cats
Pumpkin, a 2-year old Abyssinian
Amadeus, a 7-year old Nebelung
Joey, a 5-year old Chartreux
Princess, a 3-year old Savannah
Sunny, a 6-year old Oriental

2. "That's great," your vet tells you, "But what happens when I get new animals in to treat? Can you create some sort of online form for me to use?" Why, sure you can!

3. Professor Twitty needs to do some analysis on various temperatures he has been collecting. Here are the data:

Year: 2000
Month: January
Average temperatures:
   1: 27.2
   2: 28.4
   3: 24.6
   4: 25.2
   5: 26.1
   6: 26.0
   7: 28.1

Year: 2000
Month: February
Average temperatures:
   1: 22.5
   2: 24.7
   3: 25.3
   4: 24.7
   5: 26.3
   6: 27.3
   7: 28.2

Year: 2001
Month: January
Average temperatures:
   1: 26.4
   2: 28.9
   3: 30.0
   4: 31.5
   5: 28.4
   6: 27.7
   7: 30.1

Year: 2001
Month: February
Average temperatures:
   1: 29.1
   2: 29.3
   3: 30.7
   4: 32.3
   5: 31.4
   6: 30.7
   7: 30.9

"Gee, Professor," you say. "Where's the rest of the data?"

"Rest of the data?" the professor asks.

"Yeah, you know, the rest of the days in the month and the rest of the months in each of the years?"

"Goodness, me!" the professor exclaims. "A man's got to have some rest!"

"Oh, OK. What kind of analysis do you plan to do on this data?"

"Well, things like the average temperature for the month of January for both years, the average temperature for the 2nd of each month, finding out which year in which month was the warmest– that sort of thing," the professor answers.

You begin to realize you should have continued your

education so that you could get one of those cushy professorial jobs, but there it is: you're stuck being a programmer. Oh well, let's help the professor out.

4. One of your co-workers asks for your help: his son is in little league and he's volunteered to write a program to track the results of games over the season. Knowing how fanatical some parents get with regard to their children's sports activities, you're reluctant, but decide against your better judgment to help.

"What sort of information do you need to store?" you ask.

"I need to keep track of each game: who the opponent was, what the score was, whether the game was home or away, and who the winning pitcher was. I need to do this for the entire season."

Okay, you volunteered: what are you going to do?

5. Your boss comes by your desk. "Would you stop wasting company time writing little league software!? You've got real work to do. I need you to model an invoice."

"Sure, Boss. What do we need to keep track of?"

"Who the customer is, the invoice number, and the line items for each product. You do know about line items, I presume?"

"Err…sure. A line item consists of a… a…"

"A product SKU, the quantity purchased, a brief description of the item, and the price of the item."

"Got it, Boss. I'll have that in no time."

"I need to store these invoices by month, so figure that into your scheme."

Start figuring.

6. Sometime later, your boss returns. "The folks in marketing want to have an online store. So, we need a shopping cart. You know how to build one of those?"

"Of course," you lie. "Nothing to it."

The first thing you decide to do is create the variable to hold the information. What will it look like?

7. You've decided to build a "poker bot," an automated Texas Hold 'Em poker-player. You'll need to keep track of your two cards as well as the five cards that will make up the board. What will your data structures look like?

8. Your spouse has asked you to write a program to keep track of the unused AOL disks, still in their protective packaging. "Some day, these will be worth a fortune!" For each disk, you need to store the following information:
- AOL version
- Number of free minutes included
- User name
- Password

9. Your friends need help storing their "lucky" lottery numbers. Due to the long hours you've been working, your friends list has dwindled to:
Joe: The video store guy who always greets you – by the wrong name
Alicia: The "barista" at the local Starbucks coffee shop
Warren: The friendly telemarketer who you talk to each night in search of human contact outside of work

10. You decide to keep track of the number of meetings you have to attend each day for the next week. (You begin to wonder if ColdFusion's "numeric" data type will be large enough…) What does your variable look like?
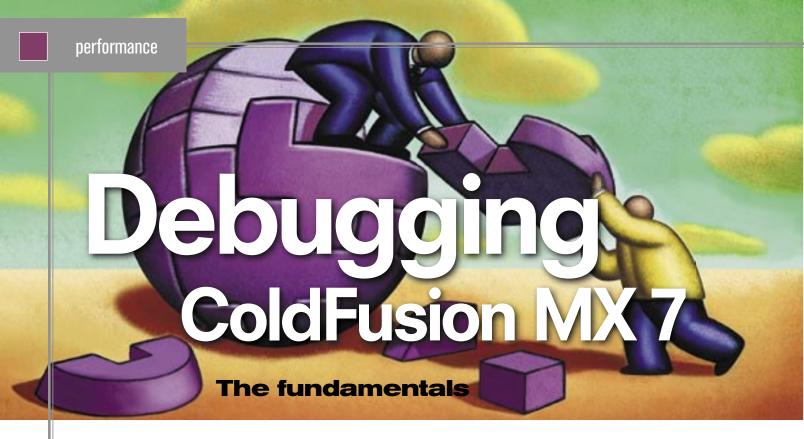
That's it for this month. My solutions are available at www.halhelms.com/etudes/arraysAndStructures.cfm. If you have an idea for an etude you'd like to see, please e-mail me at hal@halhelms.com.

About the Author
Hal Helms is the author of several books on programming. Hal teaches classes in Java, C#.NET, OO Programming with CFCs, Design Patterns in CFCs, ColdFusion Foundations, Mach-II, and Fusebox. He's the author of the popular Occasional Newsletter and his site is www.halhelms.com.

hal@halhelms.com

Debugging
ColdFusion MX 7

The fundamentals

By Jason Heaslet

Debugging. That necessary evil that none of us dare turn on in our production environments unless it is absolutely necessary. We all wonder what all that stuff does don't we? Well, let's investigate what some of it does, and what it means.

I have read through numerous scenarios, mechanisms, and tools to help in debugging my CF applications. What I have found is that a combination of some default debugging information, as well as some custom code, is generally the most appropriate solution. I certainly found the short section about the Administrator API on page 38 of the "Configuring and Administering ColdFusion MX" manual a very welcome read (yes, some people actually do read the manuals, it's sad really).

Several years ago Macromedia had an excellent class for "Performance Analysis and Tuning." If you are lucky, you can still find the course materials floating around the Net. I haven't seen it offered as a class in the past few years but I do know that they still provide it as a professional services engagement. If you are interested, I recommend calling MM and asking customer services about it. I am also under the impression that AboutWeb and UniversalMind both have a good foundation in "Performance Analysis and Tuning" engagements and teach some of the fundamentals.

Now, before we get started talking about debugging performance, let's discuss what the default settings can provide you.

Here are a few caveats about utilizing the default debugging information:

1. To utilize CF's default debugging information it must be enabled in the administrator by selecting the "Enable Debugging" checkbox.
2. When this setting is left unchecked, it will override the settings listed under the "Custom Debugging Output" heading.
   - Note: By enabling debugging, your server will incur a certain amount of performance degradation. In previous versions, this could be significant. However, in the latest version it seems to be nominal.
3. Sites utilizing CSS positioning, in general, will find that the debugging content often gets hidden behind the content, forcing the developer to comb through the source.
   - This can be resolved in your own custom debugging by utilizing JavaScript to find the bottom of your <div>s, then adjusting the div containing your debug information below that.
4. CFCs, data from Web services, complex objects, or structures are not, appropriately, represented in the default debugging settings.
5. Multiple instancing of your environment will certainly cause issues with your default debugging settings.
   - It is recommended that you supplement your default debugging with your own brand of custom debugging and error handling.

It should be noted that debugging and error handling are not exactly the same thing. However, they do generally run hand in hand. You'll likely use a number debugging techniques in your error handling. Our discussion today will focus on debugging and, more specifically, the tools used for performance debugging but not really error handling.

The type of information that a developer can get out of the default setting is certainly useful and significant. We can get our hands on the standard application, CGI, client, cookie, form, request, server, session, and URL structures as well as database activity. This is what most developers think of when referring to CF Debugging information. There are a few new features like

<CFTRACE> and <CFTIMER> that have been added to the arsenal for performance debugging. In addition, we can also find settings for Flash Form errors as well as my personal favorite, CFSTAT.

Debugging data, if enabled, will default to the localhost or 127.0.0.1. To see your debug information, your IP address must be listed within the Debugging IP Addresses section of the CF Administrator. Have you ever asked your systems administrator to add your IP address to the CF Administrator? You need to be able to see what variables the users are setting during their use of your application, right? Maybe get the timing of those long running queries? That button for "Add Current" IP can be a dangerous thing. I've seen many sites where some administrator or developer has simply clicked that button not realizing that they just added the proxy servers IP and are now broadcasting debugging information at the bottom of every page. Although, when you have debugging turned on for your application server, you should also be aware that whether you see the output or not you will incur the added performance hit. This may not affect your environment by adding 100 or so milliseconds per request, but in the grand scheme of things every little bit can count.

One of the most critical pieces that slow your applications is database connectivity and access. As a result, I think we should really start there with our discussion of specific features. Database Activity, once you have enabled this feature, will allow CF to output every query it performs, from the beginning of the request till the end of the request. It will also provide the data source it was performed against, the number of records it returned, the SQL statement, the template it was run in, the time it was run, and the time it took to process the request.

When I say CF will output every query it performs, I mean that in a literal sense. If you're the kind of developer who prefers to manipulate your code at a finer grain, to create your queries utilizing "QueryNew()" and its corresponding functions, you won't find those in the default debug information. You will, however, find that they will show up in CFSTAT, but we'll cover that later. This is one area where you will need to utilize <CFQUERY> to perform a query of queries to get your manually generated query into the debug output. It should be of additional note that the use of this technique is not really to get the timing of your manually generated query. It's really more for the benefit of having the query list so that you can remember to compensate for the timing in your design.

Here's an example:

```
<cfset qryFoo = QueryNew("col1,col2")>
<cfset tmp = QueryAddRow(qryFoo)>
<cfset tmp = QuerySetCell(qryFoo, "col1", "abc")>
<cfset tmp = QuerySetCell(qryFoo, "col2", "def")>

<cfdump label="qryFoo" var="#qryFoo#"><br />

<cfquery name="qryFoo2" dbtype="query">
    Select * from qryFoo
</cfquery>

<cfdump label="qryFoo2" var="#qryFoo2#">
```

So the actual output would look something like:

SQL Queries

qryFoo2 (Datasource=, Time=0ms, Records=1) in C:\Inetpub\wwwroot\foo.cfm @ 23:23:30.030
Select * from qryFoo

Scope Variables

CGI Variables:
AUTH_PASSWORD=
AUTH_TYPE=
AUTH_USER=
CERT_COOKIE=
CERT_FLAGS=
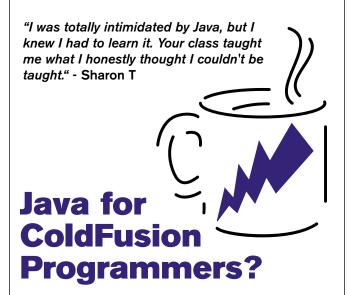CERT_ISSUER=
CERT_KEYSIZE=

Cookie Variables
CFADMIN_LASTPAGE=/CFIDE/administrator/debugging/index.cfm
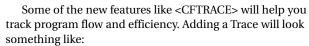JSESSIONID=8430c64ef280$C7$06$E
Debug Rendering Time: 78 ms

Note the "Debug Rendering Time" listed above. "Debug Rendering Time" is in addition to the processing time required to deliver the rest of the page.

Some of the new features like <CFTRACE> will help you track program flow and efficiency. Adding a Trace will look something like:

```
<cftrace abort="no" inline="yes" type="warning" var="FooTrace" text="This
is a trace">
```

CFTRACE 23:42:01.890] [15 ms] [C:\ Inetpub\wwwroot\foo.cfm @ line: 12] - This is a trace

Foo Trace (undefined)

And will output something like :
We can use <CFTRACE> in our code to get an idea of how our application is performing as we are utilizing it. Utilizing this method should step up your performance investigation and just like the use of getTickCount() should be removed from your final release. A partner to <CFTRACE>, and speaking of getTickCount(), is a new mechanism that performs the same functions that we used to use "getTickCount()" functions for, <CFTIMER>. It's a bit leaner than the use of getTickCount() and can be rendered in debug, outline, inline, and comment form. Here is an example of its use:

```
<cftimer Label="timerFoo" type="DEBUG">
    <cfquery name="qryFoo2" dbtype="query">
        Select * from qryFoo
    </cfquery>
</cftimer>
```

Here's an example of its output:

CFTimer Times

[31ms] *timerFoo*

My favorite, for performance debugging though, is a little used command-line tool called CFSTAT. It wasn't really advertised as part of the application server but has been shipped with it since at least version 4. It can be found in the \bin directory of your CFMX root. Its utilization is simple: CFSTAT <space> interval (in seconds).

CFSTAT 2 … the 2 referring to a 2 second interval

CFSTAT will deliver:
1. Pages per second – now and high
   - This is how many requests came in during the last interval.
2. Database requests per second – now and high
   - This will show how many query requests were performed by CF including any query manipulation you do on your own.
   - Note that this name is misleading and in reality reflects all query requests, locally built and externally requested.
3. Cache Pops per second – now and high

- "Cache Pops" refers to pulling data from cache.
- i.e., a <CFQUERY> that may be utilizing the "cachedwithin" attribute.
4. Requests queued
   - The number of requests still waiting to be processed.
5. Running requests
   - The number of requests that are currently being processed.
6. Requests timed out
   - The number of requests that exceeded your administrator time out settings.
7. Average queue time
   - The average amount of time a request was in the queue waiting to be processed (in milliseconds).
8. Average request time
   - The average amount of time it is currently taking to process requests (in milliseconds).
9. Average DB time
   - The average time required to process your query requests.
   - Note that this name is misleading and in reality reflects all query requests, local built, and externally requested.
10. Bytes in per second
    - The amount of data received by the CF Server (in bytes).
11. Bytes out per second
    - The amount of data transmitted back to the requesting browser (including whitespace) (also in bytes).

Just like any tool, the results need to be interpreted. There are numerous ways to read CFSTAT and my best recommendation would be to turn it on and watch it for a while. As you watch it, you'll start to see the patterns and can then investigate them. As you watch your output, you might start to notice something like DB/sec reaching a maximum number. That maximum number happens to coincide with the maximum number of simultaneous threads you have set in the CF Administrator. You might also find that the average DB time is higher than you expected and seems to grow with the Bytes Out/sec, indicating that your bandwidth is being affected by your output and is also possibly affecting access to your database. These are only examples of what you *could* find, not necessarily what you *will* find.

In this article I directed my focus on the performance aspects of the CFMX debugging features more so than the generic ones. However, if you have any questions about how to go about writing some custom debugging features of your own, or would like to share some of your how-tos, please feel free to e-mail me at Jason@Heaslet.net

---

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

Visit the *New*

www.SYS-CON.com

Website Today!

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

24/7

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

SYS-CON.TV
Watch video of breaking news, interviews with industry leaders, and how-to tutorials

BLOG-N-PLAY!
Read web logs from the movers and shakers or create your own blog to be read by millions

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

EDUCATION
The world's leading online i-Technology university

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

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

INTERNATIONAL SITES
Get all the news and information happening in other countries worldwide

JUMP TO THE LEADING i-TECHNOLOGY WEBSITES:

IT Solutions Guide

Information Storage+Security Journal

JDJ

Web Services Journal

.NET Developer's Journal

LinuxWorld Magazine

Linux Business News

Eclipse Developer's Journal

MX Developer's Journal

ColdFusion Developer's Journal

XML Journal

Wireless Business & Technology

Symbian Developer's Journal

WebSphere Journal

WLDJ

PowerBuilder Developer's Journal

ColdFusion U

U.S.

Alabama
Huntsville
Huntsville, AL CFUG
www.nacfug.com

Alaska
Anchorage
Alaska Macromedia User Group
www.akmmug.org

Arizona
Phoenix
www.azcfug.org

Arizona
Tucson
www.tucsoncfug.org

California
San Francisco
Bay Area CFUG
www.bacfug.net

California
Riverside
Inland Empire CFUG
www.sccfug.org

California
EL Segundo
Los Amgeles CFUG
www.sccfug.org

California
Irvine
Orange County CFUG
www.sccfug.org

California
Davis
Sacramento, CA CFUG
www.saccfug.org

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

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

California
Long Beach
Southern California CFUG
www.sccfug.org

Colorado
Denver
Denver CFUG
www.denvercfug.org/

Delaware
Kennett Square
Wilmington CFUG
www.bvcfug.org/

Delaware
Laurel
Delmarva CFUG
www.delmarva-cfug.org

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

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

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

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

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

Georgia
Atlanta
Atlanta, GA CFUG
www.acfug.org

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

Indiana
Avon
Indianapolis, IN CFUG
www.hoosierfusion.com

Indiana
Mishawaka
Northern Indiana CFUG
www.ninmug.org

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

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

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

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

Maryland
Rockville
Maryland CFUG
www.cfug-md.org

Massachusetts
Quincy
Boston, MA CFUG
www.bostoncfug.com

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

Minnesota
Brooklyn Park
Twin Cities CFUG
www.colderfusion.com

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

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

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

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

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

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

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

North Carolina
Raleigh
Raleigh, NC CFUG
www.ccfug.org
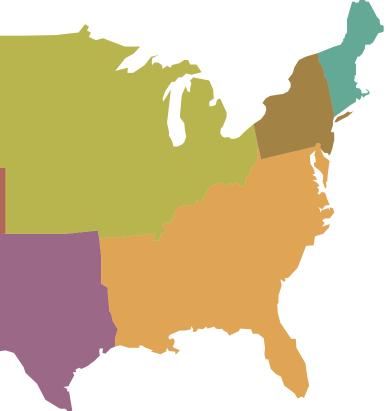
Ohio
Dayton
Greater Dayton CFUG
www.cfdayton.com

Oregon
Portland
Portland, OR CFUG
www.pdxcfug.org

User Groups

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

Pennsylvania
Carlisle
Central Penn CFUG
www.centralpenncfug.org

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

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

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

Tennessee
LaVergne
Nashville, TN CFUG
www.ncfug.com

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

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

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

Texas
Houston
Houston Area CFUG
www.houcfug.org

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

Washington
Seattle
Seattle CFUG
www.seattlecfug.com

About CFUGs

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

INTERNATIONAL

Australia
ACT CFUG
www.actcfug.com

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

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

Australia
Victoria CFUG
www.cfcentral.com.au

Australia
Western Australia CFUG
www.cfugwa.com/

Brazil
Brasilia CFUG
www.cfugdf.com.br

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

Brazil
Sao Paulo CFUG
www.cfugsp.com.br

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfugtoronto.org

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

Italy
Italy CFUG
www.cfmentor.com

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

Scotland
Scottish CFUG
www.scottishcfug.com

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

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

Spain
Spanish CFUG
www.cfugspain.org

Switzerland
Swiss CFUG
www.swisscfug.org

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

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org

Securing Web Services with SSL, IIS & Windows Authentication

The challenges

By Jeffrey Bouley

CF 7 has proven that it's an excellent way to rapidly create various Web Services at a given internal or external tier when working toward creating a Service Oriented Architecture (SOA).

Folks who have used CF 7 like this have experienced the efficiencies associated with exposing CFC methods as Web Services, allowing any number of applications and their related Web Service-supporting frameworks to call into methods declared "remote." This article isn't going to start with the basics associated with creating a Web Service with CF 7, but will address the possible challenges associated with securing them with SSL and Windows Authentication.

Once a Web Service has been created and tested one may want to apply a layer of security to it. Some options used to do this are encryption through SSL and integrating Windows Authentication with IIS to force a user or calling application to pass login credentials to a given NT domain.
So where to start?

Assuming the initial setup was executed on a development machine with no registered certificates, one must start by creating a certificate to implement SSL and then register the certificate in CF's java keystore. A keystore is a password-protected database that holds key and certificate information. Keystores are implemented in the Java runtime to validate certificates outside the Web server.

Before creating a certificate, we might suggest that a non-Microsoft-based certificate-generation tool be used since there's an issue with the Java implementation digesting and validating Microsoft-generated certificates for storage. This problem may have been resolved before this article was published, but be advised of possible problems with Microsoft-generated certificates (i.e., "Makecert") and CF 7 or J2EE application servers. I have used IBM's Keyman on several projects to create certificates. With a bit of practice, this tool will have you generating keys and certificates in no time. Keyman is free once you register and can be downloaded from http://www.alphaworks.ibm.com/tech/keyman.

To work with Keyman you must first generate a request in IIS that will be passed to Keyman. To do that, go to the properties of your Web site in IIS and select the directory security tab.

Figure 1
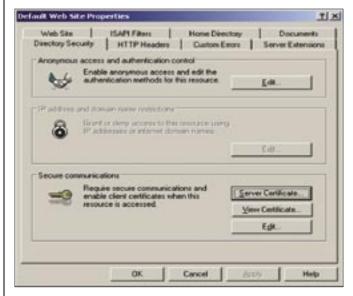
Now click on the server certificate button on the screen. Many prompts will follow; be sure to select "create a new certificate" and change such attributes as name if necessary. In the site name entry form make sure that the domain name is entered (i.e., www.yourdomain.com). This is an important step. If the domain name isn't entered correctly the certificate won't be valid after registering in the keystore and won't connect to the Web Service over https.

Don't forget to make a note of where the request text file was created; either copied in memory or stored on the hard drive. Next, open Keyman and select new token.

On development and test machines I use the default settings. Once the token has been created a key has to be generated. Go to "Actions" in the main menu and select generate key, accept the default selections. Now that the key has been generated, go to Actions again and select "create certificate." Select "self-signed certificate." This will create the private certificate needed to generate the certificate for the IIS request. Now go to "Actions" again and select "create certificate." Select "sign a PKCS#10 request" and browse to where the certificate request from IIS was saved or select load from the clipboard if it was stored there. Then save the generated certificate. The certificate has been created.

The certificate must now be imported into IIS. Open IIS and click on the server certificate again. Select "process the pending request and install the certificate." Browse to the certificate and submit. You should now be able to access the site over https. At this point it's imperative that you browse to the virtual directory where the Web Services reside in IIS and set it to require SSL. This is done by going to the properties of the directory through IIS and again selecting the directory security tab. Click edit in the secure communications area. The screen shown in Figure 3 will be presented. Pick the selections shown.

To turn on domain authentication go to the Directory Security tab in IIS and select the edit button. Select uncheck anonymous access and check basic authentication; enter the default domain and realm if necessary. I usually leave them blank and specify login parameters with CF 7.

Now that the directory has been secured with NT Authentication another layer of security is added to access the Web Services.

Note that though the warning from the basic authentication option says, "Password is sent in clear text," it won't be. It'll be

Figure 3
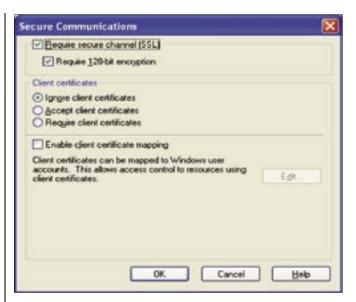
Figure 4

encrypted by enabling "Requiring secure channel" and "128-bit encryption" in the virtual directory holding the Web Services. There are only a few more steps to complete to secure our Web Service before it can be tested. At this point a certificate has been created for use with IIS and the virtual directory has been locked down with SSl; all that's left to do is register the generated certificate in CF 7's keystore to enable access to the Web Services.

The keytool command-line application that installs with CF 7 can be used to support this or the KeyTool GUI application. The KeyTool GUI application can save time in registering certificates since it's a well-thought-out interface. A good reference for the keytool command and registering certificates in a keystore can

Figure 2

Figure 5

Figure 6
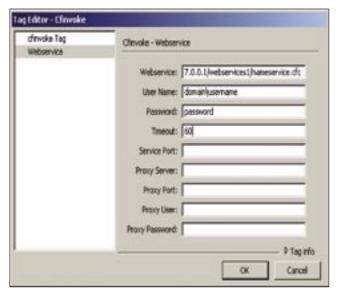
Figure 7

be found at http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html and http://www.macromedia.com/cfusion/knowledgebase/index.cfm?id=tn_19139.

I have a copy of KeyTool GUI 1.7, which is available under the GNU Public License and is free to distribute (http://www.strikefish.com/download/ktg-17-setup.zip). So I'll cover the GUI.

After installing the KeyTool GUI, run the program and open up the keystore installed with CF 7. If running in CF 7 in the J2EE configuration, the keystore is located at drive:\JRUN4\runtime\jre\lib\security\cacert and if standalone at drive:\CFusionMX7\runtime\jre\lib\security\cacert. Browse to the right location and select "All Files" in the "Files of Type" drop-down since the file doesn't have an extension. See Figure 5.

Once the cacerts file has been initialized KeyTool GUI will prompt for a password. The default password for the CF 7 install is **changeit**. Changing the password when you register the first certificate is a best practice and will add another layer of security when accessing the file separately from what the operating system provides. This can be done with the GUI by selecting the "Set KeyStore Password" option. Now that the keystore is opened, select the import trusted certificate option and browse to the certificate on the hard drive. A warning will pop up saying that a trust couldn't be established with the certificate. This is all right and is happening because the certificate isn't associated with a certificate authority. Again we note that the examples directly related to certificate generation are for development and test environments. Production servers should have certificates associated with a trusted certificate authority such as Verisign.

After registering the certificate into the keystore browse to the certificate in the list and view its properties by right-clicking and selecting "certificate details." Now save the change to cacerts and exit the KeyTool GUI. The Web Service is ready to be tested, but before you do don't forget to **RESTART THE CF 7 and JRUN SERVICES!!!!** I can't stress this enough. It's the single most common point of failure. The keystore is loaded by the JVM on startup and won't notice the additional certificate unless restarted.

A final step involves invoking the Web Service with CF 7 for a quick test. Figure 7 shows the tag editor used to enter the proper attributes to connect to the Web Service with the cfinvoke tag.
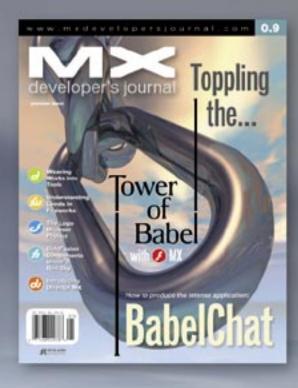
SSL and Windows Authentication when used together provide excellent security for your Web Services. Security has become paramount given the spiraling threat of hacking and online thievery. The techniques discussed in this article are provided straight out-of-the-box and can be configured and implemented in minutes. I wish all of you good luck in your quest to secure your servers and welcome any questions at my blog at http:www.strikefish.com or e-mail at jeffrey_bouley@strikefish.com.

About the Author

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

jeffrey_bouley@strikefish.com

CFMX Exam Buster 7

– No Number 2 Pencil Required

Macromedia has been offering ColdFusion Developer Certification since version 4.5 was released. Though it's arguable that there is no such thing as an exam that accurately measures a person's ability to develop applications, some employers do value such things and there's no denying that it certainly doesn't hurt to have the certification listed on one's resume. To help developers practice and prepare for the certification exam, CentraSoft offers their CFMX Exam Buster software, recently updated for the ColdFusion MX 7 Certification exam.
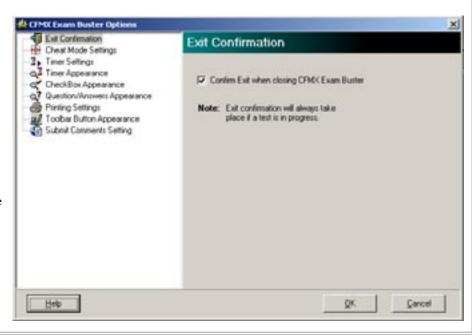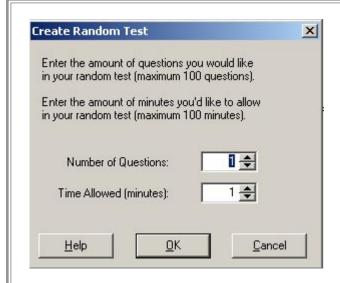
Before evaluating its value as a test preparation tool, I'll give you some information about the software itself. CFMX Exam Buster is standalone desk-top software for Windows – according to the website (http://www.centrasoft.com/) it supports versions 95,98, ME, NT 4, 2000, and XP (Home and Pro). I installed it on my laptop running Windows Enterprise Server 2003 with no problems. After downloading, installing, and registering the software you are ready to go. The software costs $39.95 and there's a demo version available, too. The demo is fully functional but only tests from a pool of 66 questions, whereas the full version draws from a pool of 875 questions.
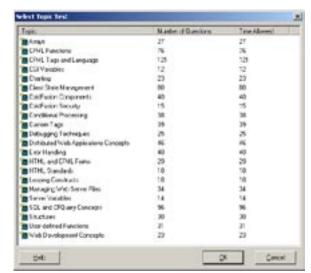
The software has several nice features that allow you to control the test taking features. It has a "no cheat mode" which prevents the test taker from being able to alt-tab to other applications while taking the exam. There are also options to change the look and feel of the check-boxes and the font, font color, and background color used for questions and answers.

The actual test-taking also has some options. One option is to take a random test. CFMX Exam Buster will prompt you for the test takers name, how many questions to ask, and how much time to allow for the exam. The
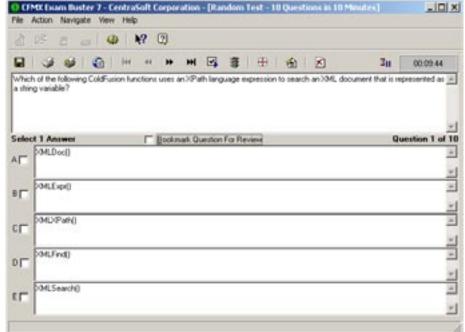
random test most closely simulates the real world exam,
but what if you really want to focus on specific areas of
expertise? Your other test taking format option is to do
what's called a topic test. CFMX Exam Buster will display
a list of topics you can choose from and then prompt you
for the test takers name before starting the topic test. The
number of topics I impressive and I'd imagine that any
develop who continuously took topic tests in their weak-
est areas would find it very useful, whether they're taking
the actual certification exam or not. During the exam the
test taker can skip forward and backwards in the ques-
tions one at a time or jump right to the start or end. They
can toggle cheat mode on and off, pause, print question

and/or answer sheets, etc., can comment on questions, and can also flag questions for review at the end of the exam.

Once a test is begun you can print the test (blank test and answer sheet) and tests may be saved or resumed if you want to take a break and visit the current exam later. When a test is complete, the report it generates is terrific – much nicer than when you complete the real-world certification exam. The software shows you a tabbed interface for viewing a pretty test summary report, a report showing all questions, the answers given, and the correct answers, and a report that just summarizes the questions answered incorrectly. One feature I found particularly clever is that every question in the summary reports not only shows your answer and the correct answer, but also displays a link to the Macromedia live docs about the function, tag, or feature that the question pertains to. This is very useful – especially if you miss a question and don't understand why.

Enough about the software and all of its great features – what about the questions themselves? I thought taking the exam with the same number of
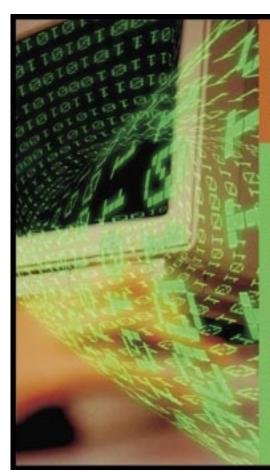
questions and time frame would be the best way to test this, especially since I have already taken the actual certification exam. The real world exam is 66 questions in 75 minutes (which I never need) so I set CFMX Exam Buster to give me the same amount of time to answer 66 questions and went at it. When I took the exam in the real world without preparing I scored 97%, finished with plenty of time, and felt that it was very easy – I probably missed 3% because of carelessness. I finished CFMX Exam Buster 7 with plenty of time but felt that it was much more difficult. I scored 93% - upon reviewing the questions I missed all but one were what I'd call a "trick" question – I missed them purely out of carelessness because I didn't read the question properly. One question I actually did get right – the exam is mistaken. That's OK – these things also happen on the real-world test. I found the software to be extremely useful and the test question much more challenging than the actual certification exam. The only thing I didn't like was the fact that if you select an answer and want to change it, you must first deselect the answer you've already chosen. That's not really a very big deal. I was pleasantly surprised by CentraSoft's CFMX Exam Buster 7 and I strongly recommend it above any other study aid to anyone preparing to take the certification exam.

About the Author

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

simon@horwith.com

The ColdFusion Blogosphere

ColdFusion MX 7.01 Just Released

By Stephen Erat
from http://www.talkingtree.com/blog/

In these last 10 minutes the CFMX Updates page on Macromedia.com was modified to include the latest revision of ColdFusion MX, 7.01. This free update for CFMX 7 provides support for Network Deploy on WebSphere, support for Oracle 10g, support for Red Hat Enterprise Linux 4 and SUSE 9 Enterprise Linux, a Mac OS X clickable and user friendly installation GUI for the familiar Server Configuration of ColdFusion MX, as well as a bundle of bug fixes.

Sandboxing with Multiple Instances

By Allana
from http://www.creative-restraint.co.uk/blog

A while back, John Beynon and I were having an issue with getting multiple instances to use their own JVM when enabling Sandboxing. This is when creating them using the new Enterprise Manager.

The problem we had was that for the new instances, we had to point the security manager JVM arguments to the default "cfusion" instance.

Any time we tried to have them point to the new instance, we got a security exception.

Well, as it turns out, it's a real doddle to fix, but very very easy to miss.

The default server instance "cfusion" lives under:
<jrun>/servers/cfusion/cfusion-ear/cfusion-war

When you create a new instance, it gets deployed to:
<jrun>/servers/<instance>/cfusion.ear/cfusion.war

Note the difference - a dot instead of a dash.

Now, I need to step back a bit and explain the issue.

When trying to run the new instance with it's own JVM, we took a copy of the original jvm.config file and simply replaced all "cfusion" entries for "newinstance".

What we didn't notice was that the dashes were also being replaced by dots ... hence the security warning.

So, when setting up your new instances to use their own JVM, remember to not only change the instance name but also that cfusion-ear needs to become cfusion.ear and cfusion-war needs to become cfusion.war

---

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
ACTIVEPDF	WWW.ACTIVEPDF.COM	866.468.6733	17
CFDYNAMICS	WWW.CFDYNAMICS.COM	866.233.9626	4
CFUNITED	WWW.CFCONF.ORG	301.424.3903	21
HAL HELMS, INC	WWW.HALHELMS.COM		35
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.4678	21
INTERAKT ONLINE	WWW.INTERAKTONLINE.COM	4031 401.68.19	3
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800.379.7729	52
MACROMEDIA	WWW.MACROMEDIA.COM	415.252.2000	11
MACROMEDIA	WWW.MACROMEDIA.COM	415.252.2000	51
PAPER THIN	WWW.PAPERTHIN.COM	800.940.3087	25
SAVVY SOFTWARE	WWW.BESAVVY.COM	866.870.6358	27
SEAPINE SOFTWARE	WWW.SEAPINE.COM/WEBDEV	888.683.6456	6

MAX is Angela Buraglia.

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

Learn

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

Connect

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

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

Register Now

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

Ideas in motion. The 2005 Macromedia® Conference.

Intermedia.NET...

Now serving the hottest ColdFusion.

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

Intermedia.NET's premier hosting services include:
- ColdFusion MX hosting
- Competitive plans
- Security sandboxes
- Custom tag registration
- Verity collections search engine
- Guaranteed service levels

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

Call us at: 1.888.379.7729
e-mail us at: sales@intermedia.NET
Visit us at: www.intermedia.NET

INTERMEDIA.NET