ColdFusion 6.0 **Preview!**

U.S. $8.99 (Canada $9.99)

COLDFUSION Developer's Journal

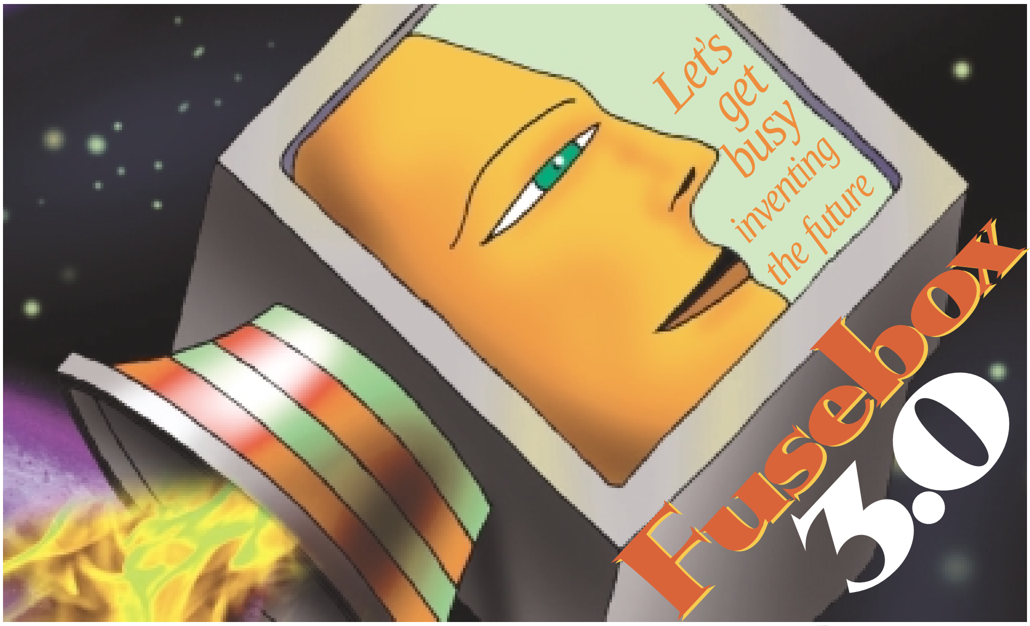
ColdFusionJournal.com

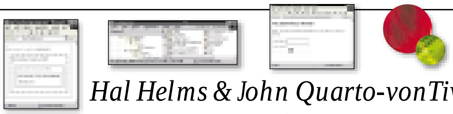
November 2001 Volume: 3 Issue: 11

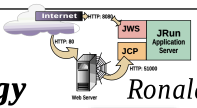
COLDFUSION *Fast Track*

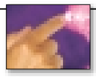
14

SYS-CON MEDIA







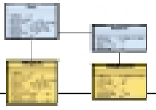


LISTING 1

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-
444553540000"codebase=http://download.macromedia.com
/pub/shockwave/cabs/flash/swflash.cab#version=5,0,
0,0 WIDTH=550 HEIGHT=400>
```

LISTING 1

```
import com.allaire.cfx.*;
public class HelloWorld1 implements CustomTag
{
   // custom tag entry point
   public void processRequest(Request request, Response response)
```



ABLECOMMERCE

www.ablecommerce.com

RACKSPACE

www.rackspace.com

COLDFUSION Developer's Journal

international advisory board

Jeremy Allaire, *CTO*, **macromedia, inc.**
Charles Arehart, *CTO*, **systemanage**
Michael Dinowitz, **house of fusion, fusion authority**
Steve Drucker, *CEO*, **fig leaf software**
Ben Forta, *products*, **macromedia**
Hal Helms, *training*, **team allaire**
Kevin Lynch, *president*, **macromedia products**
Karl Moss, *principal software developer*, **macromedia**
Ajit Sagar, *editor-in-chief*, **XML-Journal**
Michael Smith, *president*, **teratech**
Bruce Van Horn, *president*, **netsite dynamics, LLC**

department editors

editor-in-chief
Robert Diamond robert@sys-con.com
vice president, production
Jim Morgan jim@sys-con.com
executive editor
M'lou Pinkham mpinkham@sys-con.com
managing editor
Cheryl Van Sise cheryl@sys-con.com
editor
Nancy Valentine nancy@sys-con.com
associate editor
Jamie Matusow jamie@sys-con.com
associate editor
Gail Schultz gail@sys-con.com
associate editor
Brenda Greene brenda@sys-con.com
assistant editor
Lin Goetz lin@sys-con.com
product review editor
Tom Taulli
tips & techniques editor
Matt Newberry

writers in this issue

Mike Chambers, Robert Diamond, Ben Forta,
Hal Helms, Dave Horan, Guy Rish,
Michael Smith, Ed Swartz,
Ronald West, Bruce Van Horn

subscriptions:
For subscription requests please call
1 800 513-7111 or go to: **www.sys-con.com**
cover price $8.99/issue
domestic $89.99/yr. (12 issues)
canada/mexico $99.99/yr
overseas $129.99/yr
back issues $12 U.S. $15 all other

editorial offices: SYS-CON MEDIA, INC.
135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 **Fax:** 201 782-9600
COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
is published monthly (12 times a year)
for $89.99 by **SYS-CON Publications, Inc.,**
135 Chestnut Ridge Rd.,Montvale, NJ 07645

postmaster: send address changes to:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON MEDIA
135 Chestnut Ridge Rd.,Montvale, NJ 07645

For promotional reprints, contact reprint coordinator:
Carrie Gebert carrieg@sys-con.com

SYS-CON PUBLICATIONS, INC., reserves the right to revise, republish,
and authorize its readers to use the articles submitted for publication.

distribution in USA:
by **International Periodical Distributors**
674 Via De La Valle, Suite 204,
Solana Beach, CA 92075
Phone: 619 481-5928

'Playing' with ColdFusion

BY **ROBERT DIAMOND**

We've often provided coverage of the massive lists of "big name" companies that are using ColdFusion on their Web sites. I find this to be an excellent example of the health and growth of the ColdFusion industry. One new site alone of course isn't enough to propel ColdFusion forward, but every bit helps. It's certainly good news for all CF developers when a major site gives its stamp of approval to the language by relaunching or adding CF-based areas to its site.

The latest in a long line of such sites is FAO Schwartz. Mindseye redesigned its Web site using both ColdFusion and JRun to integrate extensive front- and back-end technologies and workflows to create a new and efficient content management system that links product creation, checkout, and inventory/fulfillment with buyers, catalogs, and image creation. The new site "streamlines FAO's buying, e-commerce, and inventory activities by reversing the traditional, organizational workflow. Product acquisition originates in the back-end Web application and is made available through the front-end Web site, catalog, and fulfillment systems." That's a quick and jumbled description, but again, this is more good news for the ColdFusion world.

For a complete, or as complete as you can get, list of sites on the Net running ColdFusion – hop on over to Ben Forta's extensive site at www.forta.com.

CFDJ November contains its usual line-up of stellar content. Dave Horan has reviewed the CommonSpot Content Server 2.5 from PaperThin. This latest release expands on its 2.0 addition by offering several new features that we look at here. If you're running a content-based site, it's definitely worth a look.

Part 2 of 2 on integrating ColdFusion and the XML-RPC comes from Ronald West. Part 2 demonstrates the connections between ColdFusion and the JRun servers, and will show you how to work with data in the XML-RPC. For those of you who think that Web services is just a lot of hoopla, here's a demonstration of some real-world possibilities.

Ed Swartz, a first-time writer for ***CFDJ***, has written about how to design a high-performance image display application with CF. Image rotation has lots of useful possibilities, and the code and examples here can fit into many projects. Kevin Schmidt looks behind the scenes at how to set up shopping carts, how they work, and the systems that run behind them. Mike Chambers of Macromedia contributes another article in our ongoing series covering the integration of ColdFusion and Flash.

Michael Smith has written up a review of ColdFusion Edge/JDJEdge/Web Services Edge for those of you who missed it in New York City this September. It's a good overview on what to expect from SYS-CON's ongoing series of developer conferences – conferences I hope to see you at in the future! Also Hal Helms looks at Fusebox 3, the latest version of the Fusebox specification; Ben Forta discusses Neo, the next major version of ColdFusion; and more!

ABOUT THE AUTHOR

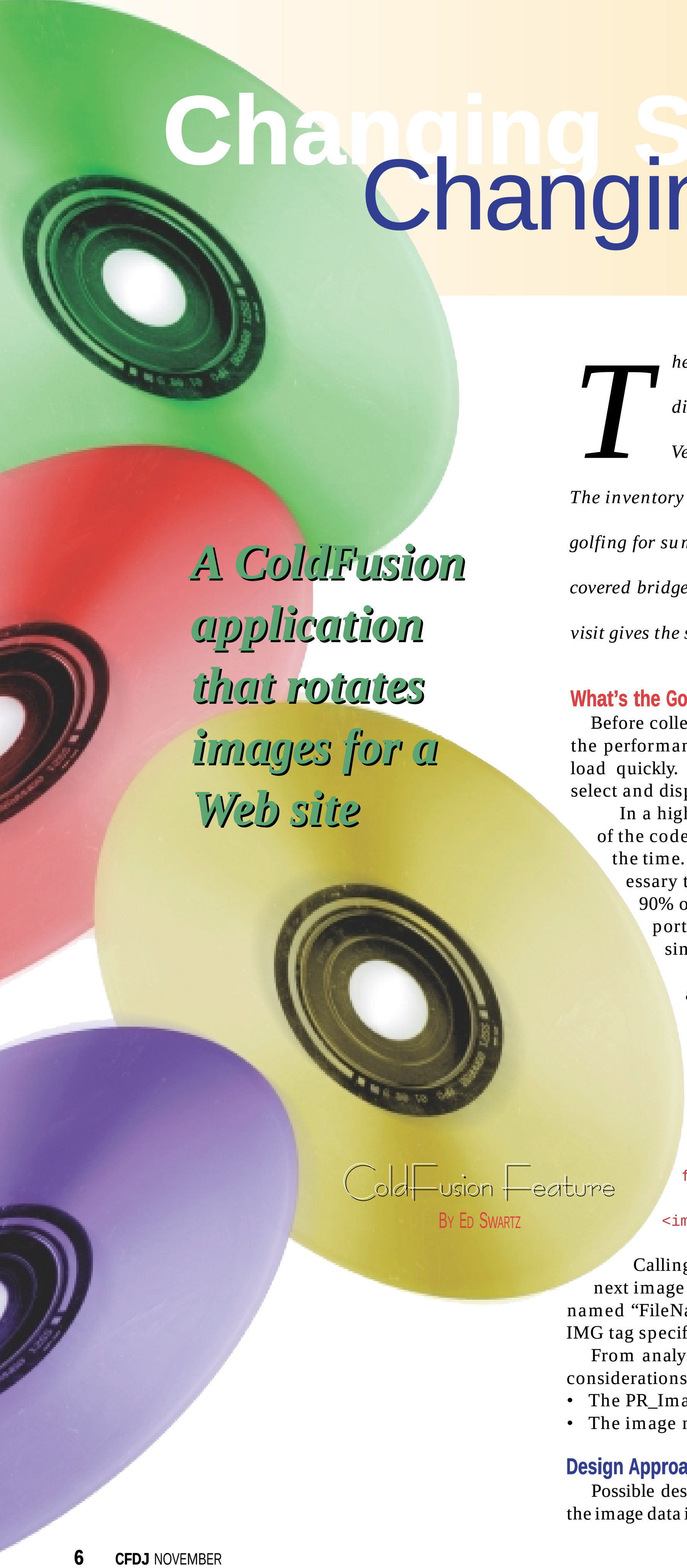
Robert Diamond is editor-in-chief of **ColdFusion Developer's Journal** *as well as* **Wireless Business & Technology**. *Named one of the "Top thirty magazine industry executives under the age of 30" in Folio magazine's November 2000 issue, Robert recently graduated from the School of Information Studies at Syracuse University with a BS in information management and technology.*

Robert Diamond

ROBERT@SYS-CON.COM

Changing Seasons,

Changing Seasons,



A ColdFusion application that rotates images for a Web site

ColdFusion Feature

By Ed Swartz

The assignment? Design a ColdFusion application to rotate the display of images for the top portion of the front page of the Vermont.com Web site. Vermont.com has an inventory of images. The inventory includes sets of images reflecting each season. Rafting and golfing for summer, mountain vistas of colorful trees for fall, and snow-covered bridges and houses for the winter. Changing images upon every visit gives the site a fresh, up-to-date quality, helping to retain visitors.

What's the Goal?

Before collecting possible design approaches, we need to understand the performance requirements. The site owner wants the front page to load quickly. Therefore, the image rotation application must rapidly select and display the next image in the rotation.

In a high-performance application the 90/10 rule states that 10% of the code will run 90% of the time, 90% of the code will run 10% of the time. When designing a high-performance application, it's necessary to look at that critical piece of the code: the 10% running 90% of the time. We're less concerned about the administrative portions of the application (the 90% running 10% of the time) since this part has little or no impact on performance.

What does the critical section of code look like and what are the ramifications of that code? The code needs to:

- Select the next image for display.
- Display the next image.

The code on the front page will appear something like:

```
<cfmodule
template=".\PR\PR_ImageGet.cfm
filename="FileName">

<img src="#FileName#>
```

Calling PR_ImageGet.cfm, a ColdFusion custom tag, selects the next image from the database and returns a file name in the variable named "FileName". The front page displays the image with an HTML IMG tag specifying the file name in the "src" attribute.

From analysis of the goal we now understand two important design considerations:

- The PR_ImageGet custom tag must execute rapidly.
- The image must be stored in a file for the HTML IMG tag to work.

Design Approaches

Possible designs include storing the image data in a database or storing the image data in files and then storing a reference to those files in a database.

Changing Images

Changing Images

When storing image data in a database the first issue that arises is: What data type do we use for the image field? Unfortunately, the data type selected varies depending on the database system utilized. For Microsoft Access the memo data type is a possible choice; for Microsoft SQL Server we can use the BLOB data type.

The problem with the Microsoft Access memo data type is that our image is in binary and the memo data type stores ASCII characters. We need some way to convert between the two formats.

The solution is to convert the binary image data into ASCII characters. Fortunately, ColdFusion provides the ToBase64() function that converts 8-bit data into 6-bit data. ToBase64() takes as a single argument a string or binary object and returns the data in base 64 format. Prior to storing an image, we need to call ToBase64() and then store the output of the function in the database. To display the image we'll need to get the data from the database and then call ToBinary() to convert the ASCII characters back into binary.

Alternately, we could convert the data to WDDX format by calling <CFWDDX> with the action attribute set to "CFML2WDDX". Upon reading the data from the database we would need to call <CFWDDX> again, but now the action attribute must be set to "WDDX2CFML" to restore the data back to its binary state.

The Microsoft SQL Server BLOB data type stores binary data. Our problem is that the HTML IMG SRC attribute requires a reference to a file and the image data is stored in a database. We need a way to get the image from the database to a file. Jukka Manner has provided the custom tags, CFX_GetImage and CFX_PutImage, available from the Macromedia Developers Exchange. Calling CFX_PutImage copies binary data from a file and stores it in a field with the BLOB data type. Conversely, calling CFX_GetImage retrieves the binary data from the database and stores it in a file.

Now we know the data types and the issues involved in storing and retrieving binary data. Next we need to outline the steps involved in displaying an image. (I'm not going to cover steps involved in storing the image since that happens infrequently.)

For Microsoft Access, with the data stored in a field of memo data type, the steps are:

- Perform a query to locate the record containing the image to be displayed.
- Call ToBinary() or <CFWDDX ACTION= "WDDX2CFML"> to convert the ASCII character string into binary.
- Store the binary data in a temporary file.
- Compose a URL that points to the image file.
- Provide the URL to the IMG SRC attribute.

Once the image is displayed the application will need to remove the temporary file.

For Microsoft SQL Server the steps are similar:

- Call CFX_GetImage(), described above, to get the image data from the database and store the data in a file.
- Compose a URL that points to the image file.
- Provide the URL to the "src" attribute of the IMG tag.

What are the trade-offs of storing the image in a database? Replicating the

database is easy. When the new copy is made, the images are copied to the new database. We don't have to worry about moving files or changing references to files, but there are several disadvantages.

It takes time to copy the image from the database to memory and then copy the image to a file. For Microsoft Access there's the additional overhead of converting the ASCII strings from base 64 or WDDX format. For a high-performance application, we'd prefer to avoid all this extra overhead. Also, when creating and storing the images in a file, we might discover we're out of disk space. If that happens at the point of needing to display the image to a user, we're in trouble.

One idea that arises at this point is: Why can't we read the image from the database and directly feed the binary image data to the HTML output stream with CFML? Wouldn't this avoid the overhead of writing the image data to a file and also skip the second file access when the IMG tag "src" attribute causes the file to be read from the server? Yes, it would be great to avoid all that overhead. I researched this idea extensively but concluded that CFML only outputs ASCII data and has no method (in Version 4.5.1) to feed binary data directly to the HTML output stream.

"The key to successful design of high-performance ColdFusion applications is to understand the final context in which the application will execute"

FIGURE 1: Rafting image

Our other design approach is to permanently store the image data in files and then place a reference to those files in the database. The steps to getting the next image to display are much simpler than the previous approach. We find the correct record, get the file reference or construct it from various fields, pass it to the IMG tag, and we're done. With fewer steps the code should execute much more quickly than "storing the image in the database" approach. Disadvantages?

It's more difficult to replicate the database. If the image file reference uses absolute file paths, we'll need to convert all the references in the database.

Keeping visitors happy with rapid page loads was the site owners' objective. Therefore, I decided to use the second design approach. If I had to spend a few extra minutes doing administrative work to replicate the database, then I was willing to make that trade-off.

Image File Reference

Once the primary design approach has been selected, we need to consider the secondary issue of the file reference. We could store an absolute path name or construct the file name from several path components. If we used absolute path names, we would need to modify them if we moved or replicated the database. I decided to steer away from those potential problems by choosing to construct the file reference from its constituent components. Starting with a full path we have:

```
C:\Images\Rafting.jpg
```

Breaking the full file name path into components, we have:

- ***path name,*** including disk and subdirectories, "c:\Images\"
- ***file name,*** "rafting"
- ***file type,*** ".jpg"

I stored the path name as an application variable in Application.cfm:

```
<cflock
scope="session"
timeout="20"
type="exclusive">

<cfset
Application.ImageServerPath
= "c:\Images\">
</cflock>
```

Thus, if the disk or subdirectory path changes, I can make one change to Application.cfm. For the file name component I created a field in the database table Images named, FileName. (What else would I name it?) A second table, named ImageTypes, stores the file types .gif and .jpg. Later I'll describe how these components are pulled together to form the full file name.

PR_ImageGet Custom Tag

The PR_ImageGet custom tag must perform these actions:

- Query the database to get a list of images available for display.
- Select and rotate between the images per an algorithm.
- Compose a file name to be used in the "src" attribute for the HTML IMG tag.

First, I'll briefly describe the database tables, then I'll discuss the preceding actions in detail.

Database Tables

The database for this application consists of two key tables: Periods and Images.

The first table, Periods, maintains a list of redefined time periods. For example, spring might be defined as <DateStart> to <DateEnd>; fall, as <DateStart> to <DateEnd>. The administrator assigns images to each of the time periods. Rafting and golfing images would be assigned to the spring and summer periods while mountain scenes of leaves turning color would be assigned to fall (see Figure 1).

The second table, Images, maintains information about each image. A Name field stores the file name portion of the file path, the PeriodType field defines whether the time period is Custom or Predefined, the PeriodStart and PeriodEnd fields store custom date ranges for image selection.

Querying the Database

First, we must get a list of the images available for display. Accessing the database can lead to potential delays. We want to ensure rapid query execution upon every call. To do this we make use of the cached-

within attribute of the <CFQUERY> tag. When this attribute is specified, the database driver will cache the results from a query in memory. Future queries with the same parameters will access the past results stored in memory, thus avoiding a potentially time-consuming database transaction.

The cachedwithin takes a time span object created with the CreateTimeSpan() function. For the image rotation application, I chose one hour as the time span:

```
<cfset
CacheTimeSpan =
CreateTimeSpan( 0, 1, 0, 0 )>
```

Now that we have a time span, I applied it to all database queries:

```
<cfquery
cachedwith=#CacheTimeSpan#
datasource="PicRotator"
name="Images">
```

For this to work we must enable query caching in the ColdFusion Administrator. Under the “Server” section in the control panel, select “Caching”. Scroll down to “Limit the maximum number of cached queries on the server to *n* queries”. If zero (0) sets the value to a positive integer, click on Apply.

Rotation Algorithm

My client and I discussed various approaches to the rotation algorithm. We settled on the following:

- The first image a visitor sees should be random, i.e., don’t always start at the first image in the rotation.
- Users should see all the images if they click on the front page a sufficient number of times within a session.

To implement this scheme I used application and session variables named Application.Row and Session.Row.

Application.Row stores an index to the starting image for the next visitor to the site. When a new visitor comes to the site, Session.Row is created with the current value of Application.Row. Application.Row is incremented. When it exceeds the number of images in the rotation, it’s reset back to 1.

Session.Row now starts at what appears to be a random point in the list of images since it is indeterminate how many new users visited the site in the time before the current user visits the site. For example, see the Image Rotation section in Listing 1.

Each time visitors come back to the front page within this session, we want to present them with the next image in the rotation. I simply increment Session.Row upon each visit to the front page and reset it to when the index exceeds RecordCount.

```
<cflock
scope="session"
timeout="20"
type="exclusive">
<cfset Session.Row=Session.Row+1>
<cfif Session.Row GT RecordCount>
 <cfset Session.Row = 1>
</cfif>
</cflock>
```

Composing the File Name

Now that we know which record to select from the query to the Images table, we need to get the file name components and form the full file name path. We did two queries from the Images table, one for images referencing Predefined time periods and one for images using a Custom time period. We could combine the two recordsets, but that would take extra processing time and I want to avoid that in this custom tag. Instead, I assume the two recordsets are contiguous and use a little conditional logic to determine from which recordset to pull the image information. See the section of code labeled Image Selection in Listing 1.

Using ImageTypeID we pull the file extension, either JPEG or GIF, from the ImageTypes table. Now we’ve got all the pieces we need to compose the full file name path. The disk and subdirectory path are stored in the application variable Application.ImageServerPath, the file name portion came from FieldName in either Images1 or Image2, and we have the file type in Type2. The full file name is composed with:

```
FileNameFull =
ImageServerPath & Name & "." &
Type2;
```

To prevent broken images coming up on the front page, we determine if the file actually exists on the server. Who knows? Somebody could have accidentally moved a directory or deleted a file.

```
if(NOT FileExists(FileNameFull))
{
FileNameFull = ImageDefault;
}
```

The caller will display the image by specifying the image file to the SRC attribute on the IMG tag. The file name must be in URL format:

```
FileNameUrl =
ImageUrlPath&Name&"."&Type2;
```

Finally we return the file name in URL format to the caller. To prevent stepping on variables in the caller’s variable scope, the caller passed us the name of a variable in which to return the file name in the attribute FileName:

```
Name = "Caller." & FileName;
SetVariable(Name,FileNameUrl);
Using <CFSILENT>
```

Comments and extraneous characters in the custom tag, such as white space, will be sent to the HTML output stream. This extraneous text will show up on the front page and could potentially upset HTML formatting. Also, quite minor, its additional characters need to be pushed down to the browser. To prevent such problems I’ve used the <CFSILENT> tag at the top and near the end of the custom tag. <CFSILENT> causes all output between the beginning and end tags to be discarded.

Understand Final Context

The key to successful design of high-performance ColdFusion applications is to understand the final context in which the application will execute. Ignoring this component of the development process leads to potentially slow-running applications and a possible redesign of the application.

About the Author

Ed Swartz is a senior software engineer and author with more than 20 years’ experience in the field. He is founder and president of Sandy Pond Consulting Group, Inc., a software development firm in Ayer, Massachusetts. He is also the author of The Engineering Minute *and* Successful Design Tips, *newsletters that provide insights into the software development process.*

ed.swartz@sandypondconsulting.com

Listing 1

```
<cftry>
<cfsilent>
 <cfinclude template="Application.cfm">

 <cflock scope="application" timeout="20" type="readonly">
 <cfscript>
  DateDefaults = Application.DateDefaults;
  ImageServerPath = Application.ImageServerPath;
  ImageUrlPath = Application.ImageUrlPath;
 </cfscript>
 </cflock>

 <cfset Debug = false>

 <CFPARAM default="false" name="debug">

 <CFSCRIPT>
  // We’ll check the query execution time for debug purpos-
es.
  // We want fast performance when getting the next image
file name.

  ExecutionTimeVar = Attributes.ExecutionTime;

  // We'll return the name of the image file to the
caller.
  // The caller will put the image file name in a <img
src="#FileName#"> tag.

  FileName = Attributes.FileName;

  // Set the default image file name just in case we have
a problem accessing the DB.

  ImageDefault = Attributes.ImageDefault;

  Variable = "Caller." & FileName;

  SetVariable( Variable, ImageDefault );

  // Assume no problems.

  Caller.StatusMessage = "";

  // Get today's month and day. The year is a 'base' year.

  Date = CreateDate( Year( DateDefaults ), Month( Now() ),
Day( Now() ) );

  DateNow = CreateODBCDate( Date );

  // To ensure fast performance when querying the DB we'll
cache the results for a few minnutes.

  CacheTimeSpan = CreateTimeSpan( 0, 1, 0, 0 );

  ExecutionTime = 0;
 </CFSCRIPT>

 <CFQUERY cachedwithin=#CacheTimeSpan#
datasource="PicRotator" name="ImageTypes">
  SELECT
   ImageTypeId,
   Type
  FROM ImageTypes
 </CFQUERY>

 <CFSET ExecutionTime = ExecutionTime +
CFQUERY.ExecutionTime>

 <CFQUERY cachedwithin=#CacheTimeSpan#
datasource="PicRotator" name="Images1">
  SELECT
   i.ImageTypeId,
   i.Name,
   i.PeriodStart,
   i.PeriodEnd,
   i.PeriodType
  FROM Images AS i
  WHERE ( i.PeriodType = 'Custom' ) AND ( i.PeriodStart <=
#DateNow# AND i.PeriodEnd >= #DateNow# )
 </CFQUERY>

 <CFSET ExecutionTime = CFQUERY.ExecutionTime>

 <CFIF Debug>
  <CF_SPCG_DumpQuery query=#Images1#>
 </CFIF>

 <CFQUERY cachedwithin=#CacheTimeSpan#
datasource="PicRotator" name="Images2">
  SELECT
   i.ImageTypeId,
   i.Name,
   i.PeriodStart,
   i.PeriodEnd,
   i.PeriodType
  FROM Images AS i,
   Periods AS p
  WHERE ( i.PeriodType = 'Predefined' AND i.PeriodId =
p.PeriodId )
   AND ( p.PeriodStart <= #DateNow# AND p.PeriodEnd >=
#DateNow# )
 </CFQUERY>

 <CFSET ExecutionTime = ExecutionTime +
CFQUERY.ExecutionTime>

 <CFIF Debug>
  <CF_SPCG_DumpQuery query=#Images2#>
 </CFIF>

 <cfscript>
  // Treat the 2 recordsets as if they are 1 contiguous
recordset.

  RecordCount = Images1.RecordCount + Images2.RecordCount;

  // Cycle the index back to the 1st record if we've gone
too far.
 </cfscript>

 // Rotation algorithm:

 <cfif NOT IsDefined( "Session.Row" )>
  <cflock scope="Application" timeout="20" type="exclu-
sive">
   <cfset OurRow = Application.Row>
   <cfset Application.Row = Application.Row + 1>
   <cfif Application.Row GT RecordCount>
    <cfset Application.Row = 1>
   </cfif>
  </cflock>

  <cflock scope="Session" timeout="20" type="exclusive">
   <cfset Session.Row = OurRow>
  </cflock>
 <cfelse>
  <cflock scope="session" timeout="20" type="exclusive">
   <cfset OurRow = Session.Row>
  </cflock>
 </cfif>

 <cflock scope="session" timeout="20" type="exclusive">
  <cfset Session.Row = Session.Row + 1>
  <cfif Session.Row GT RecordCount>
   <cfset Session.Row = 1>
  </cfif>
 </cflock>

 <cfscript>
  // Image Selection.
  if( OurRow LE Images1.RecordCount )
  {
   Name = Images1.Name[ OurRow ];
   ImageTypeId2 = Images1.ImageTypeId[ OurRow ];
  }
  else
  {
   i = OurRow - Images1.RecordCount;
   Name = Images2.Name[ i ];
   ImageTypeId2 = Images2.ImageTypeId[ i ];
  }

  // Get the file extension. Usually, GIF or JPG.

  Type2 = "";

  for( i = 1; i LE ImageTypes.RecordCount; i = i + 1 )
  {
   if( ImageTypeId2 EQ ImageTypes.ImageTypeId[ i ] )
   {
    Type2 = ImageTypes.Type[ i ];
    break;
   }
  }

  // SQL Server always returns records with trailing
spaces. Trim them off.

  Name = Trim( Name );
  Type2 = Trim( Type2 );

  // Make sure file is there before we give file name back
to caller

  FileNameFull = ImageServerPath & Name & "." & Type2;

  if( NOT FileExists( FileNameFull ) )
  {
   FileNameFull = ImageDefault;
  }

  // Caller needs path in URL format

  FileNameUrl = ImageUrlPath & Name & "." & Type2;

  Variable = "Caller." & FileName;

  SetVariable( Variable, FileNameUrl );

  Variable = "Caller." & ExecutionTimeVar;

  SetVariable( Variable, ExecutionTime );
 </cfscript>
</cfsilent>

 <cfcatch>
  <cfif Debug>
   <CF_SPCG_DumpCFCATCH struct=#CFCATCH#>
   <cfabort>
  </cfif>
  <cfset Caller.StatusMessage = "Non-Database error.">
 </cfcatch>

  <!-- We need to make sure the caller always runs. Just
return exit tag. // -->
 <cfcatch type="database">
  <cfset Caller.StatusMessage = "Database failure.">
  <cfexit method="ExitTag">
 </cfcatch>
</cftry>
```

CODE LISTING
▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶
The code listing for
this article is also located at
www.sys-con.com/coldfusion/sourcec.cfm

ColdFusion fasttrack

Michael Smith, right, interviews Gregory Silvano, CEO of CodeHound.

News from the show floor

Show Review by Michael Smith & Tobe Goldfinger

Over 2,000 developers attended the JDJEdge and Web Services Edge Conference in New York City. The conference was held September 23–26, 2001, just 12 days after the World Trade Center destruction. So we were a bit nervous coming into the city to the largest hotel in Manhattan – the Hilton New York on Avenue of the Americas, just south of Central Park. But we needn't have worried!

The conference was continuing as planned and was welcomed by Mayor Giuliani for showing confidence in the city. Against the backdrop of recent tragic events, there were certainly some glitches to overcome and last-minute speaker substitutions to arrange, but the SYS-CON Events team valiantly pulled off an exciting and information-packed event. Many speakers stepped up to the plate to cover presentations when the original speaker wasn't present.

Michael Smith interviews Judith Dinowitz of Fusion Authority.

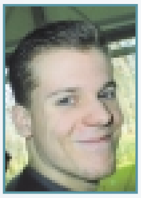
First stop, Monday, was the ColdFusion keynote speech given by Ben Forta. As usual Ben captivated the room with his charm and quick answers to any ColdFusion questions. Then I went down to hear the end of the Java keynote by the "Father of Java," Dr. James Gosling, who spoke on the history of distributed computing. His presentation put the up-and-coming world of Web services into historical perspective by comparing it to early mainframes and new PDAs. Later in the day conference attendees had the unique opportunity to "Ask the Doctor" specific questions.

The ColdFusion conference was split into two tracks – beginner and advanced. On the beginner side, Michael Smith, with the help of Charlie Arehart, presented a session on custom tags. The talk made up in enthusiasm what it lacked in polish, and the audience contributed with a lively Q & A session.

Next up was a riveting talk by Ben Forta on user-defined functions. Ben is a true evangelist when it come to UDFs and enjoyed telling everyone about the neat things that can be done with them and how they run faster than custom tags. Apparently there's even a library of free functions at the Common Function Library Project (www.cflib.org).

Over in the advanced track we learned about XML from Steve Heckler and how the ColdFusion WDDX function helps you write XML-enabled applications easily.

Steve Drucker gave a dynamic talk on using DHTML from ColdFusion. The trick is to use ColdFusion to generate special JavaScript to control the DHTML layers. As always, Steve dazzled the audience with both his code and showmanship.



Christian Schneider

Christian Schneider had flown in from Düsseldorf to speak about his live session tracker code. Basically, this code shows all visitors to a

Ben Forta delivered his sold-out session to an enthusiastic audience.

Round Up

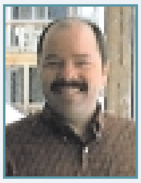
ColdFusion fasttrack

Over 1,200 delegates welcomed James Gosling's opening keynote on Monday, September 24.

site in real time, as well as the current page accessed and the time since they last clicked. Kind of spooky to realize ColdFusion makes all this information easily available via a structure in the application scope that Christian created. The code is available for download at www.christian-schneider.net.

Closing out Monday Michael Dinowitz, the organizer of the cf-talk list, Fusion Authority, and NYCFUG, spoke on two topics. Michael was walking with a cane due to a bad ankle, but to the audience he appeared to be the prophet of ColdFusion come down from the mountain! He first spoke on code reuse – something we would all like to do but aren't always able to. The methods suggested were CFINCLUDE, custom tags, and nested custom tags. He then gave a second talk on complex data types that covered arrays, structures, and the use of CFSCRIPT and functions to handle them. Both talks received loud applause from the attendees.

If It's Tuesday, It Must Be Database



Charles Arehart

Tuesday was an early start with master teacher Charlie Arehart explaining ColdFusion database basics to a packed classroom. He covered how to use data sources and basic SQL. In his second session, Charlie covered more advanced database topics such as slicing and dicing data using aggregate functions, grouping, and select distinct. He explained these underused SQL features in his easygoing style. Next door, Kevin Towers from Toronto was showing some neat data management tricks with ColdFusion and Flash.

Sneaking out to Web Services Edge we heard a keynote panel of industry leaders discussing the Web services paradigm. A spirited debate ensued that covered a variety of issues including SOAP, B2B components, and authentication. Even defining Web services was not such a simple matter. Most interesting was the audience poll, which indicated that practically no one at this point had implemented a production-ready Web service or had a system that was taking advantage of one. Clearly there's a lot of exciting talk going on in the Web services arena, but actual working examples are still hard to come by.

Back at the ColdFusion Fast Track, Michael Smith gave a talk on form handling that covered validation in ColdFusion, JavaScript, and CFINPUT. Charlie Arehart was back on his CF and database theme with performance and scalability issues, in particular using stored procedures and temporary tables to speed up your site.

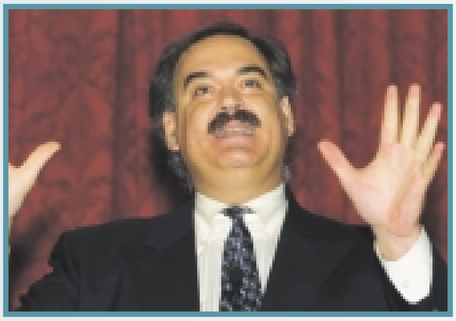
The database theme continued with Steve Drucker talking about advanced database concepts including T-SQL and PL/SQL. Steve mentioned that over 70% of all application performance problems are related to poor database design and bad SQL queries – so perhaps it was wise that we spent so much time on database issues on Tuesday!

Finally, on the beginner track Kurtis Leatham spoke on client state management using cookies, client, session, and application variables and reminded us how important it is to use CFLOCK with the last two types. The last advanced track class was given by Shlomy Gantz, who with his usual flare spoke about leveraging ColdFusion with dynamic Flash.

Java Track

While Michael was checking out the ColdFusion track, Tobe was at the Java track. With the Java-based CF 6 "Neo" coming out in 2002, it pays to be prepared with Java.

Richard Soley, CEO of the Object Management Group (OMG), gave a lively presentation on model-based application development. Often tongue-in-cheek but always on target, he reminded us IT professionals that "the next best thing" is always just around the corner. He strongly urged us to find ways to build our applications; to use not only the latest technology, but to architect for adaptability to the changes in infrastructure that will undoubtedly occur over the life of our applications. More about OMG's model-driven architecture can be found at www.omg.org/mda.


Dr. Richard Mark Soley delivered one of the keynotes.

Over the course of three days, more than 50 hour-long sessions were held, across four Java tracks and four Web services tracks, providing both high-level insight and strong technical education. There was plenty of opportunity to learn, to ask questions, and to network with other professionals. Having many great topics to choose from, I focused my attention on the latest advances in XML, JMS, EJB 2.0's message-driven beans, transaction management across a distributed Web architecture, proper testing methods, and extreme programming.

I found the presenters to be top-notch and truly knowledgeable with a wealth of information and ideas to impart. I'll certainly be taking advantage of the many great resource URLs, articles, white papers, tutorials, and more that were provided to supplement my continuing post-conference education.

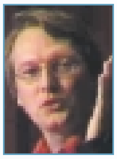
Exhibitors made numerous contacts on the Expo floor, which remained open for two days.

Don't Forget the Expo

In addition to the three conferences there was an Expo floor with about 40 companies showing their products. Most were for Java but one, CodeHound.com, was for ColdFusion too. It's a search engine for programming languages – Java, VB, and SQL are already covered and ColdFusion is due out in December. On the show floor SYS-CON radio was "Internet-casting" live via the Web and Michael and Charlie hopped on board and interviewed several people. All the ***CFDJ*** editors were at the SYS-CON booth, which was a hub of activity when the reception started at 6:00 p.m!

SYS-CON Media has long been known and respected for its excellent technical journals. It's evident that the new SYS-CON Events division is well on its way to earning the same well-deserved accolades.

About the Authors



Michael Smith is president of TeraTech (www.teratech.com/), a 12-year-old Rockville, Maryland–based consulting company that specializes in ColdFusion, Database, and Visual Basic development. Michael runs the MDCFUG and recently organized the two-day, Washington, DC–based CF2001 conference.

Tobe Goldfinger is the president of JDT Technologies, Ltd., a consulting firm specializing in ColdFusion and J2EE development. She has more than 20 years of large-scale systems consulting experience across a wide variety of technologies and is the comanager of the New York ColdFusion User's Group.

michael@teratech.com

tobe@jdttech.com

Conference registration desk: The largest gathering of Web services professionals in the industry.

MACROMEDIA

www.vue.com/macromedia

ColdFusion Feature

By Hal Helms and
John Quarto-vonTivadar

Fusebox 3.0

Let's get busy inventing the future

The art of progress is to preserve order amid change and to preserve change amid order.
—Alfred North Whitehead

It's finally here! For months, the lights have burned late and e-mails have flown furiously as work proceeded on the latest version of the Fusebox specification, version 3.0. It was released to rave reviews at the Fusebox 2001 conference held in Orlando on October 20, the Saturday prior to Macromedia's DevCon.

Someone once said that people should never see how laws or sausages are made. I think I can safely add technical standards to that list. There have been passionate arguments and some painful missteps along the way, but that's all eclipsed as people begin discovering the new power and ease-of-use in Fusebox 3.0.

What's New in Fusebox 3.0

Here's a rundown of some of the key features:

- A nested model for communication between circuits created to make reuse and distributed development easier
- A nested layout model that opens up possibilities for highly dynamic, flexible layouts
- XML-based Fusedoc providing both PDL (Program Definition Language) and documentation for fuses
- A stable set of key files that forms a Fusebox skeleton, making the learning process easier for people new to Fusebox and making it much easier to generate a new Fusebox application
- Exit fuseactions (XFAs) for greater reusability of fuses and circuits
- An API that exposes key variables within a Fusebox structure defined by the key Fusebox file

Fusebox Component Files

The core Fusebox files begin with an FBX_ prefix. A sample application is assumed with the circuit structure shown in Figure 1.

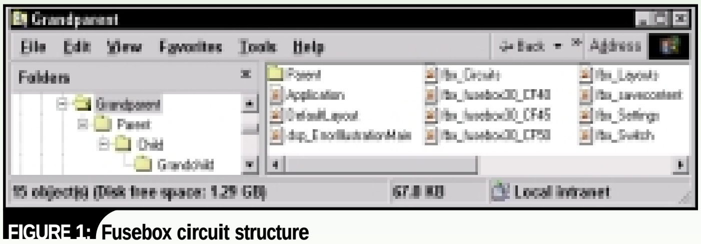


FIGURE 1: Fusebox circuit structure

MACROMEDIA

www.macromedia.com/go/usergroups

The component Fusebox files are:

FBX_Settings.cfm

This file is optional and is used to set variables that are needed by the application. Each circuit may have its own FBX_Settings.cfm file. If a fuseaction such as grandchild.main is resolved to home/parent/child/grandchild.main, FBX_ Settings.cfm will be called in each directory in order. This allows you to set general variables at higher levels and override them, if needed, in child directories. This file replaces myGlobals.cfm, app_ Globals.cfm, and app_Locals.cfm.

FBX_Circuits.cfm

Required in the home circuit, FBX_Circuits.cfm maps circuit aliases to physical paths. In the sample application shown, the contents of FBX_Circuits.cfm are:

```
<cfscript>
Fusebox.Circuits.home =
'Grandparent';
Fusebox.Circuits.parent =
'Grandparent/Parent';
Fusebox.Circuits.Child =
'Grandparent/Parent/Child';
Fusebox.Circuits.grandchild =
'Grandparent/Parent/Child/Grandchild';
</cfscript>
```

The circuit alias is a key within a structure called Circuits, with the reserved structure Fusebox. The circuit alias doesn't have to be the same name as the physical directory, as shown by aliasing the top directory (Grandparent) as home. This allows for directories at different levels to have the same name.

FBX_Switch.cfm

This file is placed in every circuit that handles fuseactions; FBX_Switch.cfm is a <cfswitch> statement with <cfcase>s for every fuseaction the individual circuit is to handle.

FBX_Layouts.cfm

Required in any circuit that implements a separate layout file, FBX_Layouts.cfm is responsible for setting the variables, Fusebox.layoutDir and Fusebox.layoutFile. The layout file pointed to in Fusebox.layoutFile must, at a minimum, output Fusebox.layout. Conditional processing is possible within FBX_Layouts.cfm. If, for example, a child sets a Boolean variable called QueryReturnedRecords, a parent might implement this simple logic in FBX_Layouts.cfm:

```
<cfif QueryReturnedRecords>
  <cfset
Fusebox.layoutFile="SortableTableLayout.cfm">
<cfelse>
  <cfset
Fusebox.layoutFile="EmptyRecordLayout
.cfm">
</cfif>
```

FBX_SaveContent.cfm

Used if the ColdFusion server version is below version 5.0, this file is the widely popular <cf_bodycontent> tag.

FBX_Fusebox30

Of the files FBX_Fusebox30_CF50.cfm, FBX_Fuse-box30_CF45.cfm, and FBX_Fusebox30_CF40.cfm, only one will be called in index.cfm, depending on which version of ColdFusion server is run-ning. The index.cfm file should have this code in it:

```
<cfif
Val(ListGetAt(Server.ColdFusion.Product-
Version, 1)) LT "5">
 <cfif
Val(ListGetAt(Server.ColdFusion.Product-
Version, 2)) is "5">
  <cfinclude template="fbx_fusebox30
_CF45.cfm">
 <cfelse>
  <cfinclude template="fbx_fusebox30_
CF40.cfm">
 </cfif>
<cfelse>
 <cfinclude
template="fbx_fusebox30_CF50.cfm">
</cfif>
```

Note: If you create circuit applications, the only file that's absolutely required for a circuit that only runs beneath other circuits is the fbx_switch.cfm file. The other files are needed only if you have special settings to set and special layouts to handle. Circuits are defined only in the home circuit. Of course, you do have to supply your own fuseactions and fuses; we had to leave some of the work for you, after all!

In addition to the core Fusebox files, DefaultLayout.cfm is provided. This file simply outputs Fusebox.layout. This is a useful file when you don't care about a given circuit adding any layouts and simply want it to display the layout. Note, however, that while the fuseaction path must have one layout file, you don't need any files in circuits that don't have any special layout requirements. In the "family" application whose directory structure is shown in Figure 1, a request for grandchild.main might, for example, have layout handled in only one of the circuits involved. In such a case the other circuits wouldn't require an FBX_Layouts.cfm file nor the use of DefaultLayout.cfm.

Reserved Variables: The Fusebox 3.0 API

There are two complex variable structures. The first, the FB_ structure, is used for internal calculations by the FBX_Fusebox_CFxx.cfm file. You should neither read nor write to this structure since it will immediately render your code non-Fusebox 3.0–compliant and may well break your application.

The second reserved variable structure is the public "API" called Fusebox (see Table 1). Again, you shouldn't make any changes to this API. You'll find some variables within this structure very useful. You can use these public variables anywhere within your code, as any changes to the Fusebox standard will support these variables going forward, even if the underlying code that calculates these variables is modified.

Core Fusebox 3.0 Concepts

With the new terminology, API, and required files under our belts, let's turn our attention to the latest additions to the "core Fusebox" methodology that makes its appearance in Fusebox 3.0. These are exit fuseactions (XFAs), Fusedocs, and nested circuits.

XFAs address issues of severability and reuse; Fusedocs address code documentation, especially in a distributed developer environment; nested circuits address inheritance, layouts, and exception handling. Although these are largely separate concepts, they do weave into the overall Fusebox scheme and are absolutely essential toward realizing the advantages that Fusebox 3 can deliver.

Exit Fuseactions (XFAs)

The first of the new concepts for Fusebox 3.0 is exit fuseactions. XFAs are the identified exit points of a particular fuse. In Fusebox the meaning of an exit point is not "Where do we go next?" (we always go back to the fusebox!), but rather, "What fuseaction are we going to take upon our return to the fusebox?"

For example, take the case where a user fills out a form to win the $1 million we're giving away. The user begins by getting an exciting, unexpected e-mail from us (really, it's not spam!) on which appears a link (see Figure 2):

```
<a href="http://scamsRus.com/index.cfm?
fuseaction=showContestForm">Click here
to win a million dollars!</a>
```

When the user submits that form, we want to store the information to a database while the user expectantly awaits a celebrity spokesperson's arrival with a substantial cashier's check. Regardless of what the spokesperson does, the form had only one possible way to return to the fusebox – by the user clicking the OK button. We might assign an exit fuseaction of “saveToDatabase” to that exit point, and represent it like this:

```
<form
action="index.cfm?fuseaction=saveTo
Database" method="post">
```

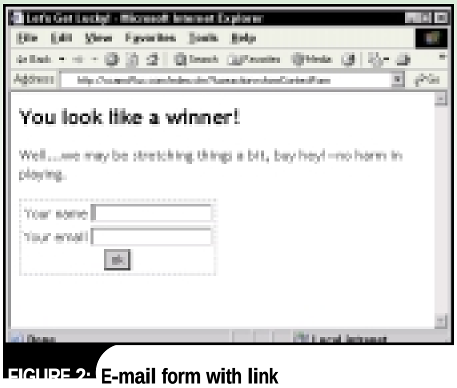
The problem with such a fuseaction is that it’s a hard reference. To reuse the fuse – either in a new application or in a new context in the same application – I need to either change code or introduce conditional logic into the fuse. Neither path is optimal.

A much better way to handle our sample case is to identify the exit points of a fuse – those paths out of the fuse – and to assign variable names to them. The fuse then becomes completely independent of the application or context in which it operates, allowing it to be easily reused. The form code looks like this:

```
<form
action="index.cfm?fuseaction=#XFA.
submitForm#" method="post">
```

The value of XFA.submitForm is then set (by the architect) in the new FBX_Switch.cfm file.

```
<cfcase value="showContestForm">
 <cfset XFA.submitForm =
"saveToDatabase">
 <cfinclude
template="qry_SaveContestEntry.cfm">
</cfcase>
```



You look like a winner!

Well....we may be stretching things a bit, but hey! –no harm in playing.

Your name

Your email

ok

FIGURE 2: E-mail form with link

Introducing XML Fusedocs

As programmers, we know that we should document our code, but we often create little or no documentation or, at best, documentation is done afterwards. At some firms documentation is taken much more seriously, but often the reason for that documentation is not conveyed. Without knowing why we should do something, how we should do it will never be clear. About the only thing that everyone seems to agree on is that documentation is something that’s good for you, like castor oil. That was the old way.

Fusedoc is different. It’s not about documenting your code; it’s about coding your documentation. The Fusedoc process makes the application architect responsible for documenting the application before coding even starts. Fusedoc supplies all the information the coder needs to complete the fuse.

Fusedoc details exactly what the fuse is expected to do, and the variables and resources it uses and produces (inputs and outputs). It takes the form of a comment at the top of the fuse file.

```
<!---
<fusedoc fuse="fuse_name" ver-
sion="2.0" language="ColdFusion">
fusedoc elements go here…
</fusedoc>
--->
```

In Fusebox 2.0, Fusedocs used a proprietary format that I developed. Starting with Fusebox 3.0, they’re now XML-based. If you’re familiar with XML, reading a Fusedoc is simple. If you’re new to XML, you may find the Fusedoc to be a great confidence builder toward working more with XML.

We’re trying to provide enough information so that a competent programmer, who knows nothing of the applica-

tion nor of the underlying database, can write the code. It's sometimes referred to as *programming by contract* because the comments provided form a sort of "work order" that tells the coder exactly what to do. The "contract" promises the programmer, "You fulfill this work order and you will be done with the code. You have no responsibility beyond what the document tells you."

With Fusebox 3, Fusedoc goes from a proprietary symbology to the open standard of XML. There's a document type definition (DTD) available at www.halhelms.com that provides the formal specification for Fusedoc. To give you a taste of what one looks like, Listing 1 provides a complete Fusedoc for a fuse, act_ValidateLogin.cfm.

More information on Fusedocs and Fusebox 3.0 is available at www.halhelms. com.

Nesting Circuits and Layouts

In Fusebox 2, integration between separate fuseboxes consisted mainly of "tricks" for sending a browser from one insular fusebox to another. But the true goal is a "drag-and-drop" type of functionality, in which a fusebox consists of one of more circuits (a file directory), all of which can access each other without direct dependence on the underlying directory structure and that, when done, can be dragged-and-dropped to another fusebox, requiring only a few small settings to be changed to the definition of the circuits.

If this type of nesting is combined with some sort of a standard with respect to how a fusebox operates within its own framework, we're able to write large amounts of reusable code. If we continue to adhere strictly to a standard for doing that, like Fusebox 3.0, then people can write entire applications that can be stamped as "Fusebox 3.0–compliant" and then dropped into our own code, saving time and money.

Public API variable	Type	May be set?	Description
Fusebox.isCustomTag	Boolean	No	Is this fusebox being called as a custom tag?
Fusebox.isHomeCircuit	Boolean	No	Is the directory of the file currently being operated on by FBX_Fusebox_CFxx.cfm the same as the home circuit of the app?
Fusebox.isTargetCircuit	Boolean	No	Is the directory of the file currently being operated on by FBX_Fusebox_CFxx.cfm the same as the target circuit of the app?
Fusebox.fuseaction	String	No	The simple fuseaction extracted from the attributes.fuseaction of form circuit.fuseaction passed in
Fusebox.circuit	String	No	The simple circuit extracted from the attributes.fuseaction of form circuit.fuseaction passed in
Fusebox.homeCircuit	String	No	The circuit alias of the home circuit of the app
Fusebox.targetCircuit	String	No	The circuit alias of the target circuit of the app; usually this is the same as fusebox.circuit unless you've made a circuits definition error
Fusebox.thisCircuit	String	No	The circuit alias of the directory of the file currently being operated on by FBX_Fusebox_CFxx.cfm
Fusebox.thisLayoutFile	String	Yes	The layout file to be applied to this circuit after the fuseaction and its fuse(s) are finished creating their content
Fusebox.thisLayoutDir	String	Yes	The relative path from the target circuit where the layout file to be applied to this circuit is located; if no special layout directory has been created, this should be set to an empty string
Fusebox.CurrentPath	String	No	The relative directory path of the file currently being operated on by FBX_Fusebox_CFxx.cfm, relative from the root of the application (i.e., the home circuit); example: dir1/dir2/dir3/
Fusebox.RootPath	String	No	The relative directory path of the file currently being operated on by the core frozen fusebox code, relative to the root of the application (i.e., the home circuit); example: ../../../
Fusebox.layout	String	No	The variable used by cfsavecontent or its equivalent custom tag that captures the generated content to that point in preparation for wrapping a layout file around it (as defined by fusebox.layoutfile); this variable must be inside each layout file for content to be passed up to the next level of nested layouts
Fusebox.SuppressErrors	Boolean	Yes	FBX_Fusebox_CFxx.cfm has some abilities built in to suppress native ColdFusion errors and instead give its best guess at what's wrong with your application (as it relates to Fusebox); default value is always FALSE, which therefore generates native ColdFusion errors. You may want to set it to TRUE while you set up your application in Fusebox style, and let it default to FALSE when you're confident that your Fusebox is set up correctly, but are testing for coding errors in your fuses unrelated to Fusebox

TABLE 1 Public API variables

Fusebox 3.0 has its own concept of inheritance that, while very different from object-oriented languages, allows children to inherit properties from their parent, grandparent, and so on. Each of these circuits can be "dragged-and-dropped" into a parent circuit, resembling those wooden Russian dolls in which one doll contains another one. This pattern continues for as long as the artistry of the maker can sustain itself. In Fusebox 3.0 the grandparent circuit may contain one or more parent circuits that may contain one or more child circuits, and so on.

Each circuit, from the top or home circuit and continuing down each intervening circuit to the target circuit, has a settings file called FBX_Settings.cfm. The more global an item in a settings file is, the "higher up" the circuits tree that item would be loaded. So if your application accesses a data source called customers, you may want to set Request.DSN in the home circuit.

Suppose you dropped in a shopping cart circuit that needs a products data source. It might set one as part of its own settings file since that DSN may only be needed by that circuit or one of its child circuits. Of course you, as the architect of your program, can determine whether you want to force it to have a certain value by using cfset or whether you want it to inherit a value from a higher-level circuit – if such a value has already been set – by using a cfparam. In earlier versions of Fusebox, you included the app_globals.cfm once and then used app_locals.cfm for all second-tier circuits. In Fusebox 3.0 there's no hard meaning to "global" or "local" – anything set in a circuit is available to all circuits nested beneath that circuit. Again, remember the Russian doll.

Fusebox automatically reads in any FBX_Settings.cfm files from all circuits in the fuseaction path. Once the target circuit is reached, FBX_Switch.cfm is called, executing the target fuseaction within that circuit. This concept has not changed from Fusebox 1.0.

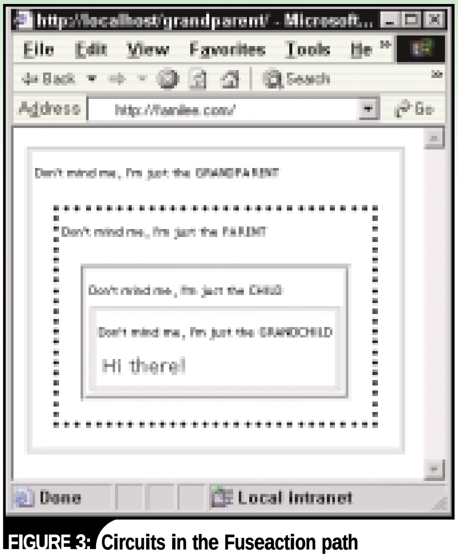
The concept of nested layouts is a very exciting addition. Having walked "down" the fuseaction circuit tree and executed the fuseaction, Fusebox now walks "up" the tree from the bottom "target" circuit where the fuseaction was found, up through each parent circuit up to the home circuit. Each of those circuits, in order, will have the opportunity to apply its own layout as determined by the FBX_Layouts file, just as you might put a Russian doll back together – each of the larger dolls taking what they got from the smaller doll and wrapping their own wrapper around it. While you can make use of every circuit in the fuseaction path, you don't need to. Figure 3 is a teaching example that shows the use of layouts at each circuit in the fuseaction path.

Migrating from Fusebox 2 to Fusebox 3

The learning curve for FB3 has been made easy due primarily to the standardization of the underlying core Fusebox code combined with a standardization in the naming convention. For these same reasons your migration path from FB2 to FB3 will be fairly easy, especially after you've achieved your first application with the new standard.

What about some of the underlying files you've dealt with before? Well, app_locals.cfm and app_globals.cfm are now combined into a single file called FBX_Settings.cfm. Application.cfm is still not a required file since the functionality you might put there can be put into FBX_Settings.cfm as well.

If you were a fan of Steve Nelson's <cf_bodycontent> custom tag for "wrapping" layout with common design elements, that functionality is still there too.



Don't mind me, I'm just the GRANDPARENT

Don't mind me, I'm just the PARENT

Don't mind me, I'm just the CHILD

Don't mind me, I'm just the GRANDCHILD

Hi there!

FIGURE 3: Circuits in the Fuseaction path

Now, though, you don't need a "header" and a "footer" file, rather you can define actual layout files that include both header and footer elements in a way that's fairly transparent. Again, the "frozen" core fusebox file will pick up your layout files as well.

In Fusebox 2, index.cfm (or the default document) had all the core functionality of Fusebox written into it. In Fusebox 3.0 a new file, FBX_Fusebox30_CFxx.cfm – where xx is a version that's appropriate for the version of the ColdFusion application server your machine is running – takes its place.

Circuits are a new idea for the standard, one that entirely replaces Fusebox 2's ad hoc "dot-dot-slash" method of making one fusebox talk to another. With Fusebox 2 a fuseaction usually had a verb-noun phrase syntax, such as addItem ToCart. In Fusebox 3, fuseactions are written with a circuit_alias.fuseaction syntax. To solve the problem of circuits on different levels having the same name, an FBX_Circuits.cfm file has been introduced. It maps individual circuits to an alias of your choosing. Nesting/inheritance in Fusebox 3.0 is done for you automatically by the core fusebox file, FBX_Fusebox_CFxx.cfm, using the FBX_ Circuits.cfm file's mappings. Assume that we have an application with the circuit structure shown in Figure 4.

Assume that a fuse within the Granddaughter circuit executes this code:

```
<cflocation
url="#self#?fuseaction=#XFA.
continue#"
addtoken="yes">
```

The variable, self, is defined in the home circuit's FBX_ Settings.cfm as index.cfm (or default file) of the home circuit –

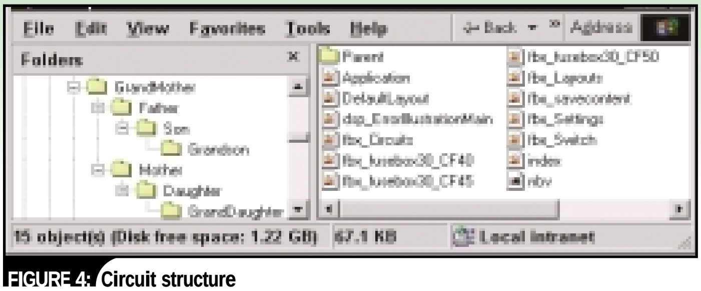


FIGURE 4: Circuit structure

Grandmother, in our case. In addition the variable XFA.continue is defined within FBX_Switch.cfm as Grandson.sayHello. All of this is under the control of the application architect.

All action returns to the home circuit's default file. This file includes the version of FBX_Fusebox30_CFxx.cfm that's appropriate for the ColdFusion server that's running. Within this file Fusebox translates the fuseaction, Grandson.sayHello, to the fuseaction path, in effect saying, "You said Grandson.sayHello, but I see from FBX_Circuits.cfm that what you really mean is Grandmother/Father/Son/Grandson.sayHello. Let me execute that for you." All that's required from you is that you properly map the circuits in FBX_Circuits.cfm, a very easy matter, and use the compound circuit_alias.fuseaction syntax.

Going Forward...

Fusebox 3.0 represents a major landmark in the Fusebox community's approach to creating a methodology that consistently produces predictable success in Web application development, but it is just that – a landmark, a waystation in an ongoing journey. Fusebox began with ColdFusion and is being adopted by developers working in PHP, ASP, and JSP.

Where do we go from here? Much work remains to be done. With Fusebox 3.0 we've achieved a platform on which we can build even more firmly. In the coming months new work will begin on making ColdFusion even better. We hope that you'll be a part of this venture.

How can you be involved? First, join the Fusebox community officially by signing up at www.fusebox.org. Next, join the Fusebox mailing list; a signup form is available at www.hal helms.com. If you'd like to participate in making the future of Fusebox, sign up for the Fusebox steering mailing list by sending an e-mail to steerFB-subscribe@topica.com.

To learn more about being a contract "Fusecoder," check out Steve Nelson's Web site at www.secretagents.com. Lee Borkman is heading up an open-source effort to make wireframes even better; information is available at www.bjork.net. Jeff Peters has some wonderful tools available for Fusecoders at www.grok fusebox.com. John Quarto-vonTivadar has articles available at www.grokdotcom.com (free newsletter signup) and www.john quarto.com. Finally, you may want to sign up for my free Occasional Newsletter at www.halhelms.com.

Alan Kay, the creator of the language Smalltalk, once said, "The best way to predict the future is to invent it." Let's get busy.

About the Authors

Hal Helms (www.halhelms.com) is a Team Allaire member who provides both on-site and remote training in ColdFusion and Fusebox.

John Quarto-vonTivadar, CTO of FutureNow, Inc., frequently speaks and writes on technology topics for a business audience in GrokDotCom.com's free newsletter and on ClickZ. His company specializes in the psychology of Web browsing and how the use of targeted language influences the online purchase decision process.

hal.helms@teamallaire.com

jcq@mindspring.com

Listing 1

```
<!---
<fusedoc
 fuse="act_ValidateLogin.cfm"
 version="2.0"
 language="ColdFusion">
 <responsibilities>
  I validate a user login. If MatchUser returns any rows, I
create a structure with the info returned by the recordset
and return to the fusebox with XFA.success; else XFA.fail-
ure.
 </responsibilities>
 <properties>
  <history
   type="create"
   date="23 Sep 2001"
   email="hal.helms@teamallaire.com"
   role="architect">
 </properties>
 <io>
  <in>
   <string name="self" />
   <string name="XFA.success" />
   <string name="XFA.failure" />
   <recordset name="MatchUser" primarykeys="userID" />
    <number name="userID" precision="integer" />
    <string name="firstName" />
    <string name="lastName" />
    <number name="userGroups" precision="integer" />
   </recordset>
  </in>
  <out>
   <structure
    name="CurrentUser"
    scope="client"
    format="wddx"
    optional="true"
    oncondition="if MatchUser returns non-empty rows"
    comments="match values in MatchUser to those in the
structure.">
    <number name="userID" precision="integer" />
    <string name="firstName" />
    <string name="lastName" />
    <number name="userGroups" precision="integer" />
   </structure>
  `<string name="fuseaction" comments="an XFA" />
  </out>
 </io>
</fusedoc>
--->

<!--- here beginneth the code --->
<cfoutput>

<cfif MatchUser.recordCount>
 <cfset str = StructNew()>
 <cfset str.userID = MatchUser.userID>
 <cfset str.firstName = MatchUser.firstName>
 <cfset str.lastName = MatchUser.lastName>
 <cfset str.userGroups = MatchUser.userGroups>

 <cfwddx
  action="CFML2WDDX"
  input="#str#"
  output="client.CurrentUser">

 <cflocation url="#self#?fuseaction=#XFA.success#"
addtoken="yes">
<cfelse>
 <cflocation url="#self#?fuseaction=#XFA.failure#"
addtoken="yes">
</cfif>

</cfoutput>
```

CODE
LISTING

The code listing for
this article is also located at
www.sys-con.com/coldfusion/sourcec.cfm

ColdFusion and XML-RPC Part 2 of 2

Extensible connectivity between almost any technology

BY RONALD WEST

In the first installment of this article ***CFDJ*** (Vol. 3, issue 9), I established the benefits of secure communication between two applications.

I discussed the need for standardized communication that allows two companies to share data that would in turn enhance the service offerings of their existing applications. Through the use of the cfservlet tag, I created the foundation for communication between the ColdFusion Application Server and the JRun Application Server.

In this article I demonstrate the actual connection between the CF Server and the JRun Server, display a few intricacies associated with the exchange of data between CF and JRun, show what XML-RPC is and what it looks like, and finally work with data returned from the connected system.

Before I continue I would like to establish the theme behind this work and any work that can become the basis for true Web services. Throughout this article it should be clear that even though I utilize CF and JRun as the framework for connectivity, the real focus here is the XML-RPC, which is completely independent of any application platform.

It's also important to note that when establishing a connection between disparate systems through the use of XML-RPC, or any other XML language, it's essential to define the system in a standardized way. That means that the data models used to exchange the standard data should be general enough to allow any system to work with them. In other words, don't use data structures (data exchange functions) that are specific only to the technology you use. Instead try and establish a standard that can easily be worked with by almost any application platform. In this case, XML-RPC will be the standard means for communication, and any platform that can send, receive, and process XML-RPC packets can work with this system.

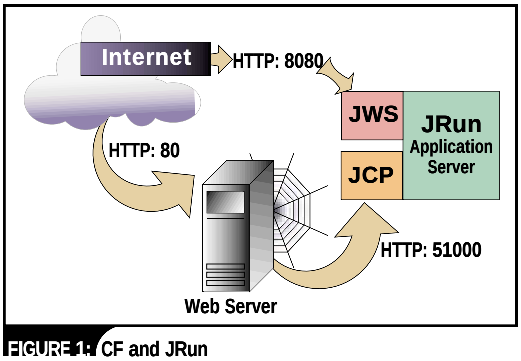


FIGURE 1: CF and JRun

Most of the work done here to create the connection and ultimately utilize the connection between CF and JRun can be performed by someone with a little knowledge of Java. In other words, it's not necessary to be an established Java champion in order to send, receive, and process XML-RPC packets to and from CF and Java. It's safe to say, however, that without a working knowledge of Java servlets, Java data structures, and the general workings of the HTTP protocol, this will seem a little difficult.

Now let's continue with the connection to JRun.

Connection Between CF and JRun

Prior to establishing a connection between CF and JRun, it's important to take a little time to discuss the framework involved here. The JRun has two basic connection points: the Java Web Server (JWS) endpoint and the Java Connector Proxy (JCP) endpoint. Although I focus entirely on the JCP endpoint, I would like to briefly discuss the difference.

The JRun Application Server comes equipped with its own Web server that can be configured to accept requests from a Web browser: the JWS. The requests to administer the JRun Application Server and to launch servlets and JSPs directly from a Web browser are handled by the JWS. The JRun Application Server is also equipped with a different connecter module that allows JRun to handle requests sent to it from another Web server: the JCP.

The JCP will handle all of the requests from the ColdFusion Application Server, but it's important to understand the difference as it can be confusing if not explained properly (see Figure 1 for clarification of the connection types).

In Part 1 I established that the use of the cfservlet tag from a ColdFusion template allows us to send data to a Java servlet. The cfservlet tag has parameter tags that allow it to send data via two different methods: value and reference. Let's handle a simple send of data via value.

In the example in Part 1, I discussed that I wanted to check and see if an Australian suburb was matched properly with its state name. To do so I need to connect to a module in the third-party application that will handle the process for us, the "CheckSuburb" module in this case. The cfservlet tag creates the connection to the servlet named *Connect*. It then sends along by value the ColdFusion variable "module" (that contains the name of the module we would like to connect with) to the JCP URL. From here the JRun Application Server loads the servlet and passes in the necessary values.

Listing 1 shows what types of methods are used with the data sent via the connection from ColdFusion. In the case of our module, I use the getParameter() method that retrieves the value of the passed-in parameter, *Module*. In effect, I create a new Java variable named *mCall*, and it gets the

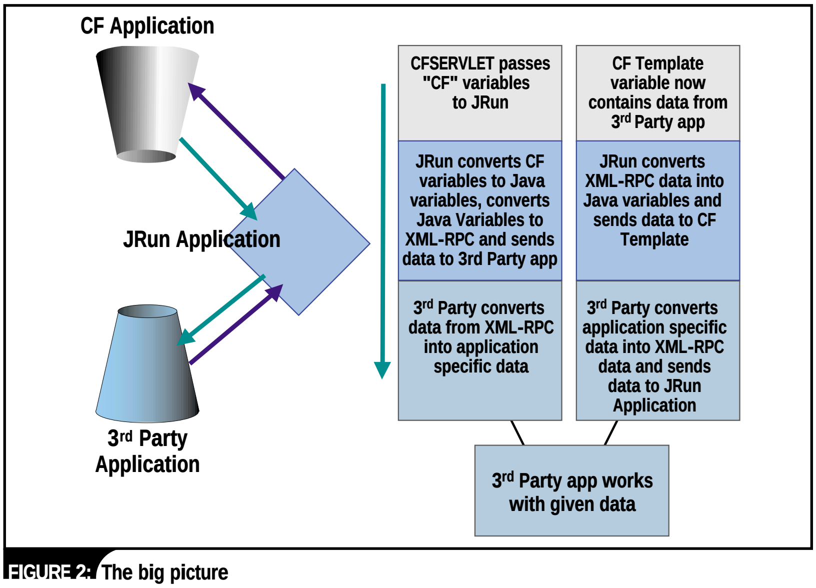

CF Application
JRun Application
3rd Party Application
CFSERVLET passes "CF" variables to JRun
CF Template variable now contains data from 3rd Party app
JRun converts CF variables to Java variables, converts Java Variables to XML-RPC and sends data to 3rd Party app
JRun converts XML-RPC data into Java variables and sends data to CF Template
3rd Party converts data from XML-RPC into application specific data
3rd Party converts application specific data into XML-RPC data and sends data to JRun Application
3rd Party app works with given data

FIGURE 2: The big picture

value of "CheckSuburb". This implementation of XML-RPC for Java contains all of the necessary components to connect to any application through the use of XML-RPC.

In the example, the get-Parameter() method allows me to send data from ColdFusion into a Java servlet. This type of connection occurs as a normal HTTP form post. The sending page connects to the processing page and through a post method sends variables that contain data. This name/value pair option will be extremely useful for expanding the simple module call, enabling us to make specific function calls that can be dependent upon the module we are attempting to connect to.

Although the simple name/value pair data connection option is useful, there's a more useful option to this system that I'll use to send data back to the Cold-Fusion template.

I can send the data to the Java servlet by reference. In the demonstration I created two types of variables to send via reference. I have the "data" variable that contains the structure with keys "State" and "Suburb" and I have the two variables "result" and "result2". Although I won't change the "data" variable, I reserve the right to do so for possible future use.

To read the data from a variable that's sent via this means, I utilize the getAttribute() method. In this demonstration, I always send a structure, so this new Java variable "data" must be cast as type Hashtable. This allows the original ColdFusion data structure to remain intact and the data to be accessible within the Java servlet.

- **Quirk #1:** ***Connecting CF and Java Through JRun***

At this point it's important to discuss one of the major differences between CFML and other application languages. ColdFusion, for reasons associated with "speed of programming," isn't case sensitive. In other words, the following holds true:

```
Dog=doG
```

Therefore, when I create the structure "data" with keys "Suburb" and "State", the ColdFusion Application Server represents the data with uppercase key names so that the "Suburb" key becomes a key named "SUBURB". ColdFusion can handle this, but most other languages can't. If our third-party application is made up of a case-sensitive language, which in this case it is, the structure won't contain the key "Suburb" that it's looking for and probably won't continue. So, for this application, I created a Java module that will parse through the structure and rename the keys appropriately (modifyKeys()).

- **Quirk #2:** ***CF Functions for Arrays***

It's important to note that there's a general issue when attempting to send and receive an array through this type of connection. ColdFusion doesn't store data in the "0" index of an array. Instead it begins the data storage in index "1". When you transmit an array to Java with no data in the "0" index, it throws an error. Also, when you receive data from Java that is of type Array and attempt to work with the CF Functions for Arrays, they'll throw an error.

It's possible through some different data mappings to work with arrays. Basically this involves some swapping of data into and out of structures. The best solution I have is to stick with structures when communicating in this format.

JRun with Third-Party Applications

Although the connection between the ColdFusion Application Server and the JRun Server is quite proprietary, you must understand that the "data" variable I send from CF to JRun is a standard data structure. The process to get that data from CF to JRun isn't of great importance. Once the connection is made, it's easier to think of the process as almost a direct connection between CF and the third-party application in that I send data to and from in some standard format. The JRun Server becomes just a quick intermediary step necessary to package and receive data in XML-RPC format.

Specifically, the "Connect" servlet takes the data that I send, formats it in XML-RPC (the addParameter() method), and sends it on to the third-party application (the send-Call() method). With an HTTP connection open, the servlet waits for a response from the application (see Figure 2).

In Listing 2 you can see what the XML-RPC packet looks like. The data structure is easy to read and contains all of the information needed to perform a checkSuburb method call. It's in a simple XML format and can be read by any system that is configured to accept this type of data.

Data from Third-Party Application

Here is where the connection gets really cool. After the JRun application sends the data, it waits for a response. In the meantime, the third-party application converts the data from XML-RPC into application platform–specific data and begins to work with it. In this case the structure data is fed through the checkSuburb module and a string-formatted data element is returned. The third-party application then packages the data into XML-RPC and sends the data back to the JRun Application Server.

The data returned to JRun can be in any format. I simply cast the data returned as type Object in order to maintain its original format. In this case the data type String will be converted from XML-RPC and placed into a Java object variable named “result”. The value given to “result” from the third-party application will be sent back to the CF template by using the setAttribute() module. The original CF variable “result” sent to the Java servlet via reference now has the value given to the Java object variable “result”. The CF variable “result” contains the data necessary to continue processing the ColdFusion request.

Error Checking and Security

Another feature of the XML-RPC connection is that it has a standardized error-reporting system. Standard errors have been created that can be used to inform either system of specific problems, such as authentication, data structure problems, or processing errors in general. The ColdFusion application can be configured to perform different tasks to handle different errors. The system can also be modified so that errors can be added and special error-handling processes can be created to encompass flexible error checking.

Security needs arise whenever the transmission of sensitive data over the Internet is discussed – especially in this case where I have potential business logic modules available to the general public. When transmitting data over the Internet, security needs can be serviced by a simple SSL connection. The described system relies heavily on the HTTP protocol, which currently has means for encryption. The XML-RPC implementations need to simply adhere to existing standards for the transmission of secure data.

Another security issue is data module access. Utilizing a username and password can restrict a module’s access; however, it’s a better practice to design modules to be protected against potential harmful access attempts.

New Pathways Opening Up

Once created, the XML-RPC connection is simple to use and beneficial. On the whole, the cfservlet tag can be seen as a custom tag or cfmodule, where data gets sent to the module, some processing of the data occurs, and then it’s returned. The process creates a useful tool to communicate between systems with disparate technologies, which is the main focus for Web services.

Many new standards are being discussed that allow for easy communication among different technology systems. They will open up new pathways for business partnerships and revenue streams. The combined power of ColdFusion and JRun creates a complete system that allows for extensible connectivity between almost any technology. As Java and XML become industry-wide standards – and ColdFusion and Java become more and more integrated – XML-RPC solutions and similar implementations will become easier to work with.

RWEST@ACESTECHNOLOGIES.COM

ABOUT THE AUTHOR

Ronald West is the director of technology for ACES Technologies, where he designs and manages large-scale ASPs for the healthcare industry. He has designed and managed Internet services for middle-sized underwriters, including intranet, extranet, and network architecture design and deployment.

Listing 1: Connect Servlet

```
public class Connect extends HttpServlet
{

 private String mCall;
 private Hashtable data;

public void service( HttpServletRequest req,
HttpServletResponse res )
      throws IOException
 {

  System.setProperty("com.tm.debug.level", "error");

  // Create a HTTP Transport.
  Transport transport = null;

  // Catch exceptions creating the transport.
  try
  {

   transport = new HTTPTransport("http://<<adderess>>" );

  }
  catch (Exception e)
  {

   System.err.println("Could not connect to server: " +
e.toString());
   System.exit(1);

  }

  // Create a Call object.
  Call call = new Call();

  mCall = req.getParameter("methodCall");
  data = (Hashtable)req.getAttribute( "data" );

  // modify all of the keys in the structure
  modifyKeys( data, mCall);

  // Set the method name in the form of
  // HANDLER.METHOD.
  call.setMethodName( mCall );

  // Add a parameter to the method.
  call.addParameter(data);

  // Send the Call, and get a Response.
  Response response = transport.sendCall(call);

  // Check to see if the response was a fault response.
  /*if (response.isFault()) {
  // Faults are structs, so cast them to a Hashtable.
   Hashtable fault = (Hashtable)response.getResponse();
   System.err.println(fault.get("faultCode"));
   System.err.println(fault.get("faultString"));
   System.exit(1);

  }*/

  // create a standard result and send it back
  Object result = response.getResponse();

  req.setAttribute( "result", result );
  req.setAttribute( "result2", data.toString() );

 }

 public void modifyKeys( Hashtable data, String methodCall
)
 {

  String val = "";
  Hashtable valueStruct;

  if( mCall.equals("checkSuburb") )
  {

   // change State key
   val = (String)data.get( "STATE" );
   data.remove( "STATE" );
   data.put( "State", val );

   // change State key
   val = (String)data.get( "SUBURB" );
   data.remove( "SUBURB" );
   data.put( "Suburb", val );

  }

 }

}
```

Listing 2: XML-RPC package

```
<?xml version="1.0"?>
 <methodCall>
   <methodName>checkSuburb</methodName>
    <params>
     <param>
      <struct>
       <member>
         <name>Suburb</name>
         <value><string>NSW</string></value>
       </member>
       <member>
         <name>State</name>
         <value><string>Sydney</string></value>
       </member>
      </struct>
     </param>
   </params>
</methodCall>
```

CODE LISTING

The code listing for this article is also located at

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

Macromedia Flash 5 and ColdFusion 5

Two techniques for quick integra

ColdFusion Feature

By Mike Chambers

Even if you have only cursory knowledge of Macromedia Flash, learning how to integrate it with ColdFusion is surprisingly easy.

While there are more advanced approaches, the techniques described in this article are extremely powerful – due in part to their simplicity – and lay the foundation for more advanced integration.

Why Flash 5 and ColdFusion 5?

Due to its ease of authoring and the ubiquity of its player, Macromedia Flash has become the standard for creating and displaying animated content on the Web. With the addition of the ECMA Script–based ActionScript in Flash 5, Flash has quickly become a popular tool as a front end to server-side content and data.

Whether it's used for creating interactive training applications, dynamic user interfaces to existing Web applications, or just simple animations, Macromedia Flash is a powerful solution for designing and developing interactive, rich content in a small, vector-based format. Content published to the Macromedia Flash (SWF – Shock Wave Flash) file format is browser and platform independent and instantly reaches over 97% of the online audience through Macromedia Flash Player.

But while Flash can display and manipulate data in a browser, this capability is much more powerful when coupled with the dynamic data generating capabilities of ColdFusion. By providing database connectivity and a server-side scripting environment, ColdFusion allows you to dynamically generate the content or data that will be displayed using Flash. As a result, you can easily keep the content on your Flash site up to date as well as create more personalized and interactive Flash-based applications.

Passing Data from ColdFusion to Flash via the URL

Creating the ColdFusion Code

There are many ways to integrate ColdFusion and Flash, but the easiest and most straightforward is to send name/value pairs from ColdFusion through the HTML that loads the Flash movie.

Let's look at the HTML necessary to load a Flash movie into a Web page. The HTML in Listing 1 was automatically created by the Flash authoring environment by clicking the HTML tab in the publish settings window (File > Publish Settings).

I won't go over every line of code here, but we need to pay particular attention to two things. First, notice that there are both OBJECT and EMBED tags, each with similar attributes. Both tags are necessary to display the Flash movie in browsers that use the ActiveX Flash plug-in (such as Internet Explorer on Windows) and browsers that use a plug-in architecture (such as Netscape and IE on Macintosh). If you know that your Flash movie is going to be viewed in only one browser, then you can use just the tag that corresponds to that browser.

The second thing to pay attention to is the sections of the tags that specify the name and path of the Flash movie to be loaded.

```
<PARAM NAME=movie VALUE="movie.swf">
```

and

```
<EMBED src="movie.swf">
```

You can use either a relative or an absolute path to the Flash movie, with relative paths being relative to where the HTML page was loaded from.

By adding a query string of name/value pairs to the end of this URL, you can pass variables and their values from ColdFusion to the Flash movie. The variables will then be available to use in the Flash movie on the main timeline as soon as the movie begins to execute.

For example:

```
<PARAM NAME=movie
VALUE="movie.swf?cfFoo=bar&cfBiff=bim
%20bam ">
```

passes the variables cfFoo with the value of "bar" to the Flash movie. The variable cfFoo could then be used in the Flash movie like any other Flash variable. Just remember that the variable will be loaded on the main timeline of the Flash movie (_level0._root).

The query string is just like any other HTML query string and allows you to pass multiple name/value pairs by separating each pair with an ampersand (&). However, just like normal HTML query strings, you must URL encode the data in the query string when passing data to Flash.

Here is a URL that passes two variables to a Flash movie. Note that the data and not the variable name is URL encoded:

```
<PARAM NAME=movie
VALUE="movie.swf?cfFoo=bar&cfBiff=bim
%20bam">
```

Using this technique you can easily pass name/value pairs from ColdFusion to Flash by having ColdFusion dynamically create the query string calling the Flash movie.

The variables are named *cfFoo* and *cfBiff*, not just *foo* and *biff*. The "cf" stands for ColdFusion and is used to make it easier within Flash to see that the variables come from ColdFusion. This isn't strictly necessary, but makes developing and debugging your Flash movies much easier.

Listing 2 is a simple ColdFusion page that initializes two ColdFusion variables and then passes them to a Flash movie by dynamically creating the URL loading the Flash movie.

We use CFSET to initialize two simple variables. (In the interest of simplicity, this example uses static variables, but in most real-world examples these values would be the result of a database query.) We then write out the HTML that will call the Flash movie. This is the same as the HTML discussed earlier, except that the query string appended to the end of the Flash movie's URL is dynamically created by ColdFusion.

```
 <cfoutput>
  <PARAM NAME=movie
VALUE="movie.swf?cfFoo=#URLEncodedFor
  mat(cfFoo)#&cfBiff=#
URLEncodedFormat(cfBiff)#">
 </cfoutput>
```

The tag containing the Flash URL is wrapped in CFOUTPUT tags. We then create the query string with two variables, cfFoo and cfBiff. Notice that we URL encode the values of the variables using the ColdFusion URLEncodedFormat function.

Here's the HTML created by ColdFusion and sent to the browser:

```
<PARAM NAME=movie
VALUE="movie.swf?cfFoo=bar&cfBiff=bim
%20bam">
```

Once the Flash movie has loaded, it will have access to two variables (cfFoo= bar, and cfBiff=bim bam) on its main timeline.

Using the Data Within Flash

Using the data within Flash is as simple as using any other ActionScript variable. Just remember that the data will be available on the main timeline of the movie (_level0._root) as soon as the movie begins to execute. All of the examples below assume that the code has been placed as a frame action on the main timeline.

Note: Don't worry if you're not familiar with Flash ActionScript. It's based on ECMAScript; thus, if you're familiar with JavaScript, then you already know the core of ActionScript.

Here's some ActionScript that checks to see that the cfFoo variable was passed in by ColdFusion:

```
if(cfFoo != null)
{
 //cfFoo was specified
}
```

The code above would exist on the main timeline of the movie. If you want to place it in a movie clip, you'd use an absolute or relative path to the cfFoo variable, such as _root.cfFoo.

This code only checks that cfFoo was passed through the URL, not whether cfFoo has a value. To check that, we check to see if cfFoo contains an empty string:

```
if(cfFoo != "")
{
 //cfFoo has a value
}
```

Once we confirm that the variable exists, we can then use it like any other Flash variable.

To simplify finding out whether a variable has been defined and has a value, we can create a function that takes a variable and returns true if it has a value and false if it doesn't or if it hasn't been defined:

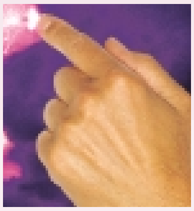
```
function hasValue(v)
{
 if(v == null || v == "")
 {
  return false;
 }
 return true;
}
```

Listing 3 is a simple example that uses the hasValue function that we just created to check whether two variables (cfFirstName, cfLastName) have been sent to Flash from ColdFusion. If the variables have been set, the code prints out their values in a dynamic text field named "fullName".

When creating your Flash movie and testing within the Flash authoring environment, you can't test the variables passed in through the URL because the Flash movie isn't called from the ColdFusion page. To get around this while developing your Flash movie, simply add some temporary variables to the main timeline of your movie while you develop within Flash.

The code in Listing 4, for example, using the example in Listing 3, allows you to develop and test your movie within the Flash authoring environment. It also allows you to isolate any errors in your Flash code. Once the Flash code is working as expected, remove the temporary variables and test your Flash movie with the ColdFusion page in a browser. At that point, if any errors occur, you know they're probably errors in the ColdFusion page.

A Couple of Points



This technique is simple and works very well; however, there are a couple of things you should watch out for when passing data to Flash from ColdFusion through the URL:

- Some browsers limit the length of query strings and thus the amount of information you can pass through a URL. Test your code thoroughly and across multiple platform/browser combinations.
- This technique works best when passing simple name/value pairs from ColdFusion to Flash.
- The technique isn't secure – anyone can view the HTML source to see the name/value pairs you're passing to Flash. Therefore, don't pass sensitive information to Flash using this technique.
- Variables passed in via the URL are passed in as String objects. If you'd like to manipulate them as a number, first convert the String to a number using one of the following ActionScript functions:

```
parseInt(String)
parseFloat(String)
```

For example:

```
cfNumber = parseInt(cfNumber);
```

This converts cfNumber from a String to an Integer.

Sending Data from Flash to ColdFusion Using getURL

We've examined a quick and easy way to dynamically send data from Cold-Fusion to Flash via the URL. Now we'll look at a quick way to send data from Flash to ColdFusion, also via the URL.

In Flash ActionScript, if you want to direct the user to another page, you can use the getURL command, which is analogous to an HTML link (<a>) tag. The syntax of the command is simple:

```
getURL(URL, target);
```

where *URL* is the URL that the Web browser will be redirected to – this is a required field. *target* is an optional argument that specifies the browser frame/name that the new page will be loaded into – the default value is "_self".

If the getURL command is called from the Flash authoring environment or a projector, then a new browser window will be opened to the URL specified in the command.

getURL issues a simple GET request to the Web server. Because of this, we can send name/value pairs from Flash to the Web server using this command. Remember that once the command is sent from the Flash movie, the browser will be redirected away from the Flash movie.

Let's put together a simple example that sends some variables from a Flash movie to a ColdFusion page using getURL and then displays the variable values in the Web browser using HTML.

We'll start by creating a Flash movie that contains a button that when pressed will make the getURL call to the ColdFusion page.

Create a new Flash movie and then place a simple button on the main timeline (you can use one of the premade buttons from Windows > Common Libraries > Buttons).

Select the button and then open the "Object Actions" panel (Window> Actions). This panel is used to associate ActionScript with button actions through the use of "on button" events. For this example we want the action to be activated when the user presses and releases the button, so we'll use the button "release" event.

```
on(release)
{
 //code here
}
```

All of the code within this statement will be processed when the button is pressed and released.

Now let's add our ActionScript code.

> **"While Flash can display and manipulate data in a browser, this capability is much more powerful when coupled with the dynamic data generating capabilities of ColdFusion"**

First we'll create two ActionScript variables to send to the ColdFusion page.

```
flFirstName = "Homer";
flLastName  = "Simpson";
```

Note that we prepend "fl" to the variable names so we know that the variables originated from Flash, which makes it easier to debug the ColdFusion code.

We'll send the variables to the ColdFusion page by appending them to the end of the URL in the getURL command calling the ColdFusion page.

```
getURL("showData.cfm?flFirstName=" +
escape(flFirstName) + "&flLastName="
+ escape(flLastName));
```

First, the "+" sign is used in ActionScript to concatenate strings. The "escape" function URL encodes its parameter and is the same as the URLEncodedFormat function in ColdFusion.

The code above dynamically creates a URL with two variables passed via the query string. Here is the URL that will be dynamically created and called when the button is pressed:

```
showData.cfm?flFirstName=Homer&flLast
Name=Simpson
```

You can use either a relative or an absolute URL, with a relative URL being relative to where the Flash movie was loaded from. If you're testing your movie from the Flash authoring environment or from a projector, you must use an absolute URL to ensure that the ColdFusion page is processed by the server.

Here is what the code attached to the button looks like:

```
on(release)
{
 flFirstName = "Homer";
 flLastName  = "Simpson";

 getURL("showData.cfm?flFirstName=" + escape(flFirstName) +
"&flLastName=" + escape(flLastName));
}
```

Save and publish the Flash movie (File > Publish).

Now all we have to do is create a simple ColdFusion page that takes the variables passed from Flash and displays them using HTML. As far as ColdFusion is concerned, the data sent from Flash is treated just as if it was sent from an HTML page. Since the data is being sent through the URL's query string, we access it within ColdFusion through the URL scope (see Listing 5).

This is an extremely simple example that takes the variables from the URL and displays them using CFOUTPUT. Of course, in a real-world application you'd most likely process the data, perhaps inserting it into a database, and then redirect the user to another page or Flash movie.

That's all it takes to send data using getURL from Flash to ColdFusion. Just remember that some browsers limit the amount of data that can be sent via the query string, and these limits also apply to Flash.

Resources

- Macromedia ColdFusion-Flash Resource Center: www.macromedia.com/software/ coldfusion/resources/flashcoldfusion
- Macromedia Flash Support Center: www.macromedia.com/support/flash
- FlashCFM – Flash/CF tutorials: www.flashcfm.com
- The single best resource available on Flash 5 ActionScript: *ActionScript: The Definitive Guide*, by Colin Moock (O'Reilly)

Conclusion

Flash 5 provides a robust client-side technology to display your data because of its browser ubiquity, ease of authoring, and similarity between ActionScript and JavaScript.

While simple, the techniques discussed in this article are the precursors of loadVariables, which allows Flash to send a request data from a ColdFusion page, and the Flash XML objects, which allow Flash and ColdFusion to exchange data in a more structured format.

About the Author

Mike Chambers, Macromedia's Flash community manager, has been creating applications using Flash, Generator, and middleware for the past three years. A coauthor of Generator and Flash Demystified, *Mike has experience working with Java, ASP, JSP, PHP, and ColdFusion.*

mesh@macromedia.com

WHAT'S NEW in CF STUDIO 5?
COLDFUSION Developer's Journal
E-Testing
Debugging Your Projects
When running the code isn't sufficient
Foundations: Defending Socrates
Where to Go from Here

Listing 1

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-
444553540000"
codebase=http://download.macromedia.com/pub/shockwave/cabs/
flash/swflash.cab#version=5,0,0,0 WIDTH=550 HEIGHT=400>

  <PARAM NAME=movie VALUE="movie.swf">
  <PARAM NAME=quality VALUE=high>
  <PARAM NAME=bgcolor VALUE=#FFFFFF>

  <EMBED src="movie.swf" quality=high bgcolor=#FFFFFF
WIDTH=550 HEIGHT=400 TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/shockwave/download/
index.cgi?P1_Prod_Version=ShockwaveFlash">
  </EMBED>

</OBJECT>
```

Listing 2

```
<cfset cfFoo="bar" />
<cfset cfBiff="bim bam" />

<html>
<body>

<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-
444553540000"
codebase=http://download.macromedia.com/pub/shockwave/cabs/
flash/swflash.cab#version=5,0,0,0 WIDTH=550 HEIGHT=400>

 <cfoutput>
  <PARAM NAME=movie
VALUE="movie.swf?cfFoo=#URLEncodedFormat(cfFoo)#&cfBiff=#
URLEncodedFormat(cfBiff)#">
 </cfoutput>

  <PARAM NAME=quality VALUE=high>
  <PARAM NAME=bgcolor VALUE=#FFFFFF>

 <cfoutput>
  <EMBED
src="movie.swf?cfFoo=#URLEncodedFormat(cfFoo)#&cfBiff=#
URLEncodedFormat(cfBiff)#"> quality=high bgcolor=#FFFFFF
WIDTH=550 HEIGHT=400 TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/shockwave/download/
index.cgi?P1_Prod_Version=ShockwaveFlash">
  </EMBED>
 </cfoutput>
```

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

Listing 3

```
if(hasValue(cfFirstName) && hasValue(cfLastName))
{
 //fullName points to a dynamic text field and will thus
 //be displayed to the user.
 fullName = cfFirstName + " " + cfLastName;
}
else
{
 //one or both of the variables were not defined,
}
```

Listing 4

```
//temp testing variables
var cfFirstName = "Homer";
var cfLastName = "Simpson";

if(hasValue(cfFirstName) && hasValue(cfLastName))
{
 //name points to a dynamic text field
 name = cfFirstName + " " + cfLastName;
}
else
{
 //one or both of the variables were not defined,
}
```

Listing 5

```
<cfset flFirstName = #URL.flFirstName# />
<cfset flLastName = #URL.flLastName# />

<html>
<body>

<cfoutput>

 <p>The first name is : <b>#flFirstName#</b></p>
 <p>The last name is : <b>#flLastName#</b></p>

</cfoutput>

</body>
</html>
```

CODE LISTING

The code listing for this article is also located at www.sys-con.com/coldfusion/sourcec.cfm

Introducing **Neo**

The entire Java universe will be accessible to CFers

BY **BEN FORTA**

At last month's Developers' Conference in Orlando I was fortunate to be one of the first to demonstrate Neo, the next major version of ColdFusion. Macromedia (and earlier, Allaire) had been dropping little Neo-related tidbits for quite a while now – whetting our appetites with glimpses of what is to come. But as those of you who were in Orlando now know, Neo is exceeding all expectations.

So for those of you who weren't there (don't make that mistake next year), and as a reminder for those who were, here are some of the highlights.

Access to the Java Universe

Simple, accessible, fast, fun. All adjectives used to describe ColdFusion and CFML. And all adjectives that I've yet to hear anyone use when describing Java. That's not to say that Java has no value. It does. Java is powerful, portable, extensible, widely supported, respected (occasionally a prerequisite), and capable.

ColdFusion's strength is simplicity; Java's strength is raw power. Each is valuable, but combine the two and you end up with more than just the sum of their parts. ColdFusion is ideally suited for scripting user interfaces, database integration, and Web interaction. Java is ideally suited for back-end processing, heavy lifting, and component creation. As I have discussed in this column before, Java and CF are a match made in e-heaven.

For CF developers, this is one of the most compelling aspects of the Neo story. In Neo the entire Java universe is accessible to CFers, all with the simplicity that you've come to expect from ColdFusion. The new <CFIMPORT> tag makes it possible to interact with Java code right from within your CFML code, and custom tag abstractions make it possible to encapsulate even that simple interface. Leveraging Java back ends in Neo is easy, even easier than in straight Java.

Of course, as Neo is ColdFusion, all this extra power comes at close to no cost – you'll still write CFML; you'll still create CFM files; you'll still use <CFQUERY>, <CFOUTPUT>, and <CFINCLUDE>; and you'll still write apps as you do now.

As for Java, all those three-letter acronyms (most beginning with *J*), all the tags and components (the volume of which dwarfs what is available for ColdFusion), products, utilities, and APIs – all that is optional. But when you are ready for it, Neo makes it all amazingly accessible.

Localization and Internationalization

One of the most common criticisms of ColdFusion is its lack of Unicode and support for true localization and internationalization. As I demonstrated in Orlando, Neo addresses this issue once and for all. Neo is built on top of Java, which supports Unicode and internationalization as a core capability. As such, Neo inherits this support, making it accessible to ColdFusion developers.

Tags, functions, your own code – they can all be localized and globalized as needed, and all with minimal effort.

Application Isolation

Another compelling benefit, also the result of Neo's being Java-based, is application isolation. Simply put, in the Neo world, applications can run in their own virtual spaces – almost their own application servers.

Why is this important? Most ColdFusion applications run on shared boxes. Prior to Neo there was no way to restart individual applications, no way to terminate rogue applications without impacting others, and no way to implement application-specific settings.

The specifics of how this feature will work, and exactly what it will enable, have not yet been publicized. But one thing is definite: application isolation will make servers and applications more reliable, more manageable, and more stable than ever before.

Compiled Code

ColdFusion 5 (and earlier) is essentially an interpreter – CFM files are read and then compiled into *p*-code in memory. The CFM files must always be present; there is no way to save or distribute compiled code.

Neo changes this. In Neo, CFM files are compiled into Java .class files that can be executed, deployed, and even distributed. Compiled applications will need a Neo license to run. Exactly how this feature will be made accessible has yet to be announced, but regardless, the fact that ColdFusion executes compiled code (instead of interpreting code) means that your apps will run faster, your source code will be more secure, and additional deployment options will be possible.

A Whole New Level

It's official: Neo is the next major version of ColdFusion, it's built on Java, and it promises to take CF development to a whole new level. All of the features and capabilities mentioned here were announced at the Developers' Conference at the first General Session. And all of these features and capabilities are the direct result of a significant architectural investment that we've been working on for a long time now – porting the core ColdFusion engine to Java. It's been a long journey, but the results are proving that it was a worthwhile one to make. Visit www.macromedia.com for more information.

ABOUT THE AUTHOR

Ben Forta is Macromedia's senior product evangelist and the author of numerous books including the recently published ColdFusion 5 Web Application Construction Kit *and its sequel,* Advanced ColdFusion 5 Development. *For more information on Ben's books visit www.forta.com.*

BEN@FORTA.COM

Missed an issue?

We've got 'em all for you on CD!

JAVA DEVELOPER'S JOURNAL

The most complete library of exclusive *JDJ* articles on one CD!

Check out over 500 articles covering topics such as...
Java Fundamentals, Advanced Java, Object Orientation, Java Applets, AWT, Swing, Threads, JavaBeans, Java & Databases, Security, Client/Server, Java Servlets, Server Side, Enterprise Java, Java Native Interface, CORBA, Libraries, Embedded Java, XML, Wireless, IDEs, and much more!

***JDJ* The Complete Works**
~~Reg $119.99~~
Buy Online
Only \$71^{99}

XML JOURNAL

The most complete library of exclusive *XML-J* articles on one CD!

Check out over 150 articles covering topics such as...
XML in Transit, XML B2B, Java & XML, The XML Files, XML & WML, Voice XML, SYS-CON Radio, XML & XSLT, XML & XSL, XML & XHTML, 2B or Not 2B, XML Industry Insider, XML Script, <e-BizML>, XML & Business, XML Demystified, XML & E-Commerce, XML Middleware, and much more!

***XML-J* The Complete Works**
~~Reg $59.99~~
Buy Online
Only \$53^{99}

COLDFUSION Developer's Journal

The most complete library of exclusive *CFDJ* articles!

Check out over 250 articles covering topics such as...
Custom Tags, ColdFusion and Java, Finding a Web Host, Conference Reports, Server Stability, Site Performance, SYS-CON Radio, ColdFusion Tips and Techniques, Using XML and XSLT with ColdFusion, Fusebox, Building E-Business Apps, Applicaton Frameworks, Error Handling, and more!

SPECIAL OFFER:
Buy CFDJ & CFA
The Complete Works
For Only
\$129^{99}

***CFDJ* The Complete Works**
~~Reg $79.99~~
Buy Online
Only \$71^{99}

CF Advisor

The most complete library of exclusive *CFA* articles!

Check out over 200 articles covering topics such as...
E-Commerce, Interviews, Custom Tags, Fusebox, Editorials, Databases, News, CF & Java, CFBasics, Reviews, Scalability, Enterprise CF, CF Applications, CF Tips & Techniques, Programming Techniques, Forms, Object-Oriented CF, WDDX, Upgrading CF, Programming Tips, Wireless, Verity, Source Code, and more!

***CFA* The Complete Works**
~~Reg $79.99~~
Buy Online
Only \$71^{99}

JDJStore.com

Order Online and Save 10% or More!

www.JDJSTORE.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

COLLECT ALL 4
FOR ONLY \$229^{99}
JDJ, XML-J
CFDJ & CFA

CommonSpot Content Server v2.5 from PaperThin, Inc.

REVIEWED BY
DAVE HORAN

Content management is fast becoming the latest buzzword for mid- to large-scale sites. If you've worked on such a site, you can understand why.

Just as Web-wide search engines have evolved to match the needs of an ever-expanding user and content base, so have the tools used to create and manage the sites a user can access.

Enter CommonSpot

The latest v2.5 release of CommonSpot Content Server from PaperThin, Inc., brings all of the content creation and management tools from v2.0, but adds several features to keep pace with the demands of site content managers eager to enhance the offerings of their site. Some new features also lighten the burden on ColdFusion developers by allowing page authors to tackle certain tasks on their own that were previously assigned to programmers, such as forms that submit their results to an e-mail address.

This product has far too many features to outline here. (See the Resource section for additional information.) Suffice it to say that it has features that any content-heavy site desires: multiple contributors, workflow and multilevel approval process, version history with rollback, personalization, roles-based security, and a host of others. For this review, I'll focus on the new features that not only make this release an improvement over the previous, but also put it head and shoulders above others in its class.

Test Environments

First off: yes, it does run on ColdFusion v5 – quite well, in fact! Further, being a CF application, the product supports all platforms CF does. At the time of this writing, it supports Microsoft Access, Microsoft SQL, and Oracle as the back-end databases.

I have used it on single processor, 128MB RAM systems up to a dual CPU 800MHz 1GB memory system. Due to the product's page caching mechanism, the real processing horsepower is required for the page authoring processes and full-text indexing, not for normal page views. If you're going to spec a server for this product, I would recommend at least a single 600MHz CPU with 256MB RAM, more if you plan on integrating many custom ColdFusion apps.

Regarding ColdFusion 5.0, although it runs well on CF5, it doesn't yet take advantage of many of the new features, such as CFGRAPH and the new integrated Verity K2 engine. CommonSpot does make extensive use of memory, hence running on CF5 can make a big difference in performance and memory utilization.

Base Features

Many existing features have been enhanced directly or indirectly through the improvements in this release. Here are a few.

Custom CF Integration

Where the canned CommonSpot elements don't fit our needs, we've been able to create our own using the Custom ColdFusion Element control. This allows a page designer to place a CF placeholder on a page where an application or CF module should be. The developer can then go and point the Custom ColdFusion element to the code it should run; the resultant HTML output is then placed on the page.

VITALS

CommonSpot Content Server v2.5
Paperthin, Inc

Address: 267 King Caesar Rd
Duxbury, MA 02332

Web: www.paperthin.com

E-mail: info@paperthin.com

Test Environment:
P111 800MHz CPU, 256MB memory, Windows 2000, MS-SQL Server 2000, ColdFusion v5.01, MS Internet Explorer v6.x

Pricing:
Starts at $20,500
See www.paperthin.com/products/pricing.cfm for details

Through this element developers have access to many structures of data related to the current state of the system, user, current page, and page elements. Using this element not only lets you create small functions to display "Welcome Dave" personalization on a page, but also lets you incorporate an entire application. Note to Fusebox developers: CommonSpot is Fusebox-friendly in this regard.

Customization Options

Many additional "hooks" have been provided in v2.5 that allow the developer to alter the look, feel, or behavior of key site elements. For example, by including a single over-

wirelessEDGE conference&expo

INTERNATIONAL WIRELESS BUSINESS & TECHNOLOGY CONFERENCE & EXPO

CONFERENCE & EXPO

CONFERENCE & EXPO

Plan to Exhibit

Provide the Resources To Implement Wireless Strategy

The conference will motivate and educate. The expo is where attendees will want to turn ideas into reality. Be ready to offer solutions.

Shaping Wireless Strategy for the Enterprise

Santa Clara, CA | **May 7-9, 2002**

WirelessEdge will provide the depth and breadth of education and product resources to allow companies to shape and implement their wireless strategy. Developers, i-technology professionals and IT/IS management will eagerly attend.

Plan to Attend the 3-DAY Conference

WHO SHOULD ATTEND

Mobile & Wireless Application Professionals who are driving their enterprise's wireless initiatives:

- Program Developers
- Development Managers
- Project Managers
- Project Leaders
- Network Managers
- Senior IT and Business Executives

Conference Tracks

Track One: Development
- WAP
- i-Mode
- Bluetooth / 802.11
- Short Messaging
- Interactive Gaming
- GPS / Location-Based
- Wireless Java
- XML & Wireless Technologies

Track Two: Connectivity
- Smart Cards
- Wireless LANs incl. Bluetooth
- UMTS/3G Networks
- Satellite Broadband

Track Three: Wireless Apps
- Education
- Health Care
- Entertainment
- Transport
- Financial Services
- Supply Chain Management

Track Four: Hardware
- Cell Phones/ WorldPhones
- PDAs
- Headphones/ Keyboards / Peripherals
- Transmitters/ Base Stations
- Tablets

Track Five: Business Futures
- Wireless in Vertical Industries
- The WWWW
- Unwired Management
- From 3W to 4W: Issues and Trends
- "Always-On" Management
- Exploiting the Bandwidth Edge
- Unplugged Valueware
- Wireless Sales & Marketing

FOR INFORMATION CALL

201 802-3069

SPEAKER PROPOSALS INVITED

WWW.SYS-CON.COM

SHAPE YOUR WIRELESS STRATEGY SAVE THE DATES!

EXCLUSIVE SPONSORSHIPS AVAILABLE

Rise above the noise. Establish your company as a market leader. Deliver your message with the marketing support of

SYS-CON MEDIA

SYS-CON EVENTS

THE LARGEST WEST COAST WIRELESS

BUSINESS & TECHNOLOGY CONFERENCE OF THE YEAR!

ride text file, the site login dialog can be customized to display or hide certain buttons or to alter the field label text. Other hooks and configuration options can control things such as search forms, content authoring menus, and page BODY tag "onload" handlers.

> "From a developer perspective, one of the greatest enhancements to the product is the addition of page element, custom-rendering handlers"

Personalization

With its user/groups-based security system, CommonSpot makes personalization easy. Scheduled page elements (those that change based on time/context/user conditions) can be tied to specific users or groups and allow the site to be personalized based on the fact that the user is a member of a specific group. Also available to developers through session variable structures and programmatic hooks are the current user information and current page information. This includes values for admin-defined page metadata fields as described in the next section.

Browser-Based

The entire system works in an almost exclusively browser-based fashion. All administration, template creation, page creation, and content editing can be done via a browser. At this time, administration and editing must be done from IE 4.x+. In practice, however, much of the template creation required the use of an HTML editor, such as CFStudio, to insert required code.

Template Inheritance

The template inheritance functionality allows for a "tree" of templates to be created, with each template tracing its ancestry back to a single ColdFusion template. If a change is made in the parent template and the site page cache is cleared, the child pages will reflect the change. Although this feature hasn't changed from earlier versions, v2.5 does allow you to alter the parent of an existing page. No more re-creating pages!

Integrated Content Security

If you've had any experience with standard users/groups-based, operating system security schemes, then you'll have an easy time adjusting to the CommonSpot system. The content security provides you with fairly granular control over who has access to what functions on a given subsite, page, or element. It's a flexible system with the potential for an equal measure of complexity.

New Features in Version 2.5

There are many brand-new features in this version; here are some of interest to developers.

Custom Authentication

One extremely powerful feature of v2.5 is custom authentication. In most cases you'll be given a data source with users, passwords, etc., to use for authentication purposes. CommonSpot allows you to authenticate against this outside database, and, with some simple queries, pull user data for use in site personalization.

Simple Forms Support

This function allows the content contributor to design and implement simple data-entry forms. You have little control over the layout and complex programmatic field validation isn't supported; however, it does validate some simple types such as e-mail address and phone. This great control lets users create form-to-e-mail and form-to-data table pages (e.g., guest book) with ease – and you don't have to code them!

Datasheet Elements

This new feature seems to be one of the most underutilized functions of the new release. It allows for the tabular display of data collected from user-defined simple forms or from custom queries. It can also be customized and extended to function as a simple drill-down application against your own database. With developers in mind, authors can define their own SQL statements to run against almost any data source on the system. The exception to this is the CommonSpot users or sites' data sources; these are reserved for internal system use only. *Note:* These data sources can be accessed using custom ColdFusion elements.

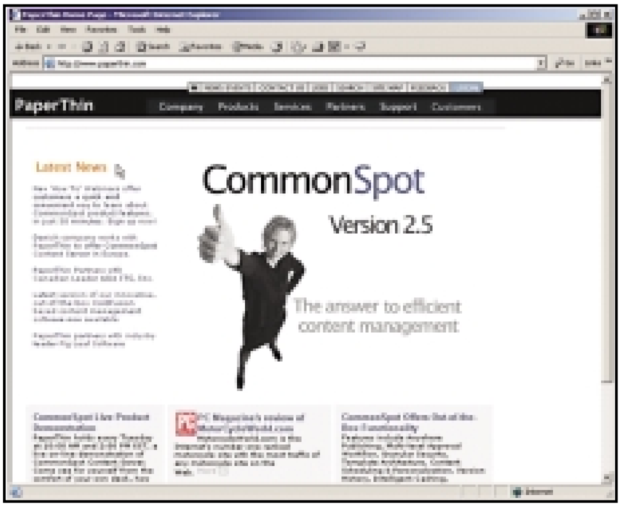
Task Management

As sites become larger and the teams managing those sites become equally large, the job of managing the work involved can be overwhelming. To help keep everyone's sanity, CommonSpot includes a new task-management feature. This allows administrators to define roles, tasks, and users to create a small project management system. Yes, I said project management. Before the managers reading this get all excited, this is not a full-fledged PM system. Rather this is more of a peer-to-peer request tracking system. It simply allows team members to make work requests for things out of their realm of responsibility and provides a simple way to track the progress of these requests.

This feature is separate from the Work Request feature of pre-2.5 releases. This can be a bit confusing. PaperThin is working to combine these functions in future releases to make it a more cohesive and streamlined tool.

Advanced Metadata Handling

Another extremely powerful feature in v2.5 is the addition of page-level metadata. Metadata can be captured based on the template a page is based on, the subsite the page resides in, and so on. These admin-defined metadata forms are defined via the browser and can include standard form elements as well as predefined menu tree controls. Data entered in these forms can be accessed programmatically or through certain page elements.



CommonSpot version 2.5

For instance, an article page can store the article abstract in the metadata while the page itself contains the article content. This abstract can be included in an article index page, but still associated with the main page.

Custom Rendering Handlers

From a developer perspective, one of the greatest enhancements to the product is the addition of page element, custom-rendering handlers. At a minimum, these allow you to control the look and feel of a "canned" CommonSpot element. When used in a page, rather than displaying the predefined HTML for an element, the system passes your handler several structures of data containing the content for the element. This allows you to redefine the element as you see fit, passing your presentation of the data to the system for rendering.

Page Sets

This new page set feature is a fantastic addition, especially for sites with articles that span multiple pages. The page set control lets you manage a set of pages under one name and easily add navigation controls between the members of the set.

Warrants a Second Look or Upgrade

CommonSpot v2.5 provides enough new features and improved documentation to warrant an upgrade if you're a current user or a second look at the product if you discounted it before. With clustered server support and back-end database support for Microsoft SQL Server and Oracle, the system can certainly scale to meet your needs.

Resources

1. www.paperthin.com
2. www.content-wire.com/Home/Index.cfm?ccs=86&cs=380

ABOUT THE AUTHOR

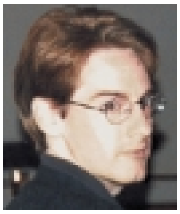
Dave Horan manages the Rochester, New York, ColdFusion Users Group and has been in the IT field for more than 15 years.

DHORAN@ROCHESTER.RR.COM

A Cold Cup O'Joe Part 6 of 8

Java CFX debugging

BY GUY RISH



With Java's ever-broadening list of class APIs, your CFX has few limitations.

However, shortcomings in the CFX model could make developing solutions a little tricky. Since the ColdFusion server bootstraps CFXs, debugging them poses certain challenges.

In Part 5 of this series (***CFDJ***, Vol. 3, issue 8) I introduced the basics of writing CFX tags in Java; this merely scratched the surface though. A simple "Hello, World" tag, even one driven by a query such as the second example I provided in Part 5 (see sidebar), is hardly sufficient to show the details of what can be done with a Java CFX tag.

As promised, in Part 6 I delve further into the CFX model by demonstrating some of the Macromedia-recommended debugging techniques, and talk about how you can do things in a Java CFX that would be a little more difficult in C++.

Check Your Tools

Of the 14 classes that comprise the Java CFAPI class library, I discussed only four of them in Part 5. These four classes form the basis of the majority of the interactions needed for working with ColdFusion's CFX interface (see Figure 1).

The CustomTag class is the foundation of every Java CFX tag. The Request class contains all the incoming parameters of the tag, and an instance of it is passed to the CFX tag entry point. The Response class enables data to be sent back to the classing logic, and an instance of it is passed to the CFX tag entry point. The Query class allows you to manipulate any ColdFusion query passed into the CFX tag. These four classes are fairly robust and you can accomplish a broad number of things with them.

Check the Printout

Print-to-screen is a tried-and-true debugging tool. Never elegant or clever, a language's print-to-screen commands are still the cornerstone of debugging. It's a minor challenge using this staple of software development in a CFX; content sent to System.out, the typical print-to-screen destination, doesn't enter the same output stream that the ColdFusion Server sends back to the Web server. However, you can manage this two ways through the Response object that the ColdFusion Server passes to your CFX when it gets called.

write

The write method of the Response class is used to return formatted output to the ColdFusion Server's output stream. Typically for the construction of an HTML user interface, write can also be used to display debugging information.

This technique, like all uses of the print-to-screen method, has a serious drawback. Not only does it disturb your screen output, but once you're finished, you must remove all calls to write that are used for debugging.

Using an update of the tried-and-true "Hello, World" example from Part 5, HelloWorld1.java (see Listing 1) shows an example of this. The results can be observed by simply executing the tag in a ColdFusion template (see Listing 2).

I registered the CFX as "cfx_helloworld" in the ColdFusion Administrator as discussed in Part 5. We'll be making a few different iterations to the "Hello, World" example throughout this article, and each time I'll update the "class file" setting of the same tag registration.

writeDebug

The writeDebug method of the Response class is a slight improvement. It still relies on the print-to-screen concept, but only under a specific circumstance. When the CFX is invoked with the DEBUG attribute set to "on," it will trip an internal flag. The writeDebug method will print only if this flag is tripped.

Replacing all the write methods with writeDebug methods will give us a CFX as seen in HelloWorld2.java (see Listing 3). Don't forget to update the CFX's "class file" setting in the ColdFusion Administrator. In a ColdFusion template (see Listing 4) I've made two calls to the newer version of our CFX, one with the DEBUG attribute set to "on" and the other with the attribute set to "off." Notice the differing results.

This is a little better, but again it's still messy.

Faking It

What if there was a way to run a custom tag from the command line? What if, instead of the ColdFusion Server instantiating the CFX, you could do it? Macromedia provides for just such a thing with the help of three classes: DebugRequest, DebugResponse, and DebugQuery. These debugging versions allow you to mimic the way the ColdFusion Server might interact with the CFX so you can test it right from the command line. This functionality is unique to Java CFXs, and while similar things can be rigged with their C++ cousins, Macromedia didn't provide a prepackaged solution for them.

The Main Thing

The code in processRequest doesn't change, but I've added a new

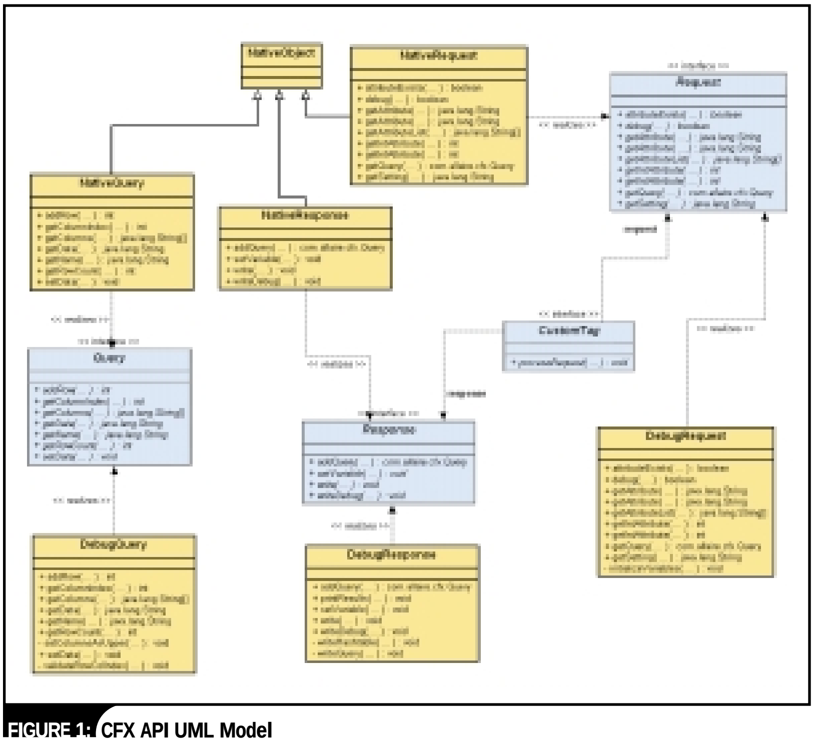


FIGURE 1: CFX API UML Model

method to the class, main. This new iteration of the CFX, HelloWorld3.java (see Listing 5), creates its own debugging versions of Request and Response and passes them to an instance of itself (see lines 1–18).

Using a Java Hashtable, I created the arguments I want passed to the tag instance (lines 8 and 9). I use this Hashtable in the constructor of the DebugRequest instance (line 11). Neither the DebugResponse instance (line 12) nor the tag (line 14) requires construction. The tag's real functionality is invoked (line 16) when processRequest is called. Then it's a simple matter to display the results via the DebugResponse class's printResults (line 18), which displays directly to System.out.

Greet Them, One and All

To demonstrate the use of the DebugQuery class I'll use the HelloQuery CFX from Part 5. This tag receives a Query to loop over as the source of its input, nothing overly complex (see sidebar). I made some adjustments to HelloQuery.java (see Listing 6) and it can now be compiled and called from the command line.

I built a hash table of attributes to pass to processRequest, just as before (lines 10 and 11). Notice that I actually add the QUERY attribute and give it a value of "names". The value is unimportant, but it's vital that the QUERY attribute exists.

If you recall from Part 5, to pass a CFML Query to a CFX it had to be passed via the QUERY attribute. The HelloQuery CFX actually tests for this attribute before retrieving the Query object with the Request's getQuery. Because it's possible through the debug versions to pass the Query without using the QUERY attribute, it's vital to create a QUERY attribute so as not to disrupt the current functionality of the tag. At first this might seem silly. If the tag is designed to receive a Query object, why check for the QUERY attribute at all? The answer to that would be: What if the QUERY attribute were being used to pass optional data? This leaves the door open for increased flexibility, especially when this flexibility can be used to overcome the data-passing limitations imposed by the CFX model. In this way nothing is assumed; the code always tries to check itself.

To create a DebugQuery instance two String arrays are needed. One array contains the names of the columns (line 14), the other is a multidimensional array, an array of column values within an array that comprises the rows (lines 15-18). An instance can then be created (line 20) and passed to the DebugRequest constructor (line 22).

The rest is the same as in the previous example.

Debug Class Dividends

All in all, using the debugging classes is not a very complicated activity. The real dividend is that you can run it from the command line, but because of that you can use traditional debuggers to step through the code as it executes. Just when you thought it was going to get tough…

Under Observation

It would hardly seem decent to write an article that contains little more than what can be found in the documentation already. Sure, I've covered a few extra little twists, but nothing you couldn't have gleaned with careful reading and a little experimentation. So I've brought something more to the table: leveraging some of the technology I used in the JVMLog class I introduced in Part 4 (***CFDJ***, Vol. 3, issue 6), you can take your debugging one step further.

The JVMLog class (reversibly) hijacks the JVM's standard I/O PrintStream instance. It does so with another simple class that I created called ObservablePrintStream. The ObservablePrintStream is a subclass of the normal PrintStream used by the JVM, so my class could polymorphically stand in for the original. The interesting thing about ObservablePrintStream is that it implements the Observer design pattern.

I was unable to use Java's standard Observable class since my PrintStream subclass would have had to inherit from something other than PrintStream – thus completely ruling out my ability to hijack the JVM's PrintStream in the first place.

The point of the Observer design pattern is to set up a relationship of objects that can communicate with each other in a publish/subscribe fashion. An object interested in getting notified whenever something happens "subscribes" to the object in which it is interested. Specifically, anything that's intended for the screen via System.out will also get sent to the subscribing objects.

What I have done is create a set of templates and Java classes that will allow you to install the ObservablePrintStream to the JVM and then "subscribe" objects so you can monitor things printed to the JVM's System.out instances.

Using the ColdFusion template ManageObserver.cfm (see Listing 7) will allow you to toggle the hijacking of the JVM's PrintStream. (Listings 7–10 can be found on the ***CFDJ*** Web site, www.sys-con.com/coldfusion/sourcec.cfm.) Using another template, ManageNetObserver.cfm (see Listing 8), you can toggle the installation of a socketed observer. Then, using either the client I created, Monitor.java (see Listing 9), or a telnet application, you can connect to port 2525 (configurable from ManageNet-Observer.cfm) and watch for anything written to the JVM's System.out to be displayed. To this end I've created a new version of the "Hello, World" CFX, HelloWorld4.java (see Listing 10), where I've done a search and replace on all the writeDebug calls to change them into println calls off System.out.

Wrapping It Up

There are a number of challenges when it comes time to debug your Java extensions, but they don't need to be insurmountable. Using the built-in tools provided by Macromedia you stand a good chance of solving most any debugging problem.

Reusable Tools

When I created the Observable-PrintStream I knew it would be a useful tool. I've gotten considerable mileage out of it in more than just my work with ColdFusion. I've no doubt that this simple tool could be leveraged in ways that I have yet to foresee.

GUYRISH30@YAHOO.COM

It seems I have an apology to make. There was an error in the source HelloQuery.java in Part 5. The for-loop on line 35 is incorrect and will cause the tag to fail at runtime with a Java IndexOutOf-BoundsException. That line should actually read:

```
 for(nRowIndex = 1;
nRowIndex <= nRowCount;
nRowIndex++)
```

Notice that the loop index starts at 1. It seems that a Query is base 1, so it was necessary to adjust the starting position and the terminating condition for the code to function correctly. I've no excuse for this error in judgment. I can only throw myself at your mercy.

ABOUT THE AUTHOR

Guy Rish works for DevX, an online IT resource center. He holds instructor certifications from Rational Software and Allaire and teaches Web development for SFSU MSP. Guy is active in the ColdFusion and OO communities. He was the technical editor for Mastering ColdFusion 5 from Sybex.

Listing 1

```
import com.allaire.cfx.*;

public class HelloWorld1 implements CustomTag
{
 // custom tag entry point
 public void processRequest(Request request,
Response response)
 {
  String strName = null;
  String strGreeting  = null;

  //

response.write((request.getAttributeList()).toStrin
g());

  // retrieve/default the incoming argument
  strName = request.getAttribute("NAME", "World");

  strGreeting = "Hello, " + strName;

  // write it back to the display stream
  response.write(strGreeting);
 }
}
```

Listing 2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">

<html>
 <head>
  <title>Hello, World CFX</title>
 </head>

 <body>

  <cfx_helloworld name="Rish">

 </body>
</html>
```

Listing 3

```
import com.allaire.cfx.*;
import java.util.Hashtable;

public class HelloWorld2 implements CustomTag
{
 // custom tag entry point
 public void processRequest(Request request,
Response response)
 {
  String strName = null;
  String strGreeting  = null;

  // retrieve/default the incoming argument
  strName = request.getAttribute("NAME", "World");

  strGreeting = "Hello, " + strName;

  // write it back to the display stream
  response.write(strGreeting);
 }
}
```

Listing 4

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">

<html>
 <head>
  <title>Hello, World CFX</title>
 </head>

 <body>

  <cfx_helloworld name="Rish">

  <cfx_helloworld name="Rish" debug="on">

 </body>
</html
```

Listing 5

```
import com.allaire.cfx.*;
import java.util.Hashtable;

public class HelloWorld3 implements CustomTag
{
 public static void main(String[] args)
 {
  HelloWorld3 tag = null;
  Hashtable attributes = null;
  DebugRequest drequest = null;
  DebugResponse dresponse = null;

  attributes = new Hashtable();
  attributes.put("NAME", "Rish");

   drequest = new DebugRequest(attributes);
   dresponse = new DebugResponse();

   tag = new HelloWorld3();

   tag.processRequest(drequest, dresponse);

  dresponse.printResults();
 }

 // custom tag entry point
 public void processRequest(Request request, Response
response)
 {
  String strName = null;
  String strGreeting  = null;

  // retrieve/default the incoming argument
  strName = request.getAttribute("NAME", "World");

  strGreeting = "Hello, " + strName;

  // write it back to the display stream
  response.write(strGreeting);
 }
}
```

Listing 6

```
// import the Java CFAPI
import com.allaire.cfx.* ;

import java.util.Hashtable;

public class HelloQuery implements CustomTag
{
 public static void main(String[] args)
 throws Exception
 {
  HelloQuery tag = null;
  Hashtable attributes = null;
  DebugQuery query = null;
  DebugRequest drequest = null;
  DebugResponse dresponse = null;

  attributes = new Hashtable();
  attributes.put("QUERY", "names");

  // build query
  String[] columns = { "NAME" };
  String[][] data =  {
      { "Guy Rish" },
      { "Nathan Dintenfass" },
      { "Charles Arehart" } };

  query = new DebugQuery("names", columns, data);

      drequest = new DebugRequest(attributes, query);
      dresponse = new DebugResponse();

  tag = new HelloQuery();

  tag.processRequest(drequest, dresponse);

  dresponse.printResults();
 }
// custom tag entry point
 public void processRequest(Request request, Response
response)
 throws Exception
 {
  String[] attributes = null;
  int nRowCount, nRowIndex, nColumnIndex;
  Query qry = null;

  if(request.attributeExists("QUERY"))
  {
   qry = request.getQuery();
  }
  else
  {
   throw new Exception("<h2>Error!</h2>No Query attribute
passed.");
  }

  if(qry == null)
  {
   throw new Exception("<h2>Error!</h2>Unable to retrieve
the query.");
  }

  nRowCount = qry.getRowCount();
  nColumnIndex = qry.getColumnIndex("NAME");

  for(nRowIndex = 1; nRowIndex <= nRowCount; nRowIndex++)
  {
   response.write("Hello, " + qry.getData(nRowIndex,
     nColumnIndex) + "<br>");
  }
 }
}
```

CODE LISTING

The code listing for this article is also located at

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

Ask the **Training Staff**

A source for your CF-related questions

BY
BRUCE
VAN HORN

Thanks to all of you for reading this column every month and for sending in questions. Whether or not they get printed, keep 'em coming! I enjoy the interaction and trying to help solve your CF problems.

This month I have room to address only two questions. Both of them, I think, are important to every CF developer out there. So here we go!

Q: ***You mentioned in your FastTrack to CF class how important it is to lock all references to Server, Application, and Session variables, but you also hinted that there was an alternative to doing this on every page. Could you share that alternative with me?***

A: Absolutely! This subject is covered in detail in Macromedia's Advanced CF class (another shameless plug for you to take these classes if you haven't already!). Let me review the issue before jumping into the solution. As you know (or should know), a variable that is set into the Server, Application, or Session scope is what we call a *shared* variable. This means that it's assigned a specific slot in the server's memory when it's created and any subsequent reads or writes to that variable are made to that same spot in memory. The problem comes when multiple users access the same pages at the same time. To avoid corrupting the data (or worse) by allowing simultaneous access to a shared variable, we need to place a lock on that variable so only one user accesses it at a time. This can be unwieldy because you need to lock every reference (reads or writes) to those variables. Furthermore, many of us have written apps that are already in production and that use Application and Session variables without any locking. Going back to change these apps can be time-consuming.

The solution offered in the Advanced CF class is easy to understand and, more important, easy to implement. The logic is this: Since shared variables need to be locked but local variables don't, why not copy all our shared variables into local variables at the beginning of each page, reference them as local variables in our code, then copy them back to shared variables at the end of the page to capture any changes? This method makes it easier for you as a developer because you don't have to worry about forgetting to lock a reference to a shared variable. Also, it can improve the performance of your application because you aren't opening and closing locks multiple times on each page for each user.

Here are the steps to freedom: first, place some code into your Application.cfm file, which will execute at the beginning of every page requested (see Listing 1). This code creates an exclusive lock for each variable scope (if you don't use Server variables, just omit this block). Inside the lock initialize a structure called “Data” in the shared scope, then copy any existing data from that scope into an appropriately named structure in the request scope (i.e., Request.Session).

Now that you've copied all the data from the shared scopes into their equivalent Request scoped containers, replace every reference to a shared variable with a reference to its Request scope equivalent. For example, all references to “session.userid” become references to “request.session.userid”. That's it! Just add “Request.” to the beginning of each variable reference. Since the Request scope is a local variable, no locks are needed.

The last step in the process is to make sure you capture any changes you may have made to these local variables and copy them back into their appropriate shared scope so the next page request is aware of the changes. Remember, the file OnRequestEnd.cfm is the same as Application.cfm except that it executes at the end of every page. Listing 2 shows the code you need to place in OnRequestEnd.cfm to copy the code back into the appropriate scopes. These few lines of code in Application.cfm and OnRequestEnd.cfm can virtually eliminate your shared-variable locking worries.

A problem with this code, however, is that if you set a variable into a shared scope and then do a CFLOCATION to another page, CFLOCATION prevents the code in OnRequestEnd.cfm from executing on that page. In these scenarios set your variable directly into the “Data” structure of the shared variable with a traditional CFLOCK around it, then do your CFLOCATION (see Listing 3).

Q: ***Sometimes I need to restart my server or make changes to application pages but I'm worried about killing the processes of active users. I want to be able to know how many users are currently using my applications so I can see if it's safe to restart the server or if I should wait until there aren't any active users. Is there an easy way to do this?***

A: Yes! This subject was covered in ***CFDJ*** by Christian Schneider in “Live Monitoring of User Sessions” (Vol. 2, issue 8) and I've used his concept on my own servers. His idea is to

> “I like to use a query object instead of a structure now that CF5.0 has the ability to run queries from existing queries”

create a structure in the Application scope (or Server scope if you want to monitor more than one application) that holds users’ IP number and a timestamp of their last request. Then you just build a report that loops over that structure to show you the data.

My suggestion is basically the same, but I like to use a query object instead of a structure now that CF5.0 has the ability to run queries from existing queries. Listing 4 contains the code for this, but I’ll walk you through the concepts of collecting and displaying the data.

First, create a custom tag (mine is called CF_TrackUsers) that builds a query in the Server scope. You can store as much or as little information about each user as you want. I recommend keeping it simple by storing only some data that uniquely identifies each user (IP number or a unique session ID), the name of the application being used, and the time of the page request. The code in Listing 4 tests to see whether the query exists and creates it if it doesn’t. It then tests to see if the data query is old or getting too large (if so, it deletes the old query and re-creates an empty query). Last, it inserts a record for this request.

The next step is to place a call to the custom tag inside every Application.cfm on your server to capture data about all your users. The last thing you need is some code that pulls data out of the query to build a report. The code for this report assumes that users aren’t active if they haven’t requested any pages for more than 15 minutes.

Copy all the code in Listing 4 into Studio and save it in the appropriate places on your server. Now you should be ready to go!

• • •

Please send your questions about ColdFusion (CFML, CF Server, or CF Studio) to AskCFDJ@sys-con.com. Visit our archive site at www.Netsite-Dynamics.com/AskCFDJ.

ABOUT THE AUTHOR

Bruce Van Horn, president of Netsite Dynamics, LLC, is a certified ColdFusion developer/instructor and a member of the **CFDJ** *International Advisory Board.*

BRUCE@NETSITEDYNAMICS.COM

Listing 1

```
<cfapplication name="MyApp" sessionmanagement="Yes">

<cflock scope="session" timeout="20" type="exclusive">
 <cfparam name="Session.Data" default="#StructNew()#">
 <cfset Request.Session = Duplicate(Session.Data)>
</cflock>
<cflock scope="application" timeout="20" type="exclusive">
 <cfparam name="Application.Data" default="#structNew()#">
 <cfset Request.Application = Duplicate(Application.Data)>
</cflock>
<cflock scope="server" timeout="20" type="exclusive">
 <cfparam name="Server.Data" default="#structNew()#">
 <cfset Request.Server = Duplicate(Server.Data)>
</cflock>
```

Listing 2

```
<cflock scope="session" timeout="20" type="exclusive">
 <cfset Session.Data = Duplicate(Request.Session)>
</cflock>
<cflock scope="application" timeout="20" type="exclusive">
 <cfset Application.Data = Duplicate(Request.Application)>
</cflock>
<cflock scope="server" timeout="20" type="exclusive">
 <cfset Server.Data = Duplicate(Request.Server)>
</cflock>
```

Listing 3

```
<cflock scope="session" timeout="20" type="exclusive">
<cfset Session.Data.loggedin = 1>
</cflock>
<cflocation url="../index.cfm">
```

Listing 4

```
<!--- Create a  custom tag, TrackUsers.cfm, to store infor-
mation about each user --->

<cflock timeout="30" throwontimeout="No" type="EXCLUSIVE"
scope="SERVER">
 <cfif (not isDefined("Server.qUsers")) OR (NOT
IsQuery(Server.qUsers))>
  <cfscript>
   Server.qUsers = QueryNew("User,AppName,RequestTime");
   QueryAddRow(Server.qUsers);
   QuerySetCell(Server.qUsers,"User",Attributes.User);
   QuerySetCell(Server.qUsers,"AppName",Attributes.AppName);

QuerySetCell(Server.qUsers,"RequestTime",CreateODBCTime(now
()));
  </cfscript>
 <cfelse>
  <cfquery name="qMaxReq" dbtype="query">
   select max(requesttime) as lastreq, Count(User) AS
RowCount
   from Server.qusers
  </cfquery>
  <cfset ElapsedTime = CreateODBCTime(now()) -
qMaxReq.LastReq>
  <cfif (TimeFormat(Variables.ElapsedTime,'m') gt 15) OR
(qMaxReq.RowCount gt 350)>
   <cfscript>
    Server.qUsers = "";
    Server.qUsers = QueryNew("User,AppName,RequestTime");
   </cfscript>
  </cfif>
  <cfscript>
   QueryAddRow(Server.qUsers);
   QuerySetCell(Server.qUsers,"User",Attributes.User);
   QuerySetCell(Server.qUsers,"AppName",Attributes.AppName);

QuerySetCell(Server.qUsers,"RequestTime",CreateODBCTime(now
()));
  </cfscript>
 </cfif>
</cflock>


<!--- Call the custom tag from every Application.cfm on
your server --->

<CF_TrackUsers User="#cgi.remote_addr#" AppName="MyApp">


<!--- Create a report page, Active_User_Report.cfm, to dis-
play data --->

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

<html>
<head>
 <meta http-equiv="Refresh"
content="30;url=users_view.cfm">
 <title>Current Users</title>
</head>

<body>
<cflock timeout="30" type="READONLY" scope="SERVER">
 <cfquery name="qShowUsers" dbtype="query">
  Select UserIP, AppName, Max(RequestTime) as LastReq,
Count(User) as Hits
  From server.qUsers
  Group by user, AppName
  Order by LastReq desc
 </cfquery>
 <cfset totalhits = server.qUsers.recordcount>
</cflock>

<h1>Current System Users</h1>

<table border="2">
<tr bgcolor="#d5d5d5">
 <td>&nbsp;</td>
 <th>User</th>
 <th>Application</th>
 <th>Idle Time</th>
 <th>Hits</th>
</tr>
<cfset dbsize = 0>
<cfset rownum = 0>
<cfoutput query="qshowusers">
<cfset dbsize = dbsize + qShowUsers.Hits>
<cfset elapsed = CreateODBCTime(now()) -
qshowusers.LastReq>
<cfif TimeFormat(elapsed,'m') lt 15>
<cfset rownum = rownum + 1>
<tr bgcolor="<cfif qshowusers.currentrow mod
2>White<cfelse>Aqua</cfif>">
 <td align="right">#variables.rownum#</td>
 <td>#qshowusers.userip#</td>
 <td>#qshowusers.AppName#</td>
 <td
align="right">#TimeFormat(variables.elapsed,'mm:ss')#</td>
 <td align="right">#qShowUsers.Hits#</td>
</tr>
</cfif>
</cfoutput>
</table>

DB Size = <cfoutput>#variables.totalhits#</cfoutput>

</body>
</html>
```

CODE LISTING

▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶

The code listing for this article is also located at

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

Subscribe Now

for Instant Access to CF Advisor

CFAdvisor.com!

12 months **$89**

24 months **$169**

Now on CD!

THE MOST COMPLETE LIBRARY OF EXCLUSIVE *CF ADVISOR* ARTICLES!

3 YEARS

40 ISSUES

200 ARTICLES

ONE CD

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

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

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

- custom tags
- functions
- variables
- framesets
- style sheets
- structures
- javascript usage
- debugging techniques
- programming practices

and much, much more...

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

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

"The Complete Works"

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

Easy-to-navigate HTML format!

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

CFBasics
Reviews
Scalability
Enterprise CF
Wireless
Verity
Source Code CF
Applications

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

Order Online and Save 10% or More!

www.JDJSTORE.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SYS-CON MEDIA

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

CFDJnews

Blue Martini Extensions for Dreamweaver UltraDev

(*San Mateo, CA*) – Extensions that Blue Martini Software, Inc., is delivering to Macromedia Dreamweaver UltraDev 4 reportedly enable developers to integrate the visual authoring power of Dreamweaver UltraDev with the Blue Martini external customer relationship management (eCRM) application suite. The new extensions enable developers to visually design JavaServer Pages and insert or modify objects and method calls for Blue Martini eCRM applications from within Dreamweaver UltraDev.

Some of the new features: live preview without staging, point-and-click authoring, wizards for JSP template creation, easy drag-and-drop tools, and integration with Blue Martini versioning.

www.bluemartini.com

New Managing Director at General Catalyst

(*Boston*) – General Catalyst, a private equity firm focused on early-stage investment opportunities and a source of operational and business-building expertise for entrepreneurs, has named David Orfao as a managing director. Most recently CEO of Allaire, Orfao will identify new investment opportunities within the enterprise software sector and help to ensure that General Catalyst portfolio companies are operationally established for rapid growth and success.

www.generalcatalyst.com

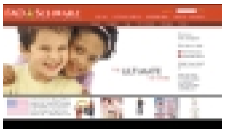
UMassOnline Builds Online Educational Platform

(*Washington, DC*) – Prometheus, a community source course management software developed by George Washington University in Washington, DC., has been selected by UMassOnline to participate in their new dual platform eLearning utility. Based on a ColdFusion application layer, Prometheus will provide course creation and management tools for the five different campuses across the University of Massachusetts system.

www.UMassOnline.net

FAO Site Relaunched with ColdFusion, JRun

(*Boston / San Francisco*) – According to Mindseye, Inc., a privately held Internet solutions company, toy retailer FAO Schwarz has relaunched its site, which Mindseye redesigned using Macromedia ColdFusion 5 and JRun 3.1.



Designed for simplicity and usability, the site features personalization, giving the consumer various options exclusive to the Web site including a view of order status, history, personal occasion reminders, and gift suggestions. It has an intuitive checkout process to make online purchasing more efficient and truly user-friendly and a custom e-mail marketing tool for direct e-mail marketing campaigns and customer tracking.

www.fao.com
www.mindseye.com
www.macromedia.com

SYS-CON Hailed Again by Inc Magazine

(*Montvale, NJ*) – **SYS-CON Media**, publisher of ***CFDJ*** and other *i*-technology magazines and a producer of *i*-technology developer conferences, has been named one of America's entrepreneurial growth leaders by *Inc* magazine, which recently released its annual list of the nation's 500 fastest-growing privately owned companies.

"We are delighted to be recognized once again as the fastest-growing publishing company in America," said Fuat Kircaali, founder and CEO of **SYS-CON Media**. The company was first honored by *Inc* in 1999.

Over the past five years, through 2000, **SYS-CON** has increased sales by 1,072%, giving it a ranking of 278 among *Inc*'s top 500. The firm was the only company in the publishing industry, and one of only 19 corporations headquartered in New Jersey, to make this year's list.

SYS-CON executives will accept the *Inc* 500 award next June at a reception held by New Mexico Governor Gary E. Johnson and *Inc* magazine in Albuquerque.

www.sys-con.com

Macromedia Flash Player 5 Shipping with Windows XP

(*Redmond, WA / San Francisco*) – Macromedia Flash Player 5 will be included with all versions of the Microsoft Windows XP operating system, the next major version of the Windows operating system.

www.macromedia.com

SYS-CON Media, Group Intelligence in Alliance

(*Montvale, NJ*) – **SYS-CON Media** and Group Intelligence have announced a strategic alliance in support of the fastest-growing Web-infrastructure market, WebSphere.

The companies will jointly deliver a full range of print, event, online, and digital services, creating the first information resource and infrastructure (InfoStructure) of the WebSphere marketplace.

WebSphere Developer's Journal, the newest *i*-technology publication from **SYS-CON Media**, will be published monthly starting in November.

www.sys-con.com
www.groupintelligence.com

CFDJnews

SYS-CON Media Launches Web Services Journal

(*Montvale, NJ*) – **SYS-CON Media** has debuted its premier issue of ***Web Services Journal***, the first print and online magazine devoted exclusively to the newest and most pervasive computing paradigm since the arrival of the Web itself.

The premier issue features articles from some of the best-informed developers, analysts, and executives in the *i*-technology world. This collector's issue includes a "CEO Round Table Discussion" led by executives and visionaries like David Litwack, CEO of Silverstream, Barry Morris, CEO of Iona, David Clarke, CEO of Cape Clear, Simon Phipps of Sun Microsystems, and Tyler Jewell of BEA Systems, among others.
www.sys-con.com/webservices

YesSoftware Introduces CodeCharge 2.0

(*San Francisco, CA*) – YesSoftware is shipping version 2.0 of CodeCharge, a Rapid Web Application Development Tool that allows developers and enthusiasts of all calibers to create database-enabled multiplatform Web applications with minimal effort.

CodeCharge 2.0 introduces a wide range of new features aimed at improving usability, functionality, adaptability, and compatibility with a wide range of development and server products, and ships with 15 examples of prebuilt Web applications including an online bookstore, club portal, yellow pages, employee directory, task management system, etc.
www.YesSoftware.com

SYS-CON Events Moving to Javits Center in 2002

(*Montvale, NJ*) – **SYS-CON Events, Inc.**, will relocate its East Coast events to the Jacob Javits Convention Center in New York City in 2002. Web Services Edge 2002 East–International Web Services Conference & Expo and JDJEdge 2002–International Java Developer Conference & Expo will take place June 24–27, 2002. **SYS-CON**'s East Coast events will complement the Technology Exchange Week New York – TECHXNY – a weeklong showcase for the most dynamic business technology players and products in the world.

"We look forward to coming back to New York City for the third year in a row and plan to double our exhibit floor by moving to the Jacob Javits Convention Center," said Grisha Davida, vice president of business development at **SYS-CON Media**. "This will allow us to host Java, XML, .Net, and all related technologies under our Web services canopy."

"We will open the Call for Papers for next year's conference on November 30, 2001," according to Cathy Walters, vice president of **SYS-CON Events**.
www.sys-con.com

CFDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
ABLECOMMERCE	WWW.ABLECOMMERCE.COM	888.801.1333	2
ACTIVEPDF	WWW.ACTIVEPDF.COM	888.389.1653	4
ASSETNOW	WWW.ASSETNOW.COM		23
CFADVISOR.COM	WWW.CFADVISOR.COM		49
CFDYNAMICS	WWW.CFDYNAMICS.COM	800.422.7957	29
CFXHOSTING	WWW.CFXHOSTING.COM	866.CFX-HOST	45
EVOLUTIONB	WWW.SYNERGYANYWHERE.COM		55
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.HOST	21
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800.379.7729	56
JAVA DEVELOPER'S JOURNAL	WWW.SYS-CON.COM	800.513.7111	47
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	37
MACROMEDIA	WWW.MACROMEDIA.COM/GO/MASTERING	877.460.8679	11
MACROMEDIA	WWW.VUE.COM/MACROMEDIA		17
MACROMEDIA	WWW.MACROMEDIA.COM/GO/USERGROUPS	888.939.2545	19
MINERVA	WWW.MINERVA.NET	888.667.7231	25
PACIFIC ONLINE	WWW.PACONLINE.NET	877.503.9870	51
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	13
RACKSPACE	WWW.RACKSPACE.COM	888.850.1470	3
SYS-CON MEDIA REPRINTS	WWW.SYS-CON.COM	201.802.3026	34
WEBLOGIC DEVELOPERS JOURNAL	WWW.SYS-CON.COM/WEBLOGIC	800.513.7111	35
WEBSPHERE DEVELOPER'S JOURNAL	WWW.SYS-CON.COM/WEBSPHERE	800.513.7111	53
WIRELESS EDGE CONFERENCE & EXPO	WWW.SYS-CON.COM	201.802.3069	39
XML-JOURNAL	WWW.SYS-CON.COM	800.513.7111	41
YES SOFTWARE	WWW.CODECHARGE.COM	650.754.9810	9

<the world's leading independent coldfusion magazine>

ColdFusion Developer's Journal

home advertise subscribe contact SYS-CON MEDIA

<dot.com>

- buyer's guide
- coldfusion forums
- mailing list
- coldfusion jobs
- coldfusion store

<magazine>

- advertise
- authors
- customer service
- editorial board
- subscribe

<content>

- archives
- digital edition
- editorial
- features
- interviews
- product reviews

<conferences>

Web Services Edge 2002
& JDJEdge 2002
Conference & Expo
June 24–27, 2002
Jacob Javits Convention
Center, New York, NY

<search cfdj>

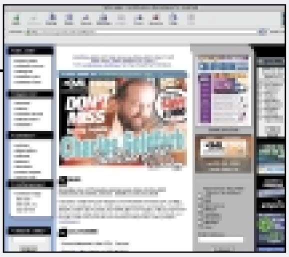
Search

November 2001

What's Online

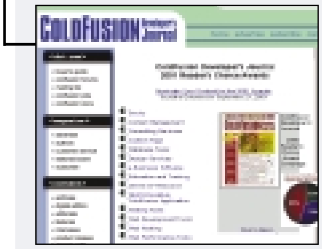
www.sys-con.com/coldfusion

CFDJ Online



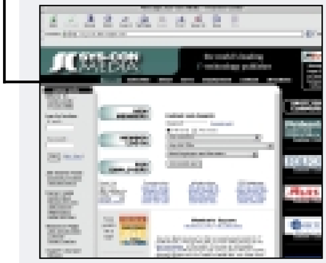
Check in every day for up-to-the-minute news, events, and developments in the ColdFusion industry. Visit www.sys-con.com/coldfusion and be the first to know what's happening.

Readers' Choice Awards – Vote Now



Make your voice heard...this is your chance to show what you think. Categories for this year's awards include Best Book, Best Consulting Service, Best Custom Tag, Best Database Tool, Best Design Service, Best E-Business Software, Best Education and Training, Best Testing Tool, Best Web Development Tool, Best Web Hosting, Best Web Site, Best Web Application, and Most Innovative CF Application. To vote in this year's awards visit www.sys-con.com/coldfusion/readerschoice2001.

Search ColdFusion Jobs

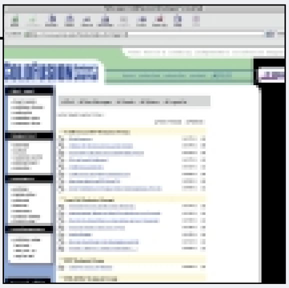


ColdFusion Developer's Journal is proud to offer an employment portal for IT professionals. Get direct access to the best companies in the nation. Learn about the "hidden job market" and how you can find it. As an IT professional curious about the market, this is the site you've been looking for.

Simply type in the keyword, job title, and location – and get instant results. You can search by salary, company, or industry.

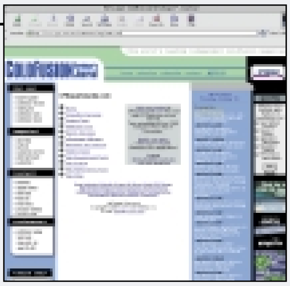
Need more help? Our experts can assist you with retirement planning, putting together a résumé, immigration issues, and more.

Answers Are a Click Away

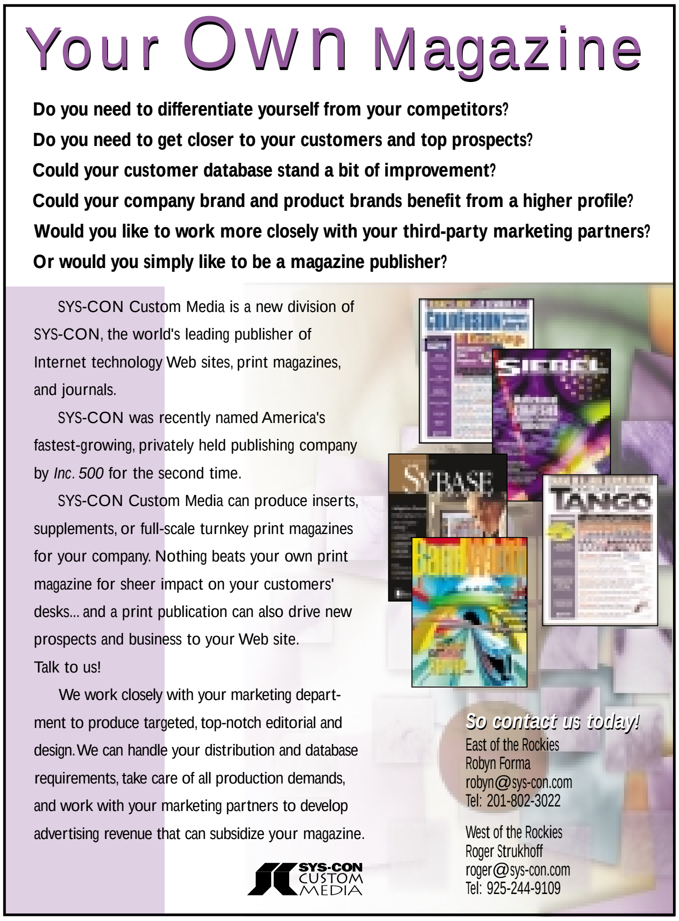


Are you blocked by your blocking factor? Are your synchronization objects out of sync? Visit the ***CFDJ*** Technical Forum (www.sys-con.com/forums/index.cfm?cfapp=14) and see what other developers have to say. You can ask a question, share a tip, improve your code, find a way to solve a glitch, and network with ColdFusion professionals. Recent postings have generated online discussions on security issues, data importing, dialog boxes, submit buttons, image info tags, ColdFusion forms, and more.

The Latest in CF at the CFBuyersGuide.com



The most-read CF resource on the Internet, CFBuyersGuide.com (www.sys-con.com/coldfusion/wbg/index.cfm), is essential for those who want to learn about the newest books, software, tools, and services related to ColdFusion. Web hosting, testing tools, books, e-business software, custom tags, Web development tools, consulting services, education and training, and much more are featured at this comprehensive site. Why waste time searching the Net for your CF needs! Discover the convenience of CFBuyersGuide.com.



SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 **Fax:** 201 782-9600

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

advertising
senior vp, sales & marketing
Carmen Gonzalez carmen@sys-con.com
vp, sales & marketing
Miles Silverman miles@sys-con.com
advertising director
Robyn Forma robyn@sys-con.com
advertising account manager
Megan Ring megan@sys-con.com
associate sales manager
Carrie Gebert carrie@sys-con.com
associate sales manager
Alisa Catalano alisa@sys-con.com
associate sales manager
Kristen Kuhnle kristin@sys-con.com

editorial
executive editor
M'lou Pinkham mpinkham@sys-con.com
editor
Nancy Valentine nancy@sys-con.com
managing editor
Cheryl Van Sise cheryl@sys-con.com
associate editor
Jamie Matusow jamie@sys-con.com
associate editor
Gail Schultz gail@sys-con.com
associate editor
Brenda Greene brenda@sys-con.com
assistant editor
Lin Goetz lin@sys-con.com

production
vice president, production
Jim Morgan jim@sys-con.com
art director
Alex Botero alex@sys-con.com
assistant art director
Cathryn Burak cathyb@sys-con.com
assistant art director
Louis F. Cuffari louis@sys-con.com
assistant art director
Richard Silverberg richards@sys-con.com
graphic designer
Abraham Addo abraham@sys-con.com
graphic designer
Aarathi Venkataraman aarathi@sys-con.com

sys-con events
vice president, events
Cathy Walters cathyw@sys-con.com
sales executive, exhibits
Richard Anderson richard@sys-con.com
sales executive, exhibits
Michael Pesnick michael@sys-con.com
conference manager
Michael Lynch mike@sys-con.com
show assistant
Niki Panagopoulos niki@sys-con.com
JDJ store manager
Anthony D. Spitzer tony@sys-con.com

web services
web designer
Stephen Kilmurray stephen@sys-con.com

accounting
chief financial officer
Bruce Kanner bruce@sys-con.com
assistant controller
Judith Calnan judith@sys-con.com
accounts payable
Joan LaRose joan@sys-con.com
accounts receivable
Jan Braidech jan@sys-con.com
accounting clerk
Patti DelVecchio patti@sys-con.com

Letters to the Editor. . .

The Power of Flash

Dennis Baldwin's article ["ColdFusion Meets Flash," Vol. 3, issue 8] was very thorough and not for the novice developer. I've looked through the source files and implemented the menu on my own site. The code was well documented and straightforward. A job well done.

joe@joeblow.com

I've been looking for something like Dennis Baldwin's article for a while. I was very disappointed though, since I couldn't go past page one. Guess I'll have to hunt for the journal, but I know it'll be difficult or impossible to find here in Russia. What a pity!

douglas.c@russia.com

Editor's Note:

To access the rest of Dennis Baldwin's article on the **CFDJ** *Web site, you need to subscribe to* **ColdFusion Developer's Journal***; this subscription provides you with free access to the digital edition.*

I dug around ***CFDJ*** and found the .txt source files but didn't see anything else, specifically the .fla and .mdb files. Could you please send me a copy of those?

Great article! I'm looking forward to experimenting with Flash and CF!

Jim Priest
priest@thecrumb.com

Editor's Note:

Thank you for your note. We have added these additional files to Dennis Baldwin's article on the **CFDJ** *Web site as a downloadable zip file.*

I read Dennis Baldwin's article and put together a very nice menu based on his code using my own graphics. It works well. Thank you.

I want to use the menu in a frameset and I'm trying to figure out how to add a URL "target" variable to the database and code. Any hints you can give would be most appreciated. Thanks for the article.

ivan.sinclair@netcentrics.net

I enjoyed Dennis Baldwin's Flash article. He mentioned the source files. Would you send a copy including the Access database, if they're not too large?

Bill Allred
bill.allred@verizon.com

ColdFusion into Flash

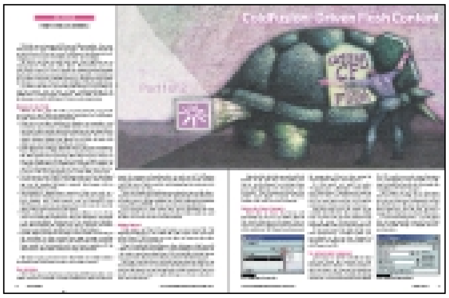
In the article by Randy H. Drisgill and Jason Montilla ["ColdFusion-Driven Flash Content," Vol. 3, issue 8] about using ColdFusion variables in Flash, they state:

Flash doesn't allow spaces between the parts of the statements, though the variable can have spaces in it. You can't have spaces after the ampersand, variable name, or equal sign, or between the statements.

I found this confusing and wondered if you could please clarify what you mean. What confused me specifically was "though the variable can have spaces in it." What does this mean?

The article was great and most helpful, keep up the good work.

Rick Miltimore
rmiltimore@creativechannel.com



Randy H. Drisgill replies:

Rick, glad you liked it. What I was trying to say was something like this.

This is okay:

```
varname=the dog is good&varname2=test
```

This is bad:

```
var name=thedog& varname2 =thedog2
```

The part after the "=" sign allows spaces for strings and such, but the part where you actually set the variable doesn't.

Is that clearer? The concept is hard to explain, but when you do it wrong, you'll know!

Randy H. Drisgill
rdrisgill@vshift.com

EVOLUTIONB

www.synergyanywhere.com

INTERMEDIA.NET

www.intermedia.net