PERSONALIZATION PAYS

U.S. $8.99 (Canada $9.99)

ColdFusion Developer's Journal

ColdFusionJournal.com

May 2000 Volume: 2 Issue: 5

SYS-CON MEDIA

ColdFusion Developer's Journal

EDITORIAL ADVISORY BOARD
STEVEN D. DRUCKER, JIM ESTEN, BEN FORTA,
STEVE NELSON, RICHARD SCHULZE, PAUL UNDERWOOD

EDITOR-IN-CHIEF ROBERT DIAMOND
ART DIRECTOR JIM MORGAN
EXECUTIVE EDITOR M'LOU PINKHAM
SENIOR EDITOR JEREMY GEELAN
PRODUCTION EDITOR CHERYL VAN SISE
ASSOCIATE EDITOR NANCY VALENTINE
PRODUCT REVIEW EDITOR TOM TAULLI
TIPS & TECHNIQUES EDITOR MATT NEWBERRY

WRITERS IN THIS ISSUE
CHARLES AREHART, JERRY BRADENBAUGH, STEVE DAN,
ROBERT DIAMOND, BEN FORTA, HAL HELMS,
DANIEL JEAN, DAVID SCHWARTZ, RAYMOND THOMPSON

SUBSCRIPTIONS
SUBSCRIBE @SYS-CON.COM
FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,
PLEASE SEND YOUR LETTERS TO
SUBSCRIPTION DEPARTMENT.

SUBSCRIPTION HOTLINE 800 513-7111
COVER PRICE $8.99/ISSUE
DOMESTIC $79/YR. (12 ISSUES)
CANADA/MEXICO $99/YR
OVERSEAS $129/YR
BACK ISSUES $12 EACH

PUBLISHER, PRESIDENT AND CEO FUAT A. KIRCAALI
VICE PRESIDENT, PRODUCTION JIM MORGAN
VICE PRESIDENT, MARKETING CARMEN GONZALEZ
CHIEF FINANCIAL OFFICER ELI HOROWITZ
ADVERTISING ACCOUNT MANAGER ROBYN FORMA
ADVERTISING ACCOUNT MANAGER MEGAN RING
ADVERTISING ASSISTANT CHRISTINE RUSSELL
ADVERTISING INTERN MATT KREMKAU
GRAPHIC DESIGNER ALEX BOTERO
GRAPHIC DESIGNER JASON KREMKAU
GRAPHIC DESIGNER ABRAHAM ADDO
GRAPHIC DESIGN INTERN AARATHI VENKATARAMAN
WEBMASTER BRUNO Y. DECAUDIN
WEB DESIGNER MICHAEL MALOOF
WEB DESIGNER STEPHEN KILMURRAY
WEB SERVICES INTERN DIGANT B. DAVE
WEB SERVICES INTERN BRYAN KREMKAU
CUSTOMER SERVICE MANAGER CAROL KILDUFF
CUSTOMER SERVICE ANN MILILLO
JDJ STORE.COM JACLYN REDMOND

EDITORIAL OFFICES
SYS-CON PUBLICATIONS, INC. 39 E. CENTRAL AVE.,
PEARL RIVER, NY 10965
TELEPHONE: 914 735-7300 **FAX:** 914 735-6547

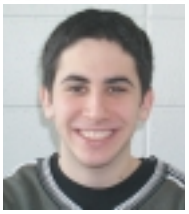
COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
IS PUBLISHED MONTHLY (12 TIMES A YEAR)
FOR $79 BY SYS-CON PUBLICATIONS, INC.,
39 E. CENTRAL AVE., PEARL RIVER, NY 10965-2306.

POSTMASTER
SEND ADDRESS CHANGES TO:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON PUBLICATIONS, INC.
39 E. CENTRAL AVE., PEARL RIVER, NY 10965-2306

WORLDWIDE DISTRIBUTION
BY **CURTIS CIRCULATION COMPANY** 739 RIVER ROAD,
NEW MILFORD, NJ 07646-3048 PHONE: 201 634-7400

DISTRIBUTED IN USA
BY **INTERNATIONAL PERIODICAL DISTRIBUTORS**
674 VIA DE LA VALLE, SUITE 204, SOLANA BEACH, CA 92075
619 481-5928

Extra, Extra, Read All About It!



BY **ROBERT DIAMOND**

Last month I promised to discuss the pros and cons of hosting a ColdFusion site in-house versus out of house in this month's editorial. But in light of some recent news and exciting additions to the world of ***CFDJ***, that editorial has been bumped to next month. I don't think anyone will miss it, though, as I've got some better stuff to write about.

People send in article proposals and column ideas all the time. Most of them are good, and some of them are really good. Every now and then one comes along that really excites me because it's genuinely innovative. Just such a proposal recently came in – submitted by Bruce Van Horn.

For those of you that may know nothing about him, Bruce is an Allaire certified instructor and president of Netsite Dynamics. What he proposed was a new column to run both in the print publication and on the ***CFDJ*** Web site: an ongoing question-and-answer session to be called "Ask the Training Staff" that would allow developers to send in questions related to CFML, CF Server, CF Studio – in other words, anything related to ColdFusion. Bruce will answer some of the questions that come in, assisted by several hand-picked certified Allaire instructors, all eager to help and collectively possessed of a lot of CF knowledge.

These instructors will also submit questions their students have asked in the classroom, along with the appropriate answers. This will be a fantastic way to cover the latest questions floating around in the industry. Every month we'll publish those questions that we think will be of greatest interest to the largest number of readers. We'll also have a more comprehensive and complete list on the Web site. Questions can be submitted to Bruce at **AskCFDJ @sys-con.com**.

The first Annual ***CFDJ*** Readers' Choice Awards – the Oscars of the ColdFusion industry – are now going strong, with over a thousand votes already logged at www.sys-con.com/coldfusion/readers-choice2000/. Unlike the Hollywood version, we won't lose any trophies or ballots. Stay tuned to see if the *Wall Street Journal* scoops us, though.

If you haven't voted yet, do; if you already have, encourage your friends to do the same – voting continues until September 1, 2000. Here's a quick update on the leaders in each category at press time:

- **Best Book:** *Advanced ColdFusion 4.0 Application Development*
- **Best Consulting Service:** Fig Leaf Software
- **Best Custom Tag:** CF_StockGrabber
- **Best Database Tool:** Macromedia Generator
- **Best Design Service:** Fig Leaf Software
- **Best E-Business Software:** AbleCommerce
- **Best Education and Training:** Cold Fusion and Macromedia Training
- **Best Testing Tool:** LoadRunner
- **Best Web Development Tool:** Macromedia Dreamweaver
- **Best Web Hosting:** Virtualscape's ColdFusion Webhosting
- **Best Web Site:** Allaire Corporation Homepage
- **Best Web Application:** Synergy-NOW
- **Most Innovative CF Application:** LoadRunner

Next month you can look forward to the first Q&A session and to my postponed thoughts on ColdFusion hosting. I love feedback, so drop me a line at rdiamond@sys-con.com with any thoughts you may have about the magazine in general, or with specific suggestions about what topics you'd like to see covered. I look forward to hearing from you.

ABOUT THE AUTHOR
Robert Diamond is editor-in-chief of **ColdFusion Developer's Journal**.

ROBERT@SYS-CON.COM

DB Conversion and CF

A User's Tale

With some clever conversion coding generated by CF, converting from one database to another can be easy. But there are a few "gotchas" along the way...

BY RAYMOND THOMPSON

The system I'm currently involved with is used to track samples from several major nuclear facilities. It's a Web-based application that allows the entry, cost bidding by laboratories, tracking, invoicing and final disposition of samples.

The system was originally character-based and had been converted to a Web-based application. ColdFusion was the language of choice for the Web application because of the capabilities of the language and the speed at which pages can be developed.

In the original scheme of things the database chosen was Ingress. A layer had to be added to the ColdFusion server that allowed an ODBC source to be created to communicate with ColdFusion. This additional layer of software was from OpenLink. It worked, but there were problems.

The biggest difficulty was the restriction that only a single database query could be active at a single time. Much effort was put into finding out where the problem was. Ingress blamed OpenLink, OpenLink blamed ColdFusion and Allaire blamed Ingress. The bottom line was that no one vendor was willing to take responsibility for the problem. It was so bad that unless ColdFusion was instructed in the datasource screen to limit the connections to one, the system would hang. This would require recycling the ColdFusion service.

It's always a challenge, in dealing with third parties, when there are problems. This was no exception, and all three vendors proved true to this philosophy. Something had to be done.

A Different DB

Since there were already several servers running Oracle, it was decided that the database would be converted to Oracle. How this was to be accomplished was initially thought to be a big issue. But as it turned out, the capabilities of ColdFusion made it easier.

One factor making the conversion difficult was that the system was still being developed when the decision was made to change databases. There were still minor changes to the database and code was still being heavily developed.

This presented two problems. One was creating something to convert the database from one format to another. The other was that the code currently being developed, and currently running, had to be able to access either database. Not

desirable circumstances, but ones that had to be handled.

DB Differences

First we had to deal with the differences in the constructs used to access the database. Since Ingress was an ODBC connection and Oracle was going to be using a native driver connection, some constructs were going to be significantly different.

The biggest difference is dates. When sending a date to the Ingress database the CreateODBCDate(field) was used. Fairly simple. Oracle was more complicated; because of Y2K requirements, it was determined that all dates would be fully qualified with a four-digit year. The default format for Oracle didn't provide the necessary format. For Oracle to properly recognize a date, the format had to be explicitly provided. Listing 1 shows the format that was used to pass dates to Oracle.

The code in Listing 1 created a ColdFusion variable that contained the exact format of the date to be represented in Oracle. The Oracle "to_date()" function was then used to cause Oracle to convert the field to a date using the format that we specified.

There were problems with some column names in the database. In Ingress column names of DATE and COMMENT were valid, as Ingress was context-sensitive when allowing names. Oracle isn't so forgiving and pitched a fit about these names.

Another difference was the way that data fields were aliased in the current scripts. The format that was used in developing the scripts to work with Ingress was "<new_name" <database _field_name>. Oracle required the format <database_field_name> as <new_name>. This turned out not to be a big problem, as Ingress supported the format that Oracle used. This was nothing more than changing the scripts. A tedious task, but easily accomplished and compatible with both databases.

Now all that was left was to find some method that could be used so ColdFusion could determine which database format was being used. By determining the database desired, dynamic scripts could be created to accommodate both databases. Why do this? Well, it had to do with the development server and the production server. The development was going to migrate to Oracle so that changes, new code and testing could continue. Because there would be a delay until the production environment was converted, and code development could not stop, code had to be developed that would run in both environments. By developing code that was self-aware, code could continue being developed and placed into the production environment without any significant hassles.

The next code (see Listing 2) determines the particular server and the database that is to be used. With this knowledge several global variables that provided the name of the datasource and, most important, the type of database being used could then be set. This allowed the creation of dynamic queries that adapted themselves to the requirements of the database. This decision was made based on the name of the server (actual name changed for security reasons). If the code was running on the development server, the mode was set to Oracle and the name of the database connection set. If the code was running on the production server, the mode was set to Ingress and the name of the database connection was set.

This code was placed in the APPLICATION.CFM script so that only one copy of the mode determination code would be required. The variables were made session variables so that the items would be available in all scripts. The modes are constants so that when the code determines the mode, hard-coded constants in the scripts were not required. Standard variables probably could have been used instead of session variables in this case.

The process of the queries modifying themselves now becomes quite easy. All that the code had to do was determine the mode and use the proper statement for the database. Listing 3 shows an example of the dynamic query code.

When the query code was run, it would produce the proper code for the database that was being used. By using the simple <CFIF>, <CFELSE> and </CFIF> constructs, the proper statement in the proper format was passed to the database. The net result of this was code that could be developed on one system, tested on that system and then moved to a different system with a different database with a very high assurance that it would work properly. Development could continue while conversion plans were being made.

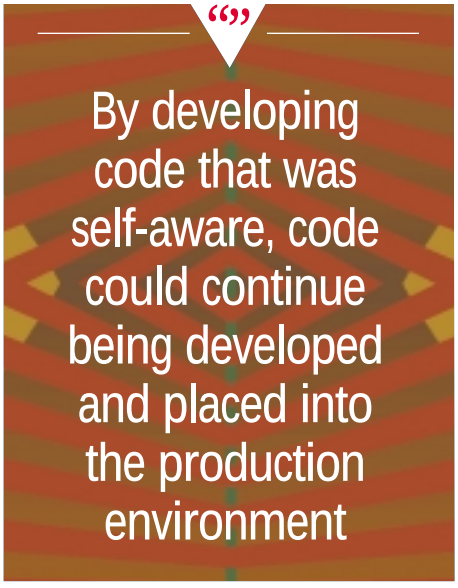
Database Conversion

With the code requirements satisfied, the next step was the actual conversion. In the development environment the tables were populated by hand with minimal testing and development data. The challenge was how to convert the Ingress database to Oracle. I wouldn't be surprised if there were conversion products on the market, but there was already a means to accomplish the conversion using ColdFusion.

CF can handle multiple datasources with ease. Since a script could access multiple data sources, why not build scripts that would read the data from the Ingress database and then insert the data into the Oracle database? On the surface it seems easy. But with 50-plus tables in the system, the thought of hand-coding all that conversion code was not something to be excited about.

Then, somewhere out of the blue, a suggestion was made to have ColdFusion generate the conversion code. Since ColdFusion could get information about the columns in the tables, it should be easy for ColdFusion to process these columns. Then it was mentioned that string and numeric columns had to be handled differently, as string columns needed quotes around the data. Also, dates would have to be processed using the to_date() Oracle function.

After a little analysis of the column names, the type of data could be determined from the name. Some of the date fields had the text "DATE" somewhere in the name. A list was made of those column names that did not have the text "DATE." The numeric columns were then identified and a list made. If the column served the same function in multiple tables, it had the same name in all the tables where it was used. This greatly simplified the process of determining the column type.



It was also observed that there would be some problems with certain column names. This was discovered during testing of the development code. In Ingress a column name of "COMMENT" had been used. Oracle got really upset at this. The

conversion code was also going to have to make allowances for this problem. Another problem: one column in Ingress was called "DATE." This also needed handling during conversion. The code generation would merely replace these column names with names that were appropriate for Oracle.

The conversion also had to allow for NULL fields. Since an empty string was not a NULL, it was decided that all empty string fields would be converted to NULL. The application code already handled NULL fields, so this wouldn't present any problems.

The next process was to build the conversion code generator. This proved to be a fairly tedious task, concerning itself with quotes, and commas, and emitting CF tags and variables with pound signs.

The conversion code also had to generate code to provide conversion statistics so that it could be determined that the conversion went properly. Considerations such as record counts input and output, table names and timing were to be provided. This also had to be created by the code generator. Allowance was also required for proper placement of commas in the SQL statements. The order in which columns would be retrieved couldn't be determined at the time, so the code generator had to allow for the last column and not emit the comma.

The code worked by writing a file of generated CFML to a file. This file would then be processed using <CFINCLUDE>. There were several steps required to get the correct code generated and get the proper variables. The first step (see Listing 4) was eliminating any prior file from the system by checking if it existed and, if so, removing the file to avoid any nasty surprises from prior runs.

"Because several of the dynamic variables are now contained within a string, ColdFusion does variable substitution before processing the string

The code would then write out some comments about the generated conversion code. The difficult part was getting quotes and pound signs in the proper format to get them written to the file. The solution was escaping the characters, which is done by using two consecutive characters to get one character. The code in Listing 5 output some variables that are being set with <CFSET> and are going to be used in the conversion code when it runs. All of the lines in the listing that are <CFFILE> have [action="APPEND" file="#ProcessFile#"] in the line. It is left off in the example to make the lines shorter.

Next, a <CFLOOP> is used to process the list that contains the names of the tables in the database. These table names had to be hand-coded in the script. Within this loop the name of the query needed to be established as it was going to be dynamically named. Some variables were set and the queries were processed. The amount of data returned was limited to one row because at this point only the names of the columns were of significance. One row of data was enough to provide the information (see Listing 6).

The query name needed to be variable to avoid ColdFusion's problem of having many queries of the same name. It's not exactly clear what happens, but there was confusion on the part of ColdFusion and column names from other tables that had started appearing. This required that the <CFQUERY> name be unique for each table. The query name was simply made to match the name of the table. A <CFQUERY> was also generated to get the count of the number of rows for information purposes.

Next, the names of the columns needed to be determined. ColdFusion provides this information in a structure called "<queryName>.ColumnList" where <queryName> is the name of the query. What it does not provide are the attributes of the columns. Since the table name was extracted from a list of tables, using the Evaluate() function with the name from the list and appending "ROWS.COLUMNLIST" allows the names of the columns to be obtained and sorted (see Listing 7).

Now the code needed to be generated to access the database. Remember, this is all inside the list loop of the tables. Because several of the dynamic variables are now contained within a string, ColdFusion does variable substitution before processing the string. The only difficulty was getting the datasource to properly reflect a variable that would exist at processing, not code-generation time. This was done by escaping the pound sign

LISTING 1

```
<CFSET CURRENTDATETIME=DATEFORMAT(NOW(),"MM/DD/YYYY") & " "
DATEFORMAT(NOW(),"HH:MM">
<CFQUERY DATASOURCE="#SESSION.DBSOURCE#">
  UPDATE UTILTBL
  SET DOLC=TO_DATE('#CURRENTDATETIME#','MM/DD/YYYY 24HH:MM')
</CFQUERY>
```

LISTING 2

```
<!-----------------------------------------------------------------
 The next variables are constants that can be used to determine
the operating mode of the system (Ingress or Oracle).
-----------------------------------------------------------------
<cfset Session.Oracle="O">
<cfset Session.Ingress="I">
<cfset Session.DBMode="">

<!-----------------------------------------------------------------
 This code will set the proper operating mode (using the vari-
ables above) by determining what path we are operating from.
-----------------------------------------------------------------
----->
<cfif CGI.Server_Name EQ "Development Server">
  <!--- Server name indicates running on Development server ---
>
  <cfset Session.DBMode=Session.Oracle>
  <cfset Session.dbSource="OracleDB">
<cfelseif CGI.Server_Name EQ "Production Server">
  <!--- Server name indicates running on Production server --->
  <cfset Session.DBMode=Session.Ingress>
  <cfset Session.dbSource="IngressDB">
</cfif>
```

LISTING 3

```
<CFSET CurrDateTime=DateFormat(Now(),"MM/DD/YYYY")
                    & " "
                    & TimeFormat(Now(),"HH:MM")>
<cfquery name="InsertAnal" datasource="#Session.dbSource#">
  INSERT INTO analmstrtbl (dolc,
  <cfif Session.DBMode EQ Session.Oracle>
    comments
  <cfelse>
    comment
  </cfif>
```

characters. The first part of the output query was also written to the generated code file (see Listing 8).

Another <CFLOOP> was created to process each individual column name to generate to the fields and data to insert. Each column was checked to determine the type and what changes needed to be made to the data. This had to be done in two parts, the first part being the column name conversion (COMMENT had to become COMMENTS). It was simple code, merely checking for the value of the column name and changing it if necessary (see Listing 9). This basic code inside the loop was repeated for each change that needed to be made to the column names. It was necessary to count the columns to know when the last column was reached, as a parenthesis had to be used on the last column instead of a comma.

Creating the data portion of the query involved a little more work. Field names had to be checked to see if special handling was required. Dates and numeric fields needed to be formatted differently. For date in particular, code had to be created that would check for NULL or empty fields during conversion and properly set to NULL. This required that a <CFIF> <CFELSE></CFIF> construct be generated. If the field contained a date, then the date-formatting code for Oracle needed to be generated. Allowance for the last column was also required here so that a parenthesis could be emitted instead of a comma (see Listing 10).

The loop was ended for the columns in the table, some additional code was emitted for statistical purposes and the loop for the table name was ended. When this script finished, the conversion code had been written by ColdFusion and accurately reflected the structure of the table at conversion time.

> Suddenly the conversion stopped. The ColdFusion server was dead, completely toast, Blue Screen of Death. It had to be rebooted

All that was left was to create a template to include the generated code. This template merely had to provide the names of the tables, and some known conversion changes (column names, fields that were dates, etc). Testing indicated that all was working well and conversion could take place at any time in the future.

Conversion Time

Developing the conversion code generator went easily after determining the cases that had to be allowed. Testing showed that the code generated would do the conversion properly. The testing was done with a stripped-down database that contained only a subset – but complete set – of data that would be faced in the conversion process. The scheme worked. We could continue to develop code that worked in both databases, and modifications to the database could be made without worrying about the conversion code. With everything prepared, the conversion was set for a three-day weekend about a month away.

When the time arrived to do the conversion, the connections were made to the databases, and the code generator was run and quickly scanned to make sure that it looked good. All was well.

It was known from the start that the code would run a long time. The timeouts for the scripts were effectively eliminated for this conversion process. The conversion was started and the wait began. A connection through SQL*NET to the destination database was available so the progress was monitored by doing simple queries on the database.

Suddenly the conversion stopped. The ColdFusion server was dead, completely toast, Blue Screen of Death. It had to be rebooted. The table that was being processed was cleaned out, the conversion-generation script modified to eliminate tables that had been converted, the code generator rerun and the conversion started again. In a short period of time the ColdFusion server died a catastrophic death again. Something was seriously wrong.

The table that was being converted was found to have a large number of rows. This should not have presented a problem, because during the trial conversion on the development server this same table had converted with no problems. A little more research revealed that the development server had 512MB of memory. The production server had 512MB of memory. Some more research. It was dis-

```
  )
  VALUES (
   <cfif Session.DBMode EQ Session.Oracle>
     to_date('#CurrDateTime#','mm/dd/yyyy hh24:mi'),
   <cfelse>
     #CreateODBCDateTime(Now())#,
   </cfif>
  '#Trim(Form.Comments)# '
  )
</cfquery>
```

LISTING 4

```
<CFSET PROCESSFILE=GETDIRECTORYFROMPATH(GETTEMPLATEPATH())
                    & "PROCESSOTHERCODE.CFM">
<CFIF FILEEXISTS(PROCESSFILE)>
   <CFFILE ACTION="DELETE" FILE="#PROCESSFILE#">
 </CFIF>
```

LISTING 5

```
<CFFILE OUTPUT="<CFSET SOURCEDB=""#DBSOURCE#"">">
 <CFFILE OUTPUT="<CFSET     DESTINATIONDB=""#ORACLEDB#"">">
 <CFFILE OUTPUT="<CFSET TIMECONVERTSTARTED=NOW()>">
 <CFFILE OUTPUT="<CFSET QUERYNAME="""">">
 <CFFILE OUTPUT="<CFSET INSERTNAME="""">">
 <CFFILE OUTPUT="<CFSET OUTTABLE="""">">
 <CFFILE OUTPUT="<CFSET OUTPUTCOUNT=0>">
 <CFFILE OUTPUT="<CFSET FILECOUNT=0>">
 <CFFILE OUTPUT="<CFSET TOTALROWCOUNT=0>">
```

LISTING 6

```
<CFLOOP INDEX="DBNAME" LIST="#DBLIST#">
 <CFSET QUERYNAME="GET#DBNAME#">
 <CFSET INSERTNAME="PUT#DBNAME#">
 <CFSET OUTTABLE="#DBNAME#">
 <CFSET FILECOUNT=FILECOUNT+1>
 <CFQUERY NAME="#QUERYNAME#" DATASOURCE="#DBSOURCE#">
   SELECT * FROM #DBNAME#
   WHERE ROWCOUNT < 2
 </CFQUERY>
 <CFQUERY NAME="#QUERYNAME#ROWS" DATASOURCE="#DBSOURCE#">
   SELECT COUNT(*) AS NUMBERROWS
   FROM #DBNAME#
 </CFQUERY>
 <CFSET QUERYROWS=EVALUATE(QUERYNAME & "ROWS.NUMBERROWS")>
```

covered that the swap space on the production server was limited, whereas the development server had a very large swap file space. This was the problem. NT is not intelligent enough to terminate the task that fills the swap space. NT just dies. No warnings, no messages, nothing.

To resolve the problem, the large table was eliminated from the list of tables. The code generator was rerun, and the conversion was restarted with the remaining tables. Everything finished normally. Then the code generation script was modified to generate code only for the problem table. This generated code was then modified by hand to limit the rows returned to something less than 33,000 rows. Since the problem table had 140,000 rows, it took several iterations to finish the conversion.

Success at last. The conversion was successful and ColdFusion had done the work. The dual database code was left in the scripts for a couple of months for contingency purposes. Eventually the dual database code was removed from all the scripts. The users were never aware that a database conversion had taken place. This was the real test of the process and it passed with flying colors.

Summary

This technique opens up a whole realm of possibilities. The first that immediately comes to mind is a dynamic query and reporting function. A template could be created that would present the user with a series of questions using standard HTML elements – something like the columns to select, what conditions to select information and output format options desired. ColdFusion could process this information and generate a script. This script could then be immediately included into the script and the results sent back to the user's screen.

A database conversion is never an easy task. Providing code that works in both environments was easy to accomplish using the dynamic nature of the ColdFusion code. ColdFusion was also used to relieve much of the tedium of building the conversion code. ColdFusion was also used to do the actual conversion process. Would it have been easier to just do the scripts manually with a lot of cutting and pasting and changing where necessary? Perhaps. But the database structure at the time of conversion would not be known in advance. It was also not known when the conversion would take place. Development could not be stopped based on an unknown date. This allowed the conversion logic to be written and tested well in advance. It also proved that a conversion for Ingress to Oracle would work with ColdFusion.

While there were glitches in the process, it went surprisingly well. It was learned in the process that having a large swap space could avoid problems with large results being returned from queries. Just because it works on a development server does not mean that it will work on a production server. Using the dynamic nature of ColdFusion allowed a difficult and tedious process to be accomplished fairly painlessly.

ABOUT THE AUTHOR

Ray Thompson is employed by Q Systems in Oak Ridge, Tennessee, a leading developer of Web-based applications. Ray has over 30 years' experience in information systems and has been involved in many aspects of systems design, deployment and management ranging from "big iron" to desktops.

rayt@qsystems.net

LISTING 7

```
<CFSET NUMROWS=EVALUATE(QUERYNAME & "ROWS.NUMBERROWS")>
 <CFSET COLUMNNAMES=EVALUATE(QUERYNAME & ".COLUMNLIST")>
 <CFSET RESULT=ARRAYCLEAR(COLUMNNAMEARRAY)>
 <CFSET COLUMNNAMEARRAY=LISTTOARRAY(COLUMNNAMES)>
 <CFSET RESULT=ARRAYSORT(COLUMNNAMEARRAY,"TEXTNOCASE")>
 <CFSET COLUMNNAMES=ARRAYTOLIST(COLUMNNAMEARRAY)>
 <CFSET QUERYCOLUMNS=LISTLEN(COLUMNNAMES)>
```

LISTING 8

```
 <CFFILE OUTPUT="<CFSET TIMEFILESTARTED=NOW()>">
 <CFFILE OUTPUT="<CFQUERY NAME=""#QUERYNAME#""
DATASOURCE=""##SOURCEDB##"">">
 <CFFILE OUTPUT="   SELECT * FROM #DBNAME#">
 <CFFILE OUTPUT="</CFQUERY>">
 <CFFILE OUTPUT="<CFQUERY NAME=""#QUERYNAME#ROWS"" DATA-
SOURCE=""##SOURCEDB##"">">
 <CFFILE OUTPUT="   SELECT COUNT(*) AS NUMBERROWS FROM
#DBNAME#">
 <CFFILE OUTPUT="</CFQUERY>">
 <CFFILE OUTPUT="<CFSET QUERYROWS = #QUERYNAME#ROWS.NUMBER-
ROWS>">
 <CFFILE OUTPUT="<!--- COLUMNS TO PROCESS = #QUERYCOLUMNS# ---
>">
 <CFFILE OUTPUT="<CFOUTPUT QUERY=""#QUERYNAME#"">">
 <CFFILE OUTPUT="   <CFQUERY DATASOURCE=""##DESTINATIONDB##""
NAME=""#INSERTNAME#"">">
 <CFFILE OUTPUT="       INSERT INTO #OUTTABLE# (">
```

LISTING 9

```
 <CFLOOP INDEX="COLUMNNAME" LIST="#COLUMNNAMES#">
   <CFSET COLUMNCOUNT=COLUMNCOUNT+1>
   <CFIF COLUMNNAME EQ "COMMENT">
     <CFSET NAMEFIXCOUNT=NAMEFIXCOUNT+1>
     <CFIF COLUMNCOUNT EQ QUERYCOLUMNS>
     <CFFILE ACTION="APPEND" FILE="#PROCESSFILE#" OUTPUT="
COMMENTS)">
   <CFELSE>
     <CFFILE ACTION="APPEND" FILE="#PROCESSFILE#" OUTPUT="
COMMENTS,">
   </CFIF>
```

LISTING 10

```
<CFFILE OUTPUT="       VALUES (">
 <CFSET COLUMNCOUNT=0>
 <CFLOOP INDEX="COLUMNNAME" LIST="#COLUMNNAMES#">
   <CFSET COLUMNCOUNT=COLUMNCOUNT+1>
   <CFIF LISTFINDNOCASE(DATEFIELDSLIST,COLUMNNAME) GT 0
     <CFSET AUTOFIXCOUNT=AUTOFIXCOUNT+1>
     <CFFILE OUTPUT="           <CFIF
LEN(TRIM(#QUERYNAME#.#COLUMNNAME#)) EQ 0>">
     <CFIF COLUMNCOUNT EQ QUERYCOLUMNS>
     <CFFILE OUTPUT="              NULL)">
   <CFELSE>
     <CFFILE OUTPUT="              NULL,">
   </CFIF>
   OUTPUT="           <CFELSE>">
   <CFIF COLUMNCOUNT EQ QUERYCOLUMNS>
     <CFFILE .. OUTPUT="
TO_DATE('##DATEFORMAT(#QUERYNAME#.#COLUMNNAME#,""MM/DD/YYYY"")##
##TIMEFORMAT(#QUERYNAME#.#COLUMNNAME#,""HH:MM"")##','MM/DD/YYYY
HH24:MI'))">
          <CFELSE>
     <CFFILE OUTPUT="
TO_DATE('##DATEFORMAT(#QUERYNAME#.#COLUMNNAME#,""MM/DD/YYYY"")##
##TIMEFORMAT(#QUERYNAME#.#COLUMNNAME#,""HH:MM"")##','MM/DD/YYYY
HH24:MI'),">
   </CFIF>
   <CFFILE OUTPUT="           </CFIF>">
```

CODE LISTING

The code listing for this article can also be located at www.ColdFusionJournal.com

It's All Very **Personal**

BY
BEN FORTA

It's a big Web out there... but personalization can help foster customer loyalty

Personalization is the hottest trend in Web application development and for CF developers implementing basic personalization isn't difficult at all. In this month's column I'd like to explain what personalization is, why you should implement it and how to go about doing so.

Why Personalize?

To understand why personalization is so important, let me throw some statistics at you:

- By January 1998 2.5 million domain names had been registered worldwide, a total that more than doubled to 6 million by January 1999, and then more than doubled again to over 13 million by January 2000.
- By the end of 1998 there were over 800 million pages available on the public Internet – more than double the number that was available one year earlier.
- Contrary to popular belief, the Web is not used primarily for entertainment; in fact, 83% of Web sites are business related.
- There are an estimated 25,000 megasites – category-killer Web sites that attract most of the traffic and users.

What all this means is that sticking out from the crowd is becoming very difficult. But attracting an audience is only part of the problem; a bigger problem is *keeping* the users you worked so hard to attract. There is little loyalty on the Web. Users switch search engines and online stores all the time, primarily because it's so easy to do so. And that's dangerous: after all, the key to a successful Web site is traffic – in particular, repeat traffic.

So how do you retain users? The answer is to make their experience such a memorable and important one that your site becomes an integral part of their online life – something they can't live without (or at least don't want to). And that's where personalization comes in.

What Is Personalization?

As the word suggests, personalization is a mechanism by which users' online experience can be customized and tailored specifically to their personal needs. It allows you to provide users with an online experience that's uniquely theirs. In this way you can ensure they derive real value from their visit, a value proposition that translates into repeat visits.

Personalization is used in many ways (you've probably used it yourself without paying much attention to it):

- **Portal sites** allow you to specify what type of news interests you.
- **E-commerce sites** track buying patterns and make recommendations based on them.
- **Financial sites** let you track your own portfolio.
- **Travel and weather sites** let you specify your location so as to provide useful information.

> ***Note:*** Allaire has announced the acquisition of Open Sesame – a leading profiling and personalization technology designed to deliver highly accurate and intelligent personalized customer interaction. This technology will be integrated into a future version of Spectra.

The common thread in these examples is that the same site looks and behaves differently based on who's looking at it. That's personalization.

Explicit Personalization

The simplest form of personalization is *explicit* personalization. This usually involves prompting users for what they want to see or know. Typically, users create a profile of themselves by filling in a questionnaire of some kind that lists all available options. (Users also expect to be able to update those selections subsequently as needed). The profile is stored in a database and can be subsequently used to filter generated pages.

This form of personalization works well for:

- Areas of interest (for example, products or services)
- Color, menu and interface customization
- Language and locale customization (just over 50% of the estimated global online population is English-speaking; Japanese, Spanish, Chinese and German each account for over 5% of the non-English-speaking population).

Explicit personalization is highly accurate. After all, users state what they want and they get it. It's also easily implemented (you could use simple databases and ColdFusion

queries), easily updated and maintained, and is very inexpensive (no special software needed).

It's also terribly unintelligent. Users will see only what they selected and not what they might be interested in (even though they were unaware of it).

Implicit Personalization

Implicit personalization happens without users even being aware of it. They go about their business as usual – meantime, their activities are tracked in the background. Exactly what's tracked, and at what level of granularity, is up to you. You might decide to track only actual purchases or you might also want to track products looked at but not purchased. You determine what information is valuable and what that value is. Once collected, the data is stored in a database and special software can use it to perform analysis and return results.

This form of personalization works well for:

- Suggestions and recommendations (the "you liked book *A*: 9,000 other people who liked that book also liked book *B*, so you might like it too" model)
- Predictions of which products and services will be popular and which ones won't
- Targeted advertising campaigns

Implicit personalization is inherently inaccurate, although statistically the accuracy improves as the volume of collected data grows. Obviously, if you only have data on a dozen users you have a margin of error far greater than if you had data on hundreds of thousands (or millions) of users. It's also highly intelligent, and presents incredible opportunities that would be impossible without it.

The big downside is cost. You'll need to buy special software to perform the analysis and make suggestions and predictions. There are several vendors with offerings in this space (and no, I am not going to make any recommendations) – none of which are cheap. Nor is writing the code that interacts with it.

Summary

It's a big Web out there…and it keeps getting bigger. To survive and thrive, you need to keep your users happy. Personalization can help foster customer loyalty, which in turn translates into happy users, which in turn translates into repeat users.

Explicit personalization is easy, so start there. Implicit personalization is more complex and expensive but (for the right site) is well worth the investment. And ColdFusion can be used for both. (Spectra users are luckier as Spectra comes with the building blocks for implementing personalization right out of the box.)

The bottom line is: personalization is here, it's hot, and it's changing users' expectations. Don't fight it, give it a try – and let your users take it all very personally.

ABOUT THE AUTHOR

Ben Forta is Allaire Corporation's product evangelist for the ColdFusion product line. He is the author of the best-selling ColdFusion 4.0 Web Application Construction Kit *and its sequel,* Advanced ColdFusion 4.0 Development *(both published by Que), and he recently released* Sams Teach Yourself SQL in 10 Minutes.

BEN@FORTA.COM

Watchfire

www.watchfire.com

REVIEWED BY
DAVID SCHWARTZ

Synergy 2.0 by evolutionB

VITALS

Synergy 2.0: *evolutionB, formerly Catouzer*

Web: *www.evolutionB.com/cfdj*

Address: *Vancouver, Canada (HQ), Toyko, London, Boston*

Test Environment: *AMD Athlon 600 MHz test server running NT 4, IIS 4 and ColdFusion 4.5.*

Pricing: *2.0 Evaluation Version is downloadable free*

Languages: *Can be run in five languages - English, Spanish, Japanese, French and German*

Synergy operates beautifully with ColdFusion

Building rich Web-based software is fun, exciting…and can take too much time. Most clients have the same core software needs: secure, browser-based infrastructure, integration with existing systems and scalability to meet future needs. A Web application platform capable of all this would save developers tens of thousands of coding hours. Many products claim to be the "we got it all" solution. Is Synergy 2.0 from evolutionB the real thing? It exceeds most developers' core needs out of the box and scales from MS Access to SQL Server/Oracle. Interested in saving time and releasing better software? This is the e-business framework for you.

What's Synergy?

On the surface Synergy 2.0 from evolutionB, formerly Catouzer, is a suite of nine real-world office/business applications. In the new Web-enabled world I believe these applications have already become central to any company. Is there any company that doesn't use e-mail? Is there any company that doesn't share files or documents, either internally or with the outside world? If your company doesn't do these things, or your clients don't, you may as well turn off your PC and go back to sleep.

Dig below that surface and you'll find firm, scalable foundations supporting these applications – foundations strong enough to build on. At the center of the applications lies a solid security interface for user authentication – thereby addressing your number one concern in this age of sensitive corporate information on the Web. Also included with Synergy 2.0 is a powerful API that allows you to create your own products as well as modify the existing suite. Sound too good to be true? Let's go!

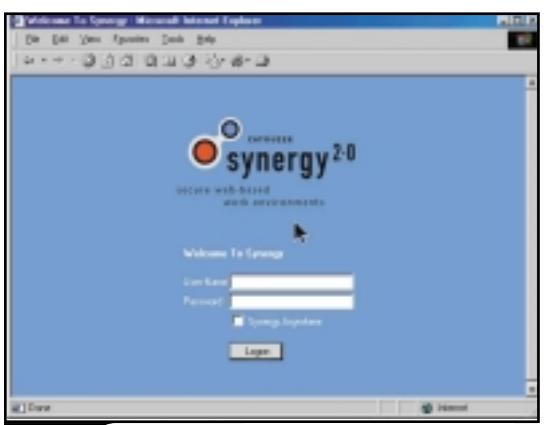


FIGURE 1: The login screen

Installation

I installed Synergy 2.0 on an AMD Athlon 600 MHz test server running NT 4, IIS 4 and ColdFusion 4.5. Installation was quick and easy. No problems – just the way I like it. Once installed, it ran fast. After prompting me, the existing version of Microsoft Data Access Components (MDAC) was upgraded automatically. The test server had several ODBC data sources prior to the Synergy install. I tested them all. Again, no problems.

Configuration

Upon completing the install I was prompted to start Synergy, which launches a browser-based login form (see Figure 1). I knew immediately that the product was on the right track: that is to say, 100% Web-based. After typing in the default administrator username and password Synergy prompts for a new password. Enter the new password, confirm and you're off! It takes about two seconds to add users and start using the product. In fact, in 10 minutes or less you could be in full swing. The admin form is really clear and simple to use and all options are presented in a logical hierarchical format. Overall, I got the impression that a lot of time was spent *planning*, an all-too-often forgotten phase of software development.

And They're Off....

Since Synergy's backbone consists of nine complete, Web-based applications that save developers many thousands of hours of planning and coding, I decided to put them all to the test. For this review I'll just cover the e-mail and document management apps.

I began with e-mail. We use Microsoft Exchange Server v5.5. It's always been a pain using <cfpop> with Exchange. This was my chance to see if Canadians only play hockey, eh! After 30 seconds of configuring my e-mail account within the Synergy admin page (yes! As I said, it's 100% browser-based), I was looking at my inbox. I opened mail with attachments, sent new mail, received and deleted. What an incredibly easy-to-

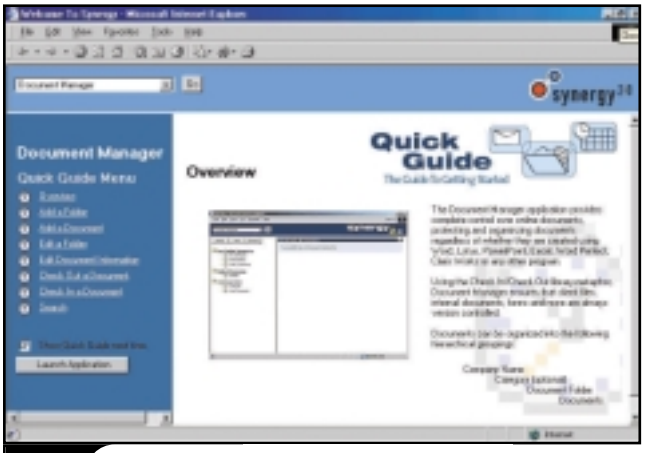


FIGURE 2: Document manager Quick Guide menu

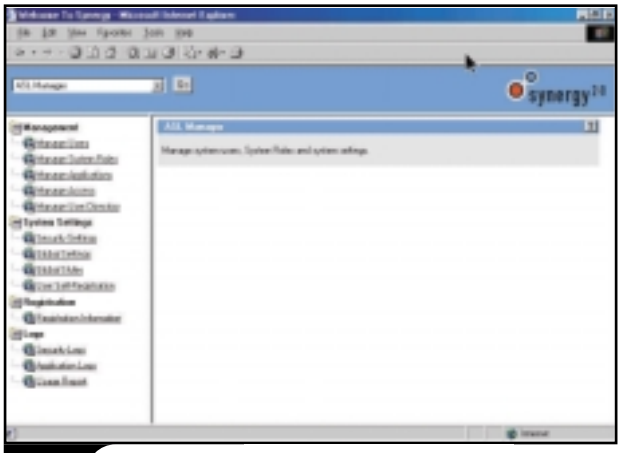


FIGURE 3: Application services layer of Synergy 2.0

use interface, the best I've seen in any browser-based product. Any developer can see the value of getting this functionality straight out of the box.

Then I moved on to document management (see Figure 2). Most systems allow you to upload, search and view documents – been there, done that. But Synergy adds two key features. First, users are able to create their own document folders on the fly. Incredible. When you upload a document you can store it logically – and next time around you'll actually find it! Second, there's a document checkout feature. If you've checked out a document, other users can't access it. (Reminds me of the video rental store: I can't rent *Pokemon* if it's already been rented.) This is critical when dealing with content that should be modified by only one person at a time. Bad for me when my son wants to rent *Pokemon*…but a good system for document security.

Documentation and Help

I'm old-fashioned when it comes to hard-copy paper manuals. An online manual just isn't the same as having a book at my side while I'm developing. (In the '80s software vendors used to brag about the number of pages in their manuals. Ah, the good old days.) My evaluation copy came with two binders, both extremely well written, clear and to the point. Developers will love the detailed 200+ page Language Reference. Of course, if you want to be hip, all documentation can also be found in PDF files.

The online help system is equally impressive. The entire system is stored in a database – no need for hundreds of HTML templates. The help editor lets you add topics and text to the help database. The help system is usually the last piece a developer creates; what with deadlines and user changes, it's often hard to complete. The Synergy help editor will change that: it's so fast and easy, you won't have any excuses. Users will love it.

Bookmarks, a full-featured sample application, is a great starting point for beginners. By studying and experimenting with the fully commented cfm files, you can see how to write a complete application. I was working with a very early release of Synergy and encountered an error loading this sample. So I called tech support, left a message and was called back within the hour by "Jason." Support was excellent. He quickly diagnosed the error and guided me to the evolutionB Web site, from which I downloaded a free patch/update. Many software vendors have eliminated free telephone tech support. Thanks, Jason! Within five minutes I was up and running again. I activated the Bookmarks application. New applications appear automatically on the Synergy main menu – a nice feature. All users have immediate access to it.

Addition and Subtraction

What if you want to add new features, create your own application or remove existing features? The Synergy application developers kit (ADK) is a custom tag-based language with over a hundred tags. The custom tags provide a tremendous amount of functionality. For example, <CF_CZASL_AUTHENTICATE> will do all the legwork necessary to log in a user. If you were to write your own login routine, it would likely require 100–200 lines of code. Forget about it! Synergy operates beautifully with ColdFusion. You can still use any standard CF tag or write your own. Or you can continue to use standard HTML. Synergy has a complete reference on HTML guidelines for integration with your own code. I experimented with several new features. Imagine my surprise to find software that works as advertised!

Security

Security is the number one concern of corporate America vis-à-vis the Web. Synergy's user authentication and access level scheme will allay the fears of even the biggest skeptics (see Figure 3). Before any users are allowed access to any feature, their identity is validated and all subsequent activity is logged in the database. You can create usergroups, assign users to the groups and assign functionality based on the group. Suppose you had a hundred users, with 10 of them in the accounting department. You'd create a group called Accounting, place the 10 users in that group and grant it access to financial documents. All 10 users would then be able to access the documents, so there's no need to go to each user and change his or her individual access rights.

Conclusion

In short: (1) I love it, (2) stop wasting time reinventing the wheel and (3) download an evaluation, it's free!

ABOUT THE AUTHOR

David Schwartz is president of Array Software Inc., a New Jersey-based software company that creates global data-driven Internet and intranet Web sites using ColdFusion, Oracle, MS SQL Server and Java. David has been developing turnkey custom database software for 13 years.

DS@ARRAYONE.COM

XML DevCon 2000

www.xmldevcon2000.com

Avoiding Unwanted Session Timeouts

How to create a TimeoutWarning custom tag

BY DANIEL JEAN

Even the most productive and effective application can be rendered useless by a poor user interface. Before we start, I want you to consider, for example, the following frustrating scenario....

Joe is a salesperson at a company that uses a Web-based online expense report system. It's the end of the month, so he logs into the system and starts filling out his expense report. Many fields need to be filled out, and in the middle of entering the data he goes searching for receipts and other paperwork. For security reasons the system is set to timeout his session if 15 minutes go by without any "activity." After working on his expense report for 25 minutes and having his session timeout without his knowing it, he presses the save button. The application has timed out and he's forced back to the login screen.

Joe isn't quite sure what just happened, so he logs in again and goes to his expense report. Of course, none of his work was saved and he curses at the "stupid program." He starts to redo his expense report, searches through a mound of paperwork and again fills out dozens of fields. This time it takes him only 16 minutes but when he presses the save button the system has again timed out his session and he's forced back to the login screen.

Angry and dejected, he calls Jane in the IT department. She quickly walks him through saving an expense report (taking less than 15 minutes) and of course it works just perfectly. Joe tries in vain to explain to her that he tried to save his expense report twice and both times it wouldn't save. When Jane gets off the phone she thinks to herself, "Typical stupid user." Joe is thinking "Typical stupid program." Frustration all round! (See Figure 1.)

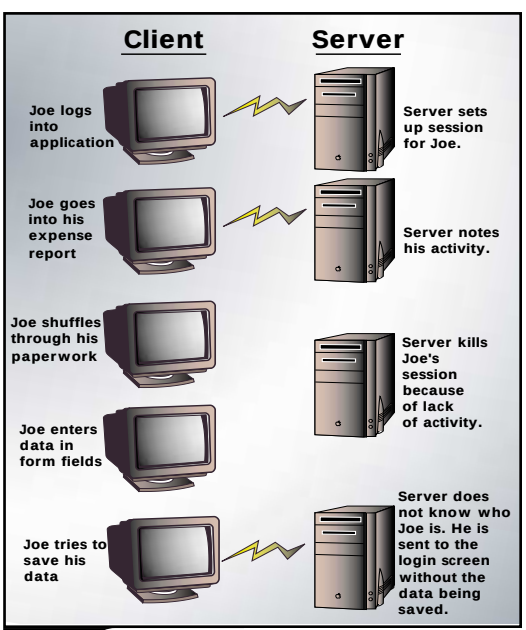


FIGURE 1: The timeline to frustration

The Problem

Developing applications for a secure intranet or extranet almost always leads to the idea of a session. Since HTTP is a stateless protocol, the ability to authenticate a user with a login is often tied to sessions. ColdFusion makes much of this easy for developers with its session-scoped variables. But they leave a security hole: if Joe logs in so his status is saved in a session variable then leaves for lunch without logging out, anyone can come in and access information using his login.

The accepted solution to this problem is to timeout the session. This means that the system terminates the session and forces Joe to log in again if there hasn't been any activity for a specified number of minutes. The problem with this solution is that activity must be server-side activity since the session is tracked on the server. By this criterion, filling out form fields doesn't count as "activity" since it doesn't include any communication with the server.

It would be nice if we could somehow warn Joe that he is about to timeout and offer him an opportunity to save his work. CF can't do this on its own since it's a server-side scripting language. Action may be taking place on the client side that the server and CF are not aware of.

JavaScript: setTimeout

JavaScript is a scripting language that runs on the client's browser and works well in conjunction with CF. Javascript has a powerful function called setTimeout (don't let the name fool you, it has nothing to do with the session timeout we've been discussing). The setTimout function allows you to run a piece of JavaScript at a specified interval:

```
setTimeout(JavascriptCode, Timeinterval);
```

The JavascriptCode is often specified as a function and the Timeinterval is specified in milliseconds. The following code will run the ShowHello function in 5,000 milliseconds (five seconds):

```
<SCRIPT Language="Javascript">
  function ShowHello()
  {
    window.alert("Hello, world!");
  }
  setTimeout('ShowHello()', 5000);
</SCRIPT>
```

We defined a function called

Shift4 Corporation

www.shift4.com

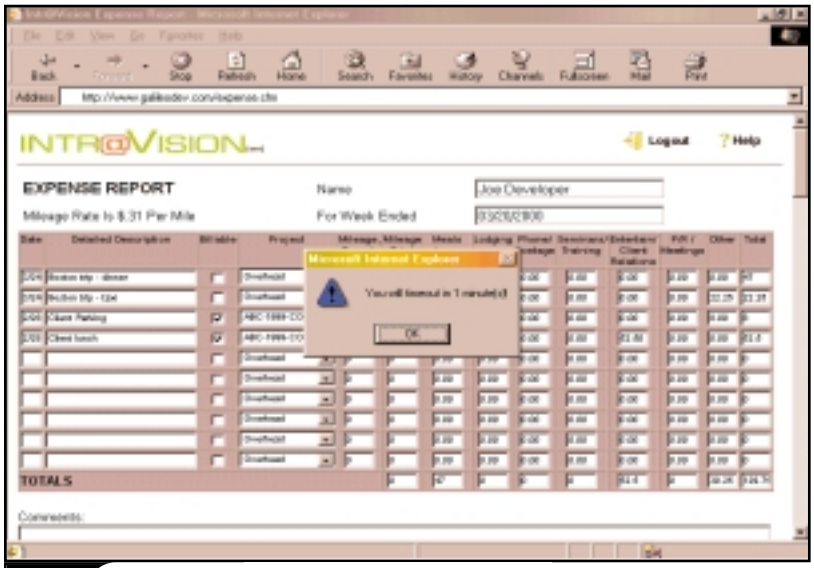


FIGURE 2: Expense report with Timeout warning

ShowHello and then use the setTimeout function to run the ShowHello function in five seconds. Neat, but the real power of the setTimeout function is that you can have the specified function reset the Timeout function so that the specified function is called continually.

```
<SCRIPT Language="Javascript">
  function ShowHello()
  {
    window.alert("Hello, world!");
    setTimeout('ShowHello()', 5000);
  }
 setTimeout('ShowHello()', 5000);
</SCRIPT>
```

After the first five seconds the ShowHello function runs, and pops up an alert that says, "Hello world."

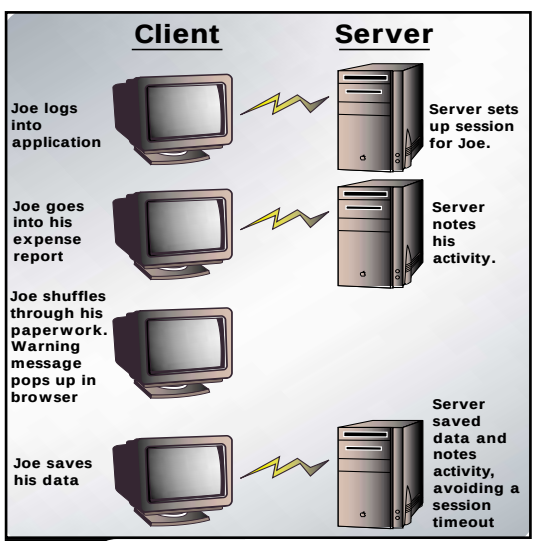


FIGURE 3: The timeline to bliss

After you press the OK button, the ShowHello function resets the setTimeout function to run itself again in another five seconds. This continues, with the result that every five seconds an alert pop ups. This works just like a timer. Hmmm, it seems we could use this same technique to help solve Joe's problem.

The Solution

By using a combination of JavaScript and CF, we can develop a custom tag that warns users that they're about to timeout, giving them a chance to save their work and refresh the page before they lose their information. It will have three inputs or attributes:

- *The number of minutes that the system is set to timeout a session*
- *The number of minutes ahead of time that we want to warn the user (the threshold)*
- *The actual warning message that will be displayed*

Using the <CFPARAM> tag, we define the attributes of the custom tag and set a default for each of them:

```
<CFPARAM Name="Attributes.TimeOutMinutes"
         Default="30">
<CFPARAM Name="Attributes.Warning-
Threshold"
         Default="2">
<CFPARAM Name="Attributes.Warning-
Message"
         Default="You will timeout in
#Attributes.WarningThreshold#
minute(s)!">
```

The next step is to set some global JavaScript variables. We need to know the time the current page was first accessed, as well as the threshold time. Using time in JavaScript is much easier if everything is converted to milliseconds, so the first thing we do is get the current Date/Time and convert it to milliseconds. Then we convert the Attributes.TimeOutMinutes to milliseconds by multiplying by 60,000 and then adding the result to the current time. This gives us the number of milliseconds that represents the exact time before the current session will timeout. We also convert to milliseconds the Attributes. Threshold, which is in minutes.

```
<CFOUTPUT>
<SCRIPT Language="Javascript">

// Global variables
StartTime = new Date();
StartMils = Date.parse(StartTime.toLo-
caleString());
TimeOutMils = StartMils + (60000 *
      #Attributes.TimeOutMinutes#);
ThresholdMils = 60000 *
      #Attributes.WarningThreshold#;
```

The next step is to create a CountDown function that will run every 10 seconds and check if it has reached the given threshold. We already learned how we can use JavaScript's setTimeout function to create a timer. The first line of this function clears JavaScript's Timeout mechanism just to be safe. Remember, although the names are similar, JavaScript's Timeout functions have nothing to do with the session timeout that we're dealing with. They are simply functions that we use to setup and run a piece of code every *x* number of milliseconds. We'll use this later to run CountDown every 10 seconds.

Next, we get the current time and convert it into milliseconds to calculate how many milliseconds are left until the timeout occurs.

```
function CountDown()
{
  clearTimeout('timerLoop');
  CurrentTime = new Date();
  CurrentMils = Date.parse(Current-
Time.toLocaleString());
  RemainingMils = TimeOutMils - Cur-
rentMils;
```

Once we have the number of milliseconds until timeout, we can compare that to the threshold milliseconds. If the number of milliseconds until timeout is less than the threshold milliseconds, then we display the warning message. If we haven't reached the threshold then we use JavaScript's setTimeOut function to run our CountDown function again in 10 seconds, thus:

```
  if(RemainingMils < ThresholdMils)
    window.alert('#Attributes.Warn-
ingMessage#');
  else
    timerLoop = setTimeout('Count-
Down()', 10000);
}
```

Finally, we have to remember to run the CountDown function for the first time to set everything in motion and start the timer. Normally, you would call it in the <BODY> tag's onLoad event, but since we're writing a custom tag, we don't want to rely on the tag's user to remember to add it to the <BODY> tag. So we call it with inline JavaScript before closing the script, as shown here:

```
// run CountDown for the first time
CountDown();
</SCRIPT>
</CFOUTPUT>
```

Save the preceding code as c:\cfusion\customtags\TimeOutWarning.cfm and you can use the tag by simply calling <CF_TIMEOUTWARNING>. For testing, add <CF_TIMEOUTWARNING WarningThreshold="1" TimeOutMinutes="2"> to a CF page. About a minute after loading the page, a warning will pop up, "You will timeout in 1 minute(s)!" (See Figure 2.)

Conclusion

As developers, we must always be conscious of user interface issues. As the scenario at the beginning of the article shows, user interface is the difference between an application that's usable and accepted and one that's unusable and the cause of frustration. Often the most productive and effective application is rendered useless because of a poor user interface. This custom tag is one way that you can help improve the user's experience and it only takes one line of code to implement.

Now let's revisit our original example (see Figure 3). This time Joe logs in and begins to fill out his expense report. While shuffling through his receipts, he hears a beep. Looking at the screen he sees a warning message and realizes he's about to timeout. He presses the save button and continues shuffling through his receipts. Once he's done, he submits his expense report for approval and he can now actually get some work done. He no longer thinks "stupid program." Instead he appreciates the way his expense check is processed within three days.

Throughout all of this, Jane in IT is able to play Solitaire without interruption.

ABOUT THE AUTHOR

Daniel Jean, CTO and cofounder of Galileo Development Systems, a provider of Web-based software solutions for the consulting and contract staffing industries, has worked in the client/server and Web development industry for 10 years.

DJEAN@GALILEODEV.COM

Using <CF_WHATEVER>

Use custom tags to have ColdFusion your way

BY JERRY BRADEN-BAUGH

I'm always looking for shortcuts – the smarter way out. Take Web page navigation: suppose you have five or six pages of content and you want each page to have links to all of the others (see Figures 1 and 2).

Let's have a look at three possible solutions for implementing this:

Solution 1: *The Lesson in Futility*

If HTML is a cool new acronym you just learned, you might consider custom hard-coding links to the other Web sites in each page. That is, you might open each content page and manually type in the code required for all the links.

Final Analysis: Effective, but about as pleasant (and as wise) as drinking bleach.

Solution 2: *A Step in the Right Direction*

The next possible workaround is to hard-code the solution for one page, then copy and paste that code into all the other pages, changing only the code for the one link to reflect all the other pages.

Final Analysis: Definitely effective, but painful to manage if changes are necessary. Besides, where's the glamour?

Solution 3: *Now That's What I'm Talking About*

HTML is old hat, and you've hopped the ColdFusion train to Web-page paradise. In other words, use a CF custom tag to add and manage the links.

Final Analysis: Smart. Very smart. Easy to create. Easy to change. I hear babes dig it, too.

Seriously, though. That's what this is about: using ColdFusion custom tags, also known as modules. Custom tags are created by you to do whatever you want. They can be great timesavers and make your code much more modular and manageable. The first part of this column, "Custom Tag Basics," lays out the basics of custom tags. In the second part, "A Working Example," I run through a simple custom tag I created so that you can try it yourself. The last section, "Guarding Your Secrets," shows you how to encrypt your custom tags if you're interested in keeping everyone from viewing them. Okay, let's do it.

> "Calling a custom tag in your code is easy: just add CF_ to the name of the file containing your code

Custom Tag Basics

Creating and using custom tags is based on two simple rules: (1) decide where the file containing the custom code will reside, and (2) know the syntax for calling the tag.

File Location

Custom tags, just like ColdFusion templates, bear the .cfm extension. The code inside doesn't look much different either. You can use any CFML tag and function you want. However, custom tags are generally stored in the \Cfusion\Custom directory. Placing templates here makes them available to all your CF applications.

Actually, you don't have to store them in that directory. There are times when you may not want to. You can also place the templates in the same directory as the template calling the custom tag. In fact, this is the first place ColdFusion looks when a custom tag is called. The difference is, only templates in that directory can call that custom tag. This separation provides global and local scope for custom tags.

Custom Tag Syntax

Calling a custom tag in your code is easy: just add CF_ to the name of the file containing your code. For example, if your custom code were in a file named CoolNewCode.cfm, your custom tag would look like this:

```
<CF_CoolNewCode>
```

Provided that CoolNewCode.cfm is correctly located, you've just called your custom tag.

Passing and Referencing Parameters

Lots of ColdFusion tags expect you to provide extra information. For example, the CFQUERY tag requires you to specify an ODBC data source in the DATASOURCE attribute. Attributes work as name-value pairs, where DATASOURCE could be the name and "MyOfficeDataSource" could be the value. It's

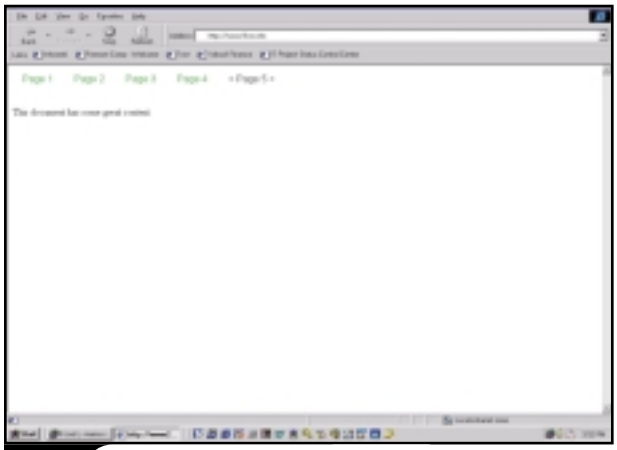


FIGURE 1: The exact same CFML that goes in this file…

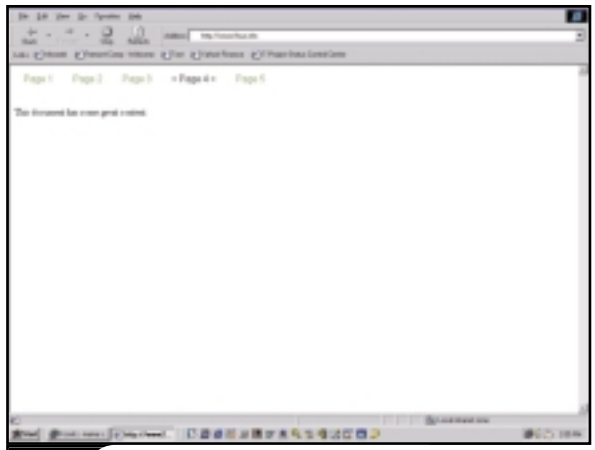


FIGURE 2: …goes in this and all the other linked files!

no different for passing parameters into custom tags. Here's an example.

```
<CF_CHANGETEXTCOLOR COLOR="Red">
If you have any compassion, you'll
make me red
</CF_CHANGETEXTCOLOR>
```

The COLOR attribute is set to Red. The code would then use that value to change the text color to red. It's that easy. Define as many attributes as you want.

Now, once you've passed in a parameter it'd be nice to know how to reference it in the custom tag code. Another easy one. Here's the code for some fictitious, three-line custom tag I just made up. Remember, it would be contained in a file called ChangeColorText.cfm.

```
<CFOUTPUT>
<font
color="#ATTRIBUTES.COLOR#">ThisTag.
GeneratedContent</font>
</CFOUTPUT>
```

This really simple code changes text color by referencing the parameter you pass in using the ATTRIBUTES structure and including it in the <font> tag. Whatever attributes you define in the call to the custom tag can be referenced this way. In this case it's thisTag.GeneratedContent (an explanation of this tag is up next).

Identifying Closing Tags

Some ColdFusion tags have closing tags (e.g., <CFQUERY></CFQUERY>, <CFLOOP></CFLOOP>, etc.). Others don't (like <CFSET> and <CFPARAM>). With custom tags, the choice is yours. If you decide you need one, as in the previous example, you'll need a way to reference the text between the tag pair. Every custom tag receives a special CF structure called *ThisTag*. Among the elements in the structure one is called *ThisTag.GeneratedContent*. It refers to all the text between the tags – in this case, "If you have any compassion, you'll make me red." You can read, write and otherwise manipulate this value. (Even make it red. Wow.)

What about <CFINCLUDE>? Good question. As you may know, using CFINCLUDE in your templates is another great way to introduce reusable code into your applications. You could load the same code from CoolNewCode.cfm anywhere in your template as easily as this:

```
<CFINCLUDE
TEMPLATE="CoolNewCode.cfm">
```

What's the difference between creating a custom tag and loading a template with CFINCLUDE?

First, there's the issue of variable scope. That is, variables defined in templates called via CFINCLUDE are visible to the calling template. The opposite is also true – templates called via CFINCLUDE can see variables defined in the calling template. With custom tags that's not the case. Custom tags have their own scope, so nothing declared within them is visible to the calling template. Variables defined in the calling template aren't visible to the code defined in the custom tag either.

Second, custom tags accept parameters via attributes. Parameters can change the output or performance of the custom tag. CFINCLUDE templates have no such quality. To do the same trick with CFINCLUDE, you have to set variables in the calling script and refer to these values again in the included template.

These differences in scope and parameter capability keep the code in calling templates and the code defined for custom tags separated.

A Working Example

Enough academics. Let's see how to put this stuff together. Remember my Web page navigation? Here's the code for the custom tag solution. The tag is called CF_Links, so we know its template is called Links.cfm and is located (unless otherwise indicated) in the \Cfusion\CustomTags\ directory. Here's the syntax of a typical call:

```
<CF_LINKS
 TEXT="Page 1,Page 2,Page 3,Page
4,Page 5"

URLS="one.cfm,two.cfm,three.cfm,four.
cfm,five.cfm"
 COLOR="Green"
 FONT="Arial"
 SIZE="16"
 >
```

The attributes simply define what link text to display, what URLs to point the links to and some other info on how to format the link text. The good news is, these few lines of neat code are all that's required to make a custom list of links. Better still, you use the *exact* same code in all your

other pages. ColdFusion will know how to eliminate the current page in view from the links provided in the URL's attribute. It does this by comparing the name of the template calling the tag with each of those in the URL's attribute.

The code is in Listing 1. Keep in mind that because of page space considerations I've eliminated many of the instructional comments from Links.cfm in this listing. After you get the download, make sure you read and understand them. More important, include similar comments in your custom tags.

Listing 1 is about a hundred lines of code. If I explained it line by line, this would be more like an O'Reilly book than an article. Fortunately, the code's pretty easy to follow: I'll just go over the highlights.

The first half of the script ensures that the tag provides the required attributes (TEXT and URLs), and that the number of link text entries and the number of link URLs provided are the same. If the calling tag passes those tests, then the link text and the URLs are each put into an array, like so:

```
<CFSET #TextArray# = #ListToArray
(ATTRIBUTES.TEXT)#>
```

and

```
<CFSET #URLArray# = #ListToArray
(ATTRIBUTES.URLS)#>
```

Guarding Your Secrets

Okay. Now you're ready to run off and revolutionize the industry with your own tags. It sure would be nice to keep others from stealing or even changing your code. The good news is, ColdFusion comes with a command-line tool called CFCRYPT that encrypts your custom tags, making them unreadable and unavailable for others to manipulate. You'll find the tool in the \Cfusion\Bin directory. Here's the syntax, straight from the program's help window:

cfcrpyt infile [outfile] [/r] [/q] [/h "This is a header"] /v "2"

- Code within brackets [ ] indicates optional parameters.
- cfcrypt calls the program.
- infile indicates the path to the name of the template or entire directory you want to encrypt.
- outfile indicates the path of the file or directory to which you want the encrypted file or directory to be written. If you don't include this parameter, the template in the infile parameter is overwritten. Be careful! Templates can't be decrypted.
- /r is a flag that forces recursive encryption of templates in subdirectories.
- /q is a flag that suppresses warning messages.
- /h allows you to specify a header to add to the top of each encrypted file.
- /v is a required flag that enforces the ColdFusion version of encryption.

You can also type cfcrypt h to view this help screen.

That's all we really need to start displaying run-of-the-mill Web links. But we still need to know how to display the link text. The next part of the script checks to see which optional attributes the calling tag has provided to set variables for later output. For example, to see what color the link text will be, the script checks for the COLOR attribute.

```
<CFIF
#IsDefined("ATTRIBUTES.COLOR")#>
 <CFSET #COLOR# = #ATTRIBUTES.COLOR#>
<CFELSE>
 <CFSET #COLOR# = "black">
</CFIF>
```

If the attribute isn't provided, a default value is set. This goes for all optional attributes. Keep in mind that your script will choke if you refer to an attribute that isn't defined in the calling tag. Make sure you check its existence using the IsDefined function.

Great. We're all ready, right? Yes and no. We know what the links will say, how they will look and to which URL each one will point to, but something is missing. There's no need to display the current page in view as a link. Why link a page back to itself? We need ColdFusion to single out the current template and display it in a form other than a link. We'll just use a handy CGI variable ColdFusion likes to call #CGI.SCRIPT_NAME#. This variable returns the path and filename of the current template. Have a closer look at the next few lines.

```
<CFSET #ReverseScriptName# =
#Reverse(CGI.SCRIPT_NAME)#>
<CFSET #ReverseCurrentTemplate# =
#Left(ReverseScriptName, Find("/",
ReverseScriptName) - 1)#>
<CFSET #CurrentTemplate# = #Reverse
(ReverseCurrentTemplate)#>
```

Remember that #CGI.SCRIPT_NAME# returns the path *and* filename of the executing script. We need only the filename, so we have to drop the path. For example, if #CGI.SCRIPTNAME# returns /ThisDir/ThatDir/OtherDir/OurFile.cfm, we need only OurFile.cfm. To get rid of that, we need to get rid of everything up to and including the last forward slash (/) before the filename. This can be done in several ways. I just reversed the value of #CGI.SCRIPT_NAME# and used the Left() function to shear off everything before the first / in the string. Reverse it again and you're in business. Let's run through that once using a path and filename of /examples/one.cfm:

```
Reverse the value to mfc.eno/selp-
maxe/
Use Left() to keep only mfc.eno
Reverse it back to one.cfm
```

You now have the filename you were looking for. Good deal.

Now for the last and final act – outputting the links. It all happens in a fairly small amount of code (see Listing 2).

The procedure is simple: iterate through the array of URLs (URLArray) using CFLOOP. As long as the current element isn't equal to the filename of the current template (CurrentTemplate), display it as a

ABOUT THE AUTHOR

Jerry Bradenbaugh is an independent Web consultant in Los Angeles, California. He's the author of The JavaScript Application Cookbook *(O'Reilly) and maintains HotSyte – The JavaScript Resource (www.serve.com/hotsyte).*

link and apply the formatting values in the form of an inline stylesheet. If it's the same as the current filename, print the link text as simple text to denote to the user that this is the current page being viewed. Several of the formatting attributes are still used for some consistency. It's pretty straightforward.

Covering All the Angles

This solution is really workable, but I failed to mention something. Maybe you already noticed it. The use of this custom tag assumes that *all* the pages linked together are in the same directory. That's why we dropped the entire path from the CGI variable. To understand this better, here's an example that won't work:

```
<CF_LINKS
 TEXT="Page 1,Page 2,Page 3,Page
4,Page 5"
URLS="../one.cfm,two.cfm,three.cfm,fo
ur.cfm,../five.cfm"
 COLOR="Green"
 FONT="Arial"
 SIZE="16"
 >
```

Sure, the links will be printed to the page, but one.cfm and five.cfm are in different directories from the other three. Eventually the links will point to the wrong files (if they even exist). You can get around this by further customizing Links.cfm and adding attributes that will accommodate files in other directories. Either way, it's still pretty cool and you should give it a try.

JERRYB@REDOAKTECH.COM

LISTING 1

```
<!--- If the #TEXT# list is defined, convert it to an array
--->
<CFIF #IsDefined("ATTRIBUTES.TEXT")#>
 <CFSET #TextArray# = #ListToArray(ATTRIBUTES.TEXT)#>
<CFELSE>
 <center><i><b>
 Error in <CF_Links>. No Link Text Defined.
 </b></i></center>
 <CFABORT>
</CFIF>
<!--- If the #URLS# list is defined, convert it to an array
--->
<CFIF #IsDefined("ATTRIBUTES.URLS")#>
 <CFSET #URLArray# = #ListToArray(ATTRIBUTES.URLS)#>
<CFELSE>
 <center><i><b>
 Error in <CF_Links>. No Link URLs Defined.
 </b></i></center>
 <CFABORT>
</CFIF>
<!--- Abort if the number of link text entries does not
equal the number of URLs --->
<CFIF #ArrayLen(TextArray)# NEQ #ArrayLen(URLArray)#>
 <CFOUTPUT>
 <center><i><b>
 Error in <CF_Links>. The number of link text
entries(#ArrayLen(TextArray)#)
 does not equal the number of URLs(#ArrayLen(URLArray)#).
 </b></i></center>
 </CFOUTPUT>
 <CFABORT>
</CFIF>
<!--- Set the values for the inline stylesheets --->
<CFIF #IsDefined("ATTRIBUTES.COLOR")#>
 <CFSET #COLOR# = #ATTRIBUTES.COLOR#>
<CFELSE>
 <CFSET #COLOR# = "black">
</CFIF>
<CFIF #IsDefined("ATTRIBUTES.FONT")#>
 <CFSET #FONT# = #ATTRIBUTES.FONT#>
<CFELSE>
 <CFSET #FONT# = "TimesRoman">
</CFIF>
<CFIF #IsDefined("ATTRIBUTES.SIZE")#>
 <CFSET #SIZE# = #ATTRIBUTES.SIZE#>
<CFELSE>
 <CFSET #SIZE# = "12">
</CFIF>
<CFIF #IsDefined("ATTRIBUTES.WEIGHT")#>
 <CFSET #WEIGHT# = #ATTRIBUTES.WEIGHT#>
<CFELSE>
 <CFSET #WEIGHT# = "normal">
</CFIF>
<CFIF #IsDefined("ATTRIBUTES.LINES")#>
 <CFSET #LINES# = #ATTRIBUTES.LINES#>
<CFELSE>
 <CFSET #LINES# = "none">
</CFIF>
<!-- Find the filename of the current executing template ---
>
<CFSET #ReverseScriptName# = #Reverse(CGI.SCRIPT_NAME)#>
<CFSET #ReverseCurrentTemplate# = #Left(ReverseScriptName,
Find("/", ReverseScriptName) - 1)#>
<CFSET #CurrentTemplate# = #Reverse(ReverseCurrentTemplate)#>
<!--- Print the links to the page --->
<CFLOOP INDEX="Idx" FROM="1" TO="#ArrayLen(TextArray)#">
    
 <CFIF #URLArray[Idx]# NEQ #CurrentTemplate#>
  <CFOUTPUT>
  <a href="#URLArray[Idx]#"
    style="color:#COLOR#;font-family:#FONT#;font-
size:#SIZE#px;font-weight:#WEIGHT#;text-decoration:#LINES#;">
    #TextArray[Idx]#</a>
  </CFOUTPUT>
  <CFELSE>
  <CFOUTPUT>
  &gt; <a style="font-family:#FONT#;font-size:#SIZE#px;font-
weight:#WEIGHT#;">
  #TextArray[Idx]#</a> &lt;
  </CFOUTPUT>
 </CFIF>
    
</CFLOOP>
```

LISTING 2

```
<!--- Print the links to the page --->
<CFLOOP INDEX="Idx" FROM="1" TO="#ArrayLen(TextArray)#">
    
 <CFIF #URLArray[Idx]# NEQ #CurrentTemplate#>
  <CFOUTPUT>
  <a href="#URLArray[Idx]#"
    style="color:#COLOR#;font-family:#FONT#;font-
size:#SIZE#px;font-weight:#WEIGHT#;text-decoration:#LINES#;">
    #TextArray[Idx]#</a>
  </CFOUTPUT>
  <CFELSE>
  <CFOUTPUT>
  &gt; <a style="font-family:#FONT#;font-size:#SIZE#px;font-
weight:#WEIGHT#;">
  #TextArray[Idx]#</a> &lt;
  </CFOUTPUT>
 </CFIF>
    
</CFLOOP>
```

CODE LISTING

The code listing for this article can also be located at www.ColdFusionJournal.com

JavaOne

www.java.sun.com/javaone/

A Beginner's Guide to ColdFusion

COLDFUSION FORMS

Part 2

Creating and building dynamic SQL statements and search screens

FROM THE BOOK
BY **BEN FORTA**

Creating Dynamic SQL Statements

Now that you're familiar with forms and how ColdFusion processes them, you can return to creating an employee search screen. The first screen enables users to search for an employee by last name. To begin, you need an INPUT field of type text. The field name can be anything you want, but using the same name as the table column to which you're comparing the value is generally a good idea.

The code in Listing 13 contains a simple HTML form, not unlike the test forms you created earlier in this chapter. [*Ed. Note:* Listings 1–12 and Figures 1–11 can be found in Part 1 (***CFDJ***, Vol. 2, issue 4).] The form contains a single text field called LastName and a submit button.

Save this form as C:\A2Z\SCRIPTS\12\EMPSRCH1.CFM and then go to it with your browser. Your display should look like the one shown in Figure 12.

The FORM ACTION attribute specifies which ColdFusion template should be used to process this search. The code ACTION="empsrch2.cfm" instructs ColdFusion to use the template EMPSRCH2.CFM, which is shown in Listing 14. Create this template and save it as C:\A2Z\SCRIPTS\12\EMPSRCH2.CFM.

The template begins with a CFQUERY tag that specifies the ODBC data source, the SQL statement to execute, and the name ColdFusion should use to refer to the results set.

The WHERE clause in Listing 14 contains a ColdFusion field rather than a static value. When ColdFusion parses templates, it replaces field names with the value contained within the field. If you search for all last names beginning with Sm, the code WHERE LastName LIKE '#LastName#%' becomes WHERE LastName LIKE 'Sm%'. If no search text is specified at all, the clause becomes WHERE LastName LIKE '%' – a wildcard search that finds all records.

You use a LIKE clause to enable users to enter partial names. The clause WHERE LastName = 'Sm' finds only employees whose last names are Sm; users with a last name of Smith are not retrieved. Using a wildcard, as in WHERE LastName LIKE 'Sm%', enables users to also search on partial names.

Try experimenting with different search strings. The sample output should look like the output shown in Figure 13. Of course, depending on the search criteria you specify, you'll see different search results.

Listing 15 contains the code for template C:\A2Z\SCRIPTS\EMPDTL2.CFM, which is an updated version of

TIP

When you're creating search screens, you can name your form fields with any descriptive name you want. When you're creating insert and update forms, however, the field name must match the table column names so that ColdFusion knows which field to save with each column. For this reason, you should get into the habit of always naming form fields with the appropriate table column name.

This article has been adapted from the second part of Chapter 12 of *ColdFusion 4 Web Application Construction Kit* by Ben Forta. Published by permission of Macmillan Publishers Ltd. and the author. Chapter 11 appeared in two parts in the February and March issues of ***ColdFusion Developer's Journal***; Part 1 of this article appeared in April. Chapter 13 will appear in forthcoming issues.

the employee detail template you created in the preceding chapter. The template is changed to enable users to send email directly to an employee by clicking his or her email address. You make this action possible by using the mailto identifier, which instructs the browser to open an email window so that the users can enter messages to be sent to the selected address. Look at the following code:

```
<A HREF="mailto:#EMail#">#EMail#</A>
```

Assume employee Kim Black is selected; the code expands into this:

```
<A HREF="mailto:kblack@a2zbooks.com">
  kblack@a2zbooks.com</A>
```

Anyone can click that address to send Kim email. In order to prevent errors, the code is conditional, based on an email address being present. <CFIF EMail IS NOT ""> evaluates to true only if the EMail field is not empty.

Building Truly Dynamic Statements

When you roll out your employee search screen, you are immediately inundated with requests. "Searching by last name is great, but what about first name or phone extension?" your users ask. Now that you have introduced the ability to search for data, your users want to be able to search on several different fields.

Adding fields to your search screen is simple enough. Add two fields: one for first name and one for phone extension. The code for the updated employee search screen is shown in Listing 16.

This form enables the users to specify text in one of three different fields, as shown in Figure 14.

You need to create a search template before you can actually perform a search. The complete search code is shown in Listing 17.

Understanding Dynamic SQL

Before you actually perform a search, take a closer look at the template in Listing 17. The CFQUERY tag is similar to the one you used in the previous search template, but in this one the SQL SELECT statement in the SQL attribute is incomplete. It does not specify a WHERE clause with which to perform a search, nor does it specify a search order. No WHERE clause is specified because the search screen has to support not one, but four search types, as follows:

- If none of the three search fields is specified, no WHERE clause should be used so that all employees can be retrieved.
- If a first name is specified, the WHERE clause needs to filter data to find only employees whose first name starts with the specified text. For example, if John is specified as the search text, the WHERE clause has to be WHERE FirstName LIKE 'John%'.
- If a last name is specified, the WHERE clause needs to filter data to find only employees whose last name starts with the specified search text. If you specify Sm as the last name, the WHERE clause must be WHERE LastName LIKE 'Sm%'.
- If you're searching by phone extension and specify 45 as the search text, a WHERE clause of WHERE PhoneExtension LIKE '45%' is needed.

How can a single search template handle all these search conditions? The answer is dynamic SQL.

When you're creating dynamic SQL statements, you break the statement into separate common SQL and specific SQL. The common SQL is the part of the SQL statement that you always want. The sample SQL statement has two: the SELECT FirstName, LastName, PhoneExtension, EmployeeID FROM Employees and the ORDER BY LastName, FirstName.

The common text is all the SQL statement you need if no search criteria are provided. If, however, search text is specified, the number of possible WHERE clauses is endless.

Take another look at Listing 17 to understand the process of creating dynamic SQL statements. The code <CFIF FirstName IS NOT ""> checks to see that the FirstName form field is not blank. This condition fails if no text is entered into the FirstName field in the search form, and any code until the next <CFELSE> <CFELSEIF> or </CFIF> is ignored.

If a value does appear in the FirstName field, the code WHERE FirstName LIKE '#FirstName#%' is processed and appended to the SQL statement. #FirstName# is a field and is replaced with whatever text is entered into the FirstName field. If John is specified as the first name to search for, this statement translates to WHERE FirstName LIKE 'John%'. This text is appended to the previous SQL statement, which now becomes the following:

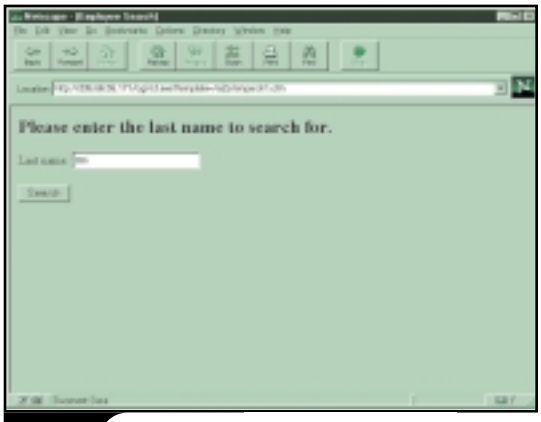


FIGURE 12: On the employee search screen, users can search for employees by last name only.

FIGURE 13: By building WHERE clauses dynamically, you can create different search conditions on-the-fly.

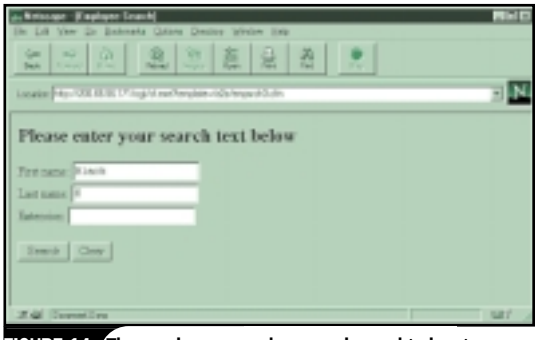


FIGURE 14: The employee search screen is used to locate employee records by name or a part thereof.

```
SELECT FirstName, LastName, PhoneExten-
sion, EmployeeID
FROM Employees WHERE FirstName LIKE
'John%'
```

All you need now is the ORDER BY clause. Even though ORDER BY is fixed and does not change with different searches, it has to be built dynamically because the ORDER BY clause must come after the WHERE clause, if one exists. After ColdFusion processes

FIGURE 15: You can use a single template with dynamic SQL to perform an infinite number of searches.

the code ORDER BY LastName, FirstName, the finished SQL statement reads as follows:

```
SELECT FirstName, LastName, PhoneExten-
sion, EmployeeID
FROM Employees WHERE FirstName LIKE
'John%'
ORDER BY LastName, FirstName
```

Note: You cannot use double quotation marks in a SQL statement. When ColdFusion encounters a double quotation mark, it thinks it has reached the end of the SQL statement. It then generates an error message because extra text appears where ColdFusion thinks there should be none. To include text strings with the SQL statement, use only single quotation marks.

Similarly, if a last name of Sm is specified as the search text, the complete SQL statement reads as follows:

```
SELECT FirstName, LastName, PhoneExten-
sion, EmployeeID
FROM Employees WHERE LastName LIKE
'Sm%'
ORDER BY LastName, FirstName
```

The code <CFIF FirstName IS NOT “”> evaluates to false because #FirstName# is actually empty; ColdFusion therefore checks the next condition – <CFELSEIF LastName IS NOT “”>. This condition evaluates to true because a last name value is provided. ColdFusion then processes the next line, the CFSQL statement, and builds the required SELECT statement.

Processing Search Results

Now that the template is complete and all the code to build dynamic SQL is in place, try performing a search. First try a search without specifying any search criteria at all. Your browser display should look like the one shown in Figure 15.

Notice that the number of employees found is displayed at the top of the table. ColdFusion stores the number of rows retrieved with a query in a field called RecordCount. To determine how many rows the Employees query retrieves, you use the code Found #Employees.RecordCount# Employees; just like any other field, ColdFusion replaces it with the actual value.

The RecordCount field is also useful for returning messages or options if a search returns no data at all. For example, the following code displays a message that states that no records are retrieved:

```
<CFIF Employees.RecordCount IS 0>
No employees located, try changing your
search criteria.
</CFIF>
```

Try performing different searches via the search form. You can search on any of the fields to retrieve specific records.

Concatenating SQL Clauses

Now try entering text in both the first name and last name fields. What happens? The answer is, not much at all. The dynamic SQL code processes only the first search criterion it encounters. If it finds text in the First Name field, it does not even check the Last Name and Extension fields. If you follow the flow of the <CFIF> statement, you see that as soon as any condition is true, either a <CFIF> or a <CFELSEIF>, ColdFusion ceases processing the condition and jumps to the </CFIF> tag.

How can you search on more than one field? Creating a WHERE clause with an unknown number of conditions is more complicated than the WHERE clauses you created earlier. If a particular condition is the first condition in the clause, you need to precede it with WHERE. If it is not the first condition, you need to precede it with an AND. Of course, you have no idea if any fields will be specified, nor do you know which fields will be specified, if any. How do you construct a dynamic SQL statement like this?

One solution is shown in Listing 18. The SQL SELECT statement has a dummy WHERE clause – WHERE EmployeeID = EmployeeID. This condition checks to see if the EmployeeID in a retrieved record matches itself, which, of course, it always does. For example, when the employee with ID 6 is retrieved, the database checks to see if 6=6. Because only one column is being checked and the condition involves no complicated arithmetic or subqueries, the check has almost no performance penalty. If no search criteria are specified, the WHERE clause remains WHERE EmployeeID = EmployeeID and all records are retrieved.

There are three sets of <CFIF> conditions immediately after CFQUERY. Each one determines whether a specific search field is provided; if so, it is appended to the SQL SELECT statement. If Kim is entered into the first name field, the new WHERE clause is WHERE EmployeeID = EmployeeID AND FirstName LIKE ‘Kim%’. This clause correctly filters out only employees whose first name begins with Kim.

Now see what happens if a second field is specified. This example assumes B is entered in the Last Name field. The <CFIF LastName IS NOT “”> evaluates to true because LastName is not empty, and the AND LastName LIKE ‘#LastName#%’ code is added from the LastName filter to the WHERE clause. The new WHERE clause reads WHERE EmployeeID = EmployeeID AND FirstName LIKE 'Kim%’ AND LastName LIKE ‘B%’. If you execute this search, only Kim Black is found.

To try this code example, save Listing 18 as C:\A2Z\SCRIPTS\12\EMPSRCH5.CFM. You also need to modify the FORM ACTION attribute in template EMPSRCH3.CFM so that it specifies EMPSRCH5.CFM as the destination template instead of EMPSRCH4.CFM.

After you create the template, use your browser to try performing different combinations of searches. You’ll find that this new search template is both powerful and flexible. Indeed, this technique for creating truly dynamic SQL SELECT statements will likely be the basis for some sophisticated database interaction in real-world applications.

Creating Dynamic Search Screens

The more power and flexibility you give your users, the more they want. Now they want to search by department, too. For example, they may

LISTING 13: EMPSRCH1.CFM – Code Listing for Employee Search Screen

```
<HTML>

<HEAD>
<TITLE>Employee Search</TITLE>
</HEAD>

<BODY>

<H2>Please enter the last name to search for.</H2>

<FORM ACTION="empsrch2.cfm" METHOD="POST">

Last name: <INPUT TYPE="text" NAME="LastName"><BR>
<P>
<INPUT TYPE="submit" VALUE="Search">

</FORM>

</BODY>

</HTML>
```

Listing 14: EMPSRCH2.CFM – Using a Passed Form Field in a SQL WHERE Clause

```
<
CFQUERY
DATASOURCE="A2Z"
NAME="Employees"
>
SELECT FirstName, LastName, PhoneExtension, EmployeeID
FROM Employees
WHERE LastName LIKE '#LastName#%'
ORDER BY LastName, FirstName
</CFQUERY>

<HTML>

<HEAD>
<TITLE>Employee List</TITLE>
</HEAD>

<BODY>

<CENTER>

<TABLE BORDER=5>

<CFOUTPUT QUERY="Employees">
<TR>
<TD>
<A HREF="empdtl2.cfm?EmployeeID=#EmployeeID#">#LastName#,
#FirstName#</A>
</TD>
<TD>
Ext. #PhoneExtension#
</TD>
</TR>
</CFOUTPUT>

</TABLE>

</CENTER>

</BODY>

</HTML>
```

LISTING 15: EMPDTL2.CFM – Passing Parameters to Templates

```
<
CFQUERY
```

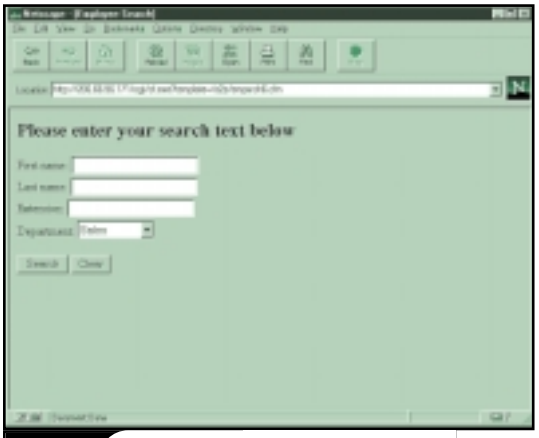


FIGURE 16: Search forms should use a mixture of input types to create a user-friendly interface.

want to display all the Sales department personnel or find a particular employee by specifying a name within a specific department.

The search screen is simple enough to create. A list of departments is the perfect place to use a list box. You can create a list box with the HTML <SELECT> tag and then list all the departments as <OPTION> tags within.

Before you modify the search template, however, remember that you're creating data-driven applications. You don't want to have to enter the departments manually in the list box. Rather, you want the list box to be driven by the data in the Departments table. This way, you can acquire changes automatically when departments are added or if a department name changes.

The code in Listing 19 demonstrates a data-driven form. The CFQUERY at the top of the template should be familiar to you by now. It creates a result set called Departments, which contains the ID and name of each department in the database.

The body of the form is essentially the same as the one you created in Listing 16, with the exception of the new DepartmentID field. DepartmentID is a list box that displays the names of all the departments. The <SELECT> tag creates the list box, and it is terminated with the </SELECT> tag. The individual entries in the list box are specified with the <OPTION> tag, but here that tag is within a CFOUTPUT block. This block is executed once for each row retrieved by the CFQUERY, creating an <OPTION> entry for each one.

The code in Listing 19 demonstrates the process of building data-driven forms. The SELECT NAME attribute (don't confuse it with the SQL SELECT statement) contains the name of the field, which is the same as the column in the Employees table that you need to compare against. The CFQUERY block creates the individual options, using the ID field as the VALUE and the department name as the description. When ColdFusion processes department 2, the Sales department, the code <OPTION VALUE="#id#">#Department# is translated into <OPTION VALUE="2">Sales.

Also notice that you need to include a blank <OPTION> line in the list box. Remember that list boxes always must have a selection. Save Listing 19 as C:\A2Z\SCRIPTS\12\EMPSRCH6.CFM.

The completed search screen is shown in Figure 16.

Now you have one last thing left to do: You need to modify the search template to include the new DepartmentID field. The code in Listing 20 is updated to include one more possible WHERE condition. The code AND DepartmentID = #DepartmentID# adds the DepartmentID field only if it is specified in the search screen, thereby enabling users to search on as many or as few fields as they want.

The final WHERE condition in Listing 20 is different from the prior listing in two ways. First, the condition checks for equality instead of LIKE. The value you're comparing is an ID value – a number – and numbers either match or don't match. Wildcards don't apply to numbers.

Second, no quotation marks appear around the field identifier. The prior three fields all require quotation marks because they are text values. DepartmentID is a number, and numbers don't need quotation marks around them. In fact, if you do put quotation marks around the number, you generate an ODBC error because you're comparing a numeric field to a string.

BEN@FORTA.COM

```
DATASOURCE="A2Z"
NAME="Employee"
>
 SELECT LastName,
 FirstName,
 MiddleInit,
 Title,
 PhoneExtension,
 PhoneCellular,
 PhonePager,
 EMail
 FROM Employees
 WHERE EmployeeID = #EmployeeID#
</CFQUERY>

<HTML>

<CFOUTPUT QUERY="Employee">

<HEAD>
<TITLE>#LastName#, #FirstName# #MiddleInit#</TITLE>
</HEAD>

<BODY>

<H1>#LastName#, #FirstName#</H1>

<HR>

Title: #Title#
<BR>
Extension: #PhoneExtension#
<BR>
Cellular: #PhoneCellular#
<BR>
Pager: #PhonePager#
<BR>
E-Mail: <CFIF EMail IS NOT ""><A
HREF="mailto:#EMail#">#EMail#</A></CFIF>

</BODY>

</CFOUTPUT>

</HTML>
```

LISTING 16: EMPSRCH3.CFM – Employee Search Screen

```
<HTML>
```

```
<HEAD>
<TITLE>Employee Search</TITLE>
</HEAD>

<BODY>

<H2>Please enter your search text below</H2>

<FORM ACTION="empsrch4.cfm" METHOD="POST">

First name: <INPUT TYPE="text" NAME="FirstName"><BR>
Last name: <INPUT TYPE="text" NAME="LastName"><BR>
Extension: <INPUT TYPE="text" NAME="PhoneExtension">
<P>
<INPUT TYPE="submit" VALUE="Search">
<INPUT TYPE="reset" VALUE="Clear">

</FORM>

</BODY>

</HTML>
```

LISTING 17: EMPSRCH4.CFM – Building SQL Statements Dynamically

```
<
CFQUERY
DATASOURCE="A2Z"
NAME="Employees"
>
 SELECT FirstName, LastName, PhoneExtension, EmployeeID
 FROM Employees

 <CFIF FirstName IS NOT "">
  WHERE FirstName LIKE '#FirstName#%'
 <CFELSEIF LastName IS NOT "">
  WHERE LastName LIKE '#LastName#%'
 <CFELSEIF PhoneExtension IS NOT "">
  WHERE PhoneExtension LIKE '#PhoneExtension#%'
 </CFIF>

 ORDER BY LastName, FirstName

</CFQUERY>

<HTML>

<HEAD>
<TITLE>Employee List</TITLE>
</HEAD>

<BODY>

<CENTER>

<TABLE BORDER=5>

<CFOUTPUT>
<TR>
<TH COLSPAN=2>
<H3>Found #Employees.RecordCount# Employees</H3>
</TH>
</TR>
</CFOUTPUT>

<CFOUTPUT QUERY="Employees">
<TR>
<TD>
<A HREF="empdtl2.cfm?EmployeeID=#EmployeeID#">#LastName#,
#FirstName#</A>
```

```
</TD>
<TD>
Ext. #PhoneExtension#
</TD>
</TR>
</CFOUTPUT>

</TABLE>

</CENTER>

</BODY>

</HTML>
```

LISTING 18: EMPSRCH5.CFM – Employee Search Template That Concatenates WHERE Clauses

```
<
CFQUERY
DATASOURCE="A2Z"
NAME="Employees"
>
SELECT FirstName, LastName, PhoneExtension, EmployeeID
FROM Employees
WHERE EmployeeID = EmployeeID

<CFIF FirstName IS NOT "">
AND FirstName LIKE '#FirstName#%'
</CFIF>

<CFIF LastName IS NOT "">
AND LastName LIKE '#LastName#%'
</CFIF>

<CFIF PhoneExtension IS NOT "">
AND PhoneExtension LIKE '#PhoneExtension#%'
</CFIF>

ORDER BY LastName, FirstName

</CFQUERY>

<HTML>

<HEAD>
<TITLE>Employee List</TITLE>
</HEAD>

<BODY>

<CENTER>

<TABLE BORDER=5>

<CFOUTPUT>
<TR>
<TH COLSPAN=2>
<H3>Found #Employees.RecordCount# Employees</H3>
</TH>
</TR>

</CFOUTPUT>

<CFOUTPUT QUERY="Employees">
<TR>
<TD>
<A HREF="empdtl2.cfm?EmployeeID=#EmployeeID#">#LastName#,
#FirstName#</A>
</TD>
<TD>
Ext. #PhoneExtension#
</TD>
</TR>
</CFOUTPUT>

</TABLE>

</CENTER>

</BODY>

</HTML>
```

LISTING 19: EMPSRCH6.CFM – Data-Driven Employee Search Template

```
<
CFQUERY
DATASOURCE="A2Z"
NAME="Departments"
>
SELECT ID, Department
FROM Departments
ORDER BY Department
</CFQUERY>

<HTML>

<HEAD>
<TITLE>Employee Search</TITLE>
</HEAD>

<BODY>

<H2>Please enter your search text
 below</H2>

<FORM ACTION="empsrch7.cfm" METHOD="POST">

First name: <INPUT TYPE="text" NAME="FirstName"><BR>

Last name: <INPUT TYPE="text" NAME="LastName"><BR>

Extension: <INPUT TYPE="text" NAME="PhoneExtension"><BR>

Department:
<SELECT NAME="DepartmentID">
<OPTION>
<CFOUTPUT QUERY="Departments">
<OPTION VALUE="#id#">#Department#
</CFOUTPUT>
</SELECT>

<P>
<INPUT TYPE="submit" VALUE="Search">
<INPUT TYPE="reset" VALUE="Clear">

</FORM>

</BODY>

</HTML>
```

LISTING 20: EMPSRCH7.CFM – Final Employee Search Template

```
<
CFQUERY
DATASOURCE="A2Z"
NAME="Employees"
>
SELECT FirstName, LastName, PhoneExtension, EmployeeID
FROM Employees
WHERE EmployeeID = EmployeeID


<CFIF FirstName IS NOT "">
AND FirstName LIKE '#FirstName#%'
</CFIF>

<CFIF LastName IS NOT "">
AND LastName LIKE '#LastName#%'
</CFIF>
```

```
<CFIF PhoneExtension IS NOT "">
AND PhoneExtension LIKE '#PhoneExtension#%'
</CFIF>

<CFIF DepartmentID IS NOT "">
AND DepartmentID = #DepartmentID#
</CFIF>

ORDER BY LastName, FirstName

</CFQUERY>

<HTML>

<HEAD>
<TITLE>Employee List</TITLE>
</HEAD>

<BODY>

<CENTER>

<TABLE BORDER=5>

<CFOUTPUT>
<TR>
<TH COLSPAN=2>
<H3>Found #Employees.RecordCount# Employees</H3>
</TH>
</TR>
</CFOUTPUT>

<CFOUTPUT QUERY="Employees">
<TR>
<TD>
```

```
<A HREF="empdtl2.cfm?EmployeeID=#EmployeeID#">#LastName#,
#FirstName#</A>
</TD>
<TD>

Ext. #PhoneExtension#
</TD>
</TR>
</CFOUTPUT>

</TABLE>

</CENTER>

</BODY>

</HTML>
```

CODE LISTING

The code listing for this article can also be located at www.ColdFusionJournal.com

n 2000

on2000.con

Sharing the Wealth

A proposal for a 'ColdFusion repository' – a Web-based registry of CF tips and tutorials

BY HAL HELMS

For ColdFusion developers, this really is the best of times. Demand for developers is exploding, taking salaries and consulting rates along with it.

The early religious wars, waged over the matter of whether CF was a "real" programming language, have largely been won – not so much through reasoned disputation and determined appraisal as by dint of the sheer number of developers who have embraced ColdFusion.

Many of these developers come not from the traditional ranks of programmers but are nonprogrammers from business domains such as human resources, purchasing, customer service and operations. They've taken up CF for the simple reason that they need it to get their jobs done. This migration of nonprogrammers to the software development world provides clear testimony to the changing nature of the world economy.

Like all people new to a territory, these novice developers have urgent questions that require swift answers, and many of us CF "old pros" have responded with sites featuring various tips and tutorials.

This is a good start, but problems remain. Students of CF must now keep track of an ever-expanding list of sites that have valuable information, but are entirely separate from one another. Links to other sites are a step in the right direction, but they don't go far enough. Besides, Web site owners have their own agendas and are reluctant to send users to other sites (e.g., www.sun.com has a notable lack of links to www.microsoft.com).

What about a central repository? Surely we could have a single site that would house all tips and tutorials? But this won't work either. If Web authors are reluctant to link to others, they certainly won't line up to give away their intellectual property; nor should they – creating and maintaining these various individual sites takes a great deal of work.

A Modest Proposal

I've been thinking about this recently, fully aware that any solution would need to respect the wishes of Web site owners and ColdFusion students alike. It seems to me we can manage both if we think of the Web not as a collection of sites but as a network of resources. In fact, we can take a lesson from the most unlikely of places: shopping malls.

Years ago, stores fought for exclusive possession of what we might call "footprints" – the pre-Web version of "eyeballs," I suppose. This fierce rivalry was immortalized by Hollywood in the battle between two rival New York City department stores, Macy's and Gimble's, in the movie *Miracle on 34th Street*. In those benighted days, every store knew that there was a zero-sum battle for customers – one store's gain was another store's loss. Great marketing campaigns were waged, the terminology of war being adopted to reflect the philosophy of want and conquest.

Today, on the other hand, stores generally understand that customers aren't spoils of war. To prosper, companies nowadays must cater to, not capture, the customer – to make shopping easier, stores cluster themselves in configurations known by such names as "Restaurant Row" and "Car Alley." A further step in this transition has been the shopping mall: a central point where customers0-cum-rulers may rove throughout their domain and choose freely whom they buy goods and services from. Some have named this business strategy *coopetition*, a fusion of *cooperation* and *competition*.

Registering Tips and Tutorials

How does all this help us? I've devised a system that lets sites remain unique entities while at the same time

creating a shared environment for the good of the customer, just as competing merchants come together in the shared environment of a mall.

Our "mall" is a Web-based tips and tutorials registry. Sites that wish to share their information with others – and receive the increased traffic from that sharing – add a custom tag to each tutorial they wish to register. When activated, this tag registers the tip in a local central repository. Since the repository is realized as a WDDX packet saved as a file on the local server, such a repository would be a help to sites that don't want the administrative overhead of using a database.

Using WDDX lets us take the entire repository that has been effected as an array of structures and convert this into a string. Making arrays and structures persistent is difficult but strings are simple constructs and we can save them easily.

If you've been following the development of the Fusedoc documentation standard, you can see from the fusedoc in Listing 1 that the custom tag requires a single parameter – the URL for the tip to be registered – and allows for additional pieces of information that further define the tip. The last parameter, "registerToWeb," determines whether the tip should register itself with the central Web repository (more on this below).

Going Global

Tutorials that incorporate a call to this page are activated only if the tip page is invoked with a URL variable of "register," as in www.mySite.com/myTip.cfm?register=yes. When called in this way, the registration tag registers itself with the local site by placing each tip or tutorial into a saved pool of other tips. The parameters for each tip (such as keywords and title) can be

LISTING 1

```
Listing 1:
<!-- registerTip.cfm -->

<!---
|| Responsibilities: If caller.URL.register exists, I read in
a TipsRepository.cfm file. The tips repository is a WDDX
packet that, when de-serialized, becomes an array of struc-
tures. The structure keys match the parameters passed to me.
If no tips repository exists, I create a new 1D array.

 If the URL passed to me matches one in the tips repository,
I update the other attributes for this entry. Otherwise, I
create a structure that holds the parameters shown below. I
add this structure to the array and re-serialize the array
of structures and save the file. If "registerToWeb" is YES,
I register the tag with the central repository at the Team
Allaire site.

||
--> URL: a fully qualified URL address
--> [tipID]: a STRING. Defaults to CreateUUID()
--> [title]: a STRING. Defaults to "A ColdFusion Tip"
--> [synopsis]: a STRING. Defaults to ""
--> [keywords]: a comma-delim list. Defaults to ""
--> [author]: a STRING. Defaults to ""
--> [lastUpdate]: a STRING. Defaults to today.
--> [registerToWeb]: a BOOLEAN to determine whether to reg-
ister the tag in a web repository. Defaults to YES.
+++ [TipsRepository.cfm]: a FILE
||
END FUSEDOC--->
```

LISTING 2

```
<cfmodule
 template="registerTip.cfm"
 URL="www.houseoffusion.com/javaApplets.cfm"
 localRepository="TipsRepository.cfm"
 title="Integrating Java Applets into CFML"
 synopsis="With Java fully a mainstream technology, ColdFu-
sion developers need a way to tie Java applets into ColdFu-
sion development. This article lays a groundwork for this
and provides sample CFML code and a sample Java applet to
get developers started."
 keywords="Java,integration,applet,advanced technology"
 author="Michael Dinowitz"
 lastPublished="1/28/2000"
 registerToWeb="Yes">
```

LISTING 3

```
<!-- SampleClient.cfm -->

<!---
|| Responsibilities: I show an admittedly simplistic way to
use the Tips Registry on the Web. Note that these tips can
come from anywhere and will take the user to the site that
hosts them. This enables sites to share info without giving
away site users.
||
END FUSEDOC--->

<cfoutput>

<h2>Sample Client for the Tips Registry</h2>

<cfhttp method="GET" url="http://www.teamallaire.com/tutori-
als/TipsRegistry.cfm">
<cfwddx
 action="WDDX2CFML"
 input="#cfhttp.filecontent#"
 output="tipsList">

<table border="1" bgcolor="c0c0c0" width="600">
 <tr>
  <td><b>Title</b></td>
  <td><b>Synopsis</b></td>
  <td><b>Author</b></td>
  <td><b>Link</b></td>
 </tr>
<cfloop from="1" to="#ArrayLen(tipsList)#" index="i">
 <tr>
  <td>#tipsList[i].title#</td>
  <td>#tipsList[i].synopsis#</td>
  <td>#tipsList[i].author#</td>
  <td><a href="http://#tipsList[i].URL#">View tip</a></td>
 </tr>
</cfloop>
</table>
</cfoutput>
```

CODE LISTING

The code listing for this article can also be located at www.ColdFusionJournal.com

used by the site to provide search capabilities and organizational structure to the collection of tips.

That's very helpful. Nevertheless, we can do much better than this. By adding the capability for each tag to register itself in a Web repository, we can make the entire contents of that registry available to all sites that wish to make use of it. We get shared content of tutorials – surely a boon to new CF developers – without asking sites to give away their hard-won loyal users.

Suppose, as an example, that you're in the habit of frequenting the excellent site run by Michael Dinowitz, www.houseoffusion.com. You go there regularly because you like Michael's insightful articles and his writing style. However you may also wish to keep track of new tutorials at www.teamallaire.com/tutorials or in Duncan Campbell's fine zine at www.defusion.com.

Let's further say that Michael decides to write an article on "Integrating Java Applets with CFML." He would insert a tag, shown in Listing 2, into his article code.

My own effort is considerably feebler. I create a tutorial – aimed, apparently, at really young CF developers – on "Working with the Pokemon DTD in ColdFusion." I would add a tag to the tutorial as follows:

```
<cfmodule
 template="registerTip.cfm"
 URL="www.teamallaire.com/tutorials/pokemon.cfm">
```

Since I didn't specify any of the optional parameters such as title, synopsis and keywords, the registerTag.cfm code will provide generic default values for these, and both Michael's tutorial and my own will be registered with the Web repository. Under my scheme these two tutorials, together with all other tips that register themselves, would be available as links for Michael to post and for you to use. But they would also be available to Duncan, me and anyone else who would like to use them.

If you run the sample client in Listing 3, you will see the code connects to the Team Allaire site and retrieves the file containing the WDDX packet. The code listing shows a rudimentary table that displays the decoded information. The upshot is that all sites can share and receive tips and tutorials and no one site need be the "master" site.

But wait! The repository is held at www.teamallaire.com – doesn't that give the Team Allaire members some advantage? Not at all. What's there is available for all to use and the specifics of where it's hosted are really immaterial. (I can conceive of a further refinement where the materials are replicated on different sites, so that one downed site wouldn't spoil the program.) Rather, advantage goes to those people or companies inventive and energetic enough to create clear navigable structures for the growing compendium of tips.

Conclusion

I hope that other CF developers and site owners will adopt this plan, and that I've demonstrated that such cooperation is in everyone's best interest and that ColdFusion itself offers the technology to make such a sharing scheme simple and easy. "Joining" couldn't be simpler – you just download the custom tag available at www.teamallaire.com/hal.

Last, be sure to check out the following four sites that have helped test the system and are implementing the CF tips registry today:

- www.fusionauthority.com
- www.pacificgroup.com
- www.geocities.com/SiliconValley/Heights/1863/code.html
- www. webtricks.com

ABOUT THE AUTHOR

Hal Helms is a Team Allaire member living in Atlanta, Georgia. A frequent writer on ColdFusion and Fusebox, he also offers training and mentoring on these subjects.

HAL.HELMS@TEAMALLAIRE.COM

Calling All Custom Tags PART 2

There's probably more to using custom tags than you originally thought

BY CHARLES AREHART

In my **Journeyman** column last month (***CFDJ*** Vol. 2, issue 4) we began looking at how – and why – to call custom tags in different ways.

So far, we've learned that the CF_ approach will look first locally, then in the shared \cfusion\customtags directory, while the CFMODULE NAME= approach will look only in the shared directory.

But what if you *can't* place files in the shared customtags directory? Perhaps you work in a tightly controlled corporate environment where you have to get all sorts of permissions to place files in the shared directory or, worse, you're in a hosted environment where you're extremely limited in what you can do and they absolutely refuse to place any tags you'd like to use in the shared directory?

Well, it's good to know that at least one solution exists. You can place the intended tag in the local directory and execute it from there using the CF_ approach. For many folks, this will be a revelation. If you have refrained until now from creating custom tags or exploring and enjoying the vast benefits of preexisting custom tags available in the Allaire Developer's Exchange (www.allaire.com), then let me welcome you to this world! You're not forced to use the shared directory. Go to town!

But there's a catch. You now know that you can place the intended custom tag in your local calling template directory and you're good to go. But what if you have code in several subdirectories? If you place the custom tag in all those directories, it will work; but you're exposing yourself to the risks and challenges of keeping all those versions updated and in sync. Of course, if you have access to the shared customtags directory, just place it there. But what if you don't? Are you forced to make those copies and suffer the indignities of multiple redundant and possibly out-of-sync versions?

A Template by Any Other Name...

Well, as always, Allaire has got you covered. If you need to execute a custom tag in someplace other than either the local directory or the shared directory, you want to use the CFMODULE TEMPLATE= approach, completing the triumvirate.

This approach allows you to name the specific location – anywhere on the CF server – in which the custom tag's file can be found. This could be used to name some alternative "global" directory that all can see or, more useful, it could name some "subglobal" directory that all the templates under the directories of a related group of applications could see. They could all point to this one directory, thus eliminating the redundancy problem.

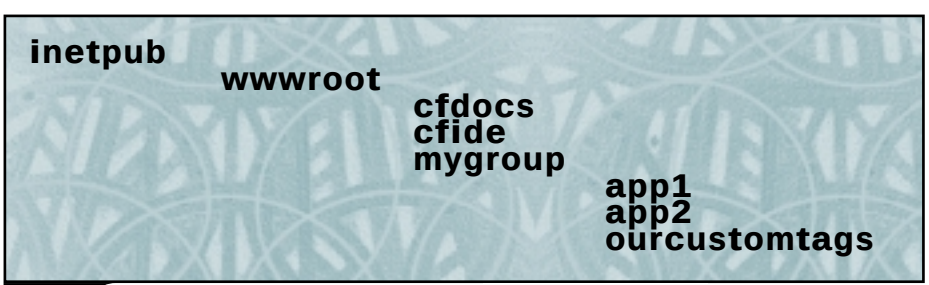


FIGURE 1: Directory structure

And the coolest thing for those in a hosted or otherwise tightly controlled environment is that this "subglobal" directory, as I've termed it, could be placed anywhere in the directory structure that you *do* have control over, making it great for sharing a custom tag among all your (and only your) applications.

As with the CFMODULE NAME= approach, there are actually two different ways to use the TEMPLATE= approach. The first is for you to specify the physical path to the custom tag relative to the location of the caller. At the end of the last article, we introduced an example custom tag called format.cfm. If that were placed in the current directory with the template calling it, we'd be using it in what I have referred to as a "local" mode. It's intended to be used by templates in the same directory.

Using the CFMODULE TEMPLATE approach, we could call this local custom tag with the following syntax:

```
<CFMODULE TEMPLATE="format.cfm"
TEXT="#fname#">
```

Note that the difference (besides use of the TEMPLATE attribute) is that the .cfm file extension is now used. Now it may appear that this is just a lot of extra work with little benefit, and in fact in this specific use you really haven't gained much over the CF_ approach. (Actually, there is one difference: it will now execute *only* the local version and not any shared one. But that's not usually a valuable benefit.)

But the key benefit comes when you specify some relative path information before the name of the cus-

tom tag template file name. For instance, consider a directory structure as depicted in Figure 1. Assume we have several programs in a "mygroup" directory (under our Web root, where our Web files are located – we're *not* discussing here subdirectories under the shared customtags directory). If there were no further subdirectories under APP1 or APP2, then all the templates in those directories could call upon a shared customtags directory, depicted in the figure as "ourcustomtags."

Assuming our format.cfm file were placed in this directory, we could easily call it from any of our "mygroup" subdirectory templates using relative naming conventions that are common to folks familiar with using the operating system command prompt or even HTML tags like <IMG SRC> and <A HREF>. We specify a path above ours using ".." notation, which refers to our parent. And we can refer to a sibling (a directory at the same level as our own under our parent) as "..\sibling".

In our case, if our code is running in, say, the APP1 directory, we could call the format.cfm in the mygroup \ourcustomtags directory using the following syntax:

```
<CFMODULE
TEMPLATE="..\ourcustomtags\format.cfm"
TEXT="#fname#">
```

Notice the "..\" preceding "ourcustomtags\". These tell CF to go up to our parent and down to the ourcustomtags directory to find format.cfm. (In the last article, we learned that our fictitious format.cfm custom tag was designed to accept a parameter called "text," and we were passing it the value of a local variable called "fname.")

For those in hosted environments, this is A Good Thing, because you can use this approach to share a given custom tag among all the apps in your server directory. The only drawback is that as soon as we start to put subdirectories under our APP1 or APP2 directories, for example, the relative paths for programs placed in those subdirectories become a little clumsy, since we now need to specify a grandparent reference, as in "..\..\ourcustomtags\format.cfm". This can become difficult to read and maintain.

My Kingdom for a Mapping

Like the cavalry coming to our rescue, Allaire foresaw this need and created an elegant – if often misunderstood, little used and rather poorly documented – solution called "directory mappings."

A directory mapping is a setting made in the ColdFusion Administrator that maps a virtual name to a physical directory path (see Figure 2). That is, we could create a virtual name in the administrator to refer to our "mygroup\ourcustom-tags" directory with a single word. Let's say we called it "mygrouptags". (That is, let's say we had the administrator create a mapping of the work "mygrouptags" to the physical directory"c:\inetpub\wwwroot\mygroup\ourcustomtags".)

Now we could refer to the tag in that location using this new mapping. The key to using mappings in the TEMPLATE= approach (as well as with CFINCLUDE, which can also use mappings) is to specify a "\" and

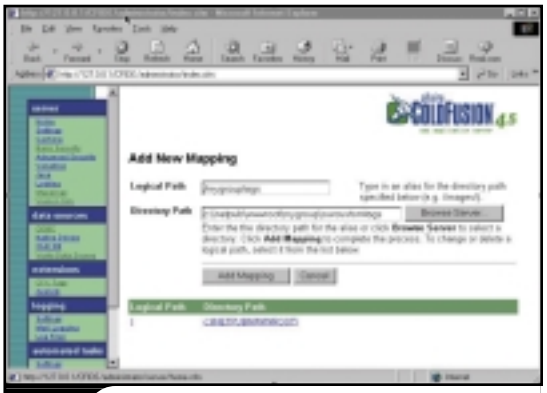


FIGURE 2: Adding a new directory mapping in CF

the name of the mapping before the name of the custom tag file name. So our example would become:

```
<CFMODULE TEMPLATE="\mygrouptags\for-
mat.cfm" TEXT="#fname#">
```

Pretty nifty! And the beauty of this approach is that it doesn't matter where in our entire mygroups directory structure our calling template is located. We can even move code around within the directories and the call to the custom tag will *always* use this syntax to refer to the location of the custom tag. Not only that, another great benefit of directory mappings is that should anyone decide to move the "ourcustomtags" directory anywhere else in our tree structure – or really anywhere on the server – our code will again not have to change. Only the mapping in the administrator needs to change.

As with the CFMODULE NAME= approach, this call will not find a local version of the custom tag. It will find and use only that version in the named directory. And we can't really have different "versions" of the tag unless we create and use different mappings.

Furthermore, this approach opens up a bit of a security hole. You are making this mapping in the administrator; thus, technically, others on the server could become aware of and access the custom tag in your directory more easily. Of course, that's a thorny issue that deserves its own entire article. Mapping doesn't spawn the security problem in and of itself; it does, however, arguably increase exposure to the problem. There is a solution, if you're willing to make the commitment to using Advanced Security features. We'll address it briefly in a moment since it is indeed an important issue.

Getting to the "Root" of the Problem

As an aside, I said that the first TEMPLATE= approach allowed you to specify a relative path and that the second approach allows specification of a virtual path via a mapping. You may be wondering if there's a way simply to specify an absolute path. In fact, you may wonder why that "/" designation in front of the mapping name doesn't mean "look in the root." That's an interesting issue.

In fact, there is a predefined mapping called simply "/" defined in the administrator by default. And it does map to the Web root of the Web server on which CF is installed – such as c:\inetpub\wwwroot on an IIS server installed on the c: drive. So you can, in fact, use that to specify subdirectories off the root in a form that looks like an absolute path, such as "\myapp\customtags\format.cfm". This means, of course, that in some situations there is no real need to create a separate mapping for your particular directory.

Just be aware that things can get confusing in a hosted environment, where your "server" is actually just a "virtual server" under the directory tree of the entire Web server. In such a case, this "absolute" path defined in the administrator is relative to the root of the CF server, not relative to your own "virtual root."

One last note. I said that this TEMPLATE= approach allows access to a custom tag located anywhere on the CF server. If you can point to it with a relative or mapped path, you can run it. Actually, there's no limitation that says the path specified even needs to be to a file on the same server. Since we're specifying the physical path to the file, it can really be anywhere that the server can find it, whether on a mapped drive or a UNC path. As long as the server can see it, we can place and run our custom tags there.

Security Matters

When we describe the ability for custom tags to be placed and executed from shared directories, we open up the possibility that the tag can be used in an unauthorized or at least unexpected way.

If we're in a secure environment, perhaps we may have a tag that performs a function to return data that may be restricted. In a hosted environment, perhaps we have paid for a commercial custom tag and must restrict access for licensing issues.

Advanced Security to the Rescue

Again, Allaire has provided an answer, but the price to be paid is that you must enable what's called Advanced Security. This is a mode that became available in Release 4.0, but it's not the default mode of operation (which is called Basic Security).

Using Advanced Security requires considerable forethought and some rather lengthy setup procedures to create and administer users, resources and policies to map users to resources. Though the benefit is very likely well worth the price, it seems many have been slow to make the move. Maybe this is due to the rather sparse documentation or to the rather daunting user interface for administering it. Fortunately, both have been improved in 4.5. There's also coverage of the topic in Ben Forta's book *Advanced ColdFusion Development*, published by Que.

As for custom tags, they are just one of the many "resources" that can be protected in Advanced Security. Protectable resources actually include all manner of things such as files, directories and data sources, as well as applications, tags, functions and, indeed, custom tags.

You owe it to yourself and your organization to look into and take the time to implement Advanced Security. The benefits are manifold. And in a secured or hosted environment, it's imperative (though hosted environments are among the last to consider advanced security, paradoxically).

By the way, some presume that it's a feature available only in the Enterprise version of ColdFusion. But it's not. All that's missing from the Professional version is "Server Sandbox" security – which, while it may seem to encompass what we're describing here, in fact doesn't. Do check out Advanced Security.

Tales from the Crypt

Another matter that encompass-

Sys-Con Media

www.sys-con.com

es security is the matter of encrypting (or encrypted) custom tags. This is a process by which the CFML code within a template is manipulated in such a way (via an Allaire-provided utility) that the code is no longer human-readable. The CF Server can read and execute it, but no one looking at the file can see what the code is doing.

Encrypting tags is a valuable option for secured environments, and especially for commercially sold tags. The process is actually just an encoding, not an encryption, of the tag. In fact, as of Release 4.5, Allaire now calls it *encoding*, and the utility, formerly called CFCRYPT, is now called CFENCODE. For more information see the Allaire manual *Developing Web Applications with ColdFusion*.

You're No Custom Tag

Before concluding this discussion of custom tags, I'd be remiss not to mention a couple of related aspects of reusing code that technically doesn't involve custom tags but that may be of use to you as you consider whether and how to reuse code. I'll just mention these in passing for you to consider.

CFEXECUTE

The new CFEXECUTE tag in 4.5 allows you to execute any process on the server machine, including shell scripts, services, executables and batch files. See the *Language Reference* for more information and an example of executing the Windows NetStat network monitoring program. CFEXECUTE works in UNIX environments as well, and you can imagine that it would open the door to reusing existing user-written code also.

CFHTTP

If you want to execute some code that's not on your machine but is accessible via a Web site, whether it's a ColdFusion routine or even a CGI script, ASP page or simple HTML file, you can do that with the CFHTTP tag. This allows you to execute any page at any URL (Web site address) that can be reached from your server. You can even pass parameters to the page, such as to emulate formfields being passed to a form action page, or to pass expected cookie, cgi or URL variables to the page being executed. The result of the page execution is returned to your program in a variable called cfhttp.filecontent.

CFOBJECT

CFOBJECT allows you to call methods within COM, CORBA and Java objects. This opens the door to your calling programs written in languages other than CFML and even running on other servers than the CF server. This can be a tremendous opportunity for reusing legacy code within your organization or for simply choosing a more suitable environment for executing some process.

Java Processing

There are several other tags (and a function) for executing Java code. CFSERVLET executes a Java servlet on a JRun engine. (JRun is a technology for running JavaServer Pages and Servlets, which Allaire acquired last year.) CFSERVLET is designed to be consistent with the HTML <SERVLET> tag. Again, if you have an investment in such programs, you can reuse that code now within ColdFusion.

The createobject function allows you to create and use JAVA CFXs or objects, and by extension Enterprise JavaBean (EJB) objects. (As of 4.5, EJBs can be called via CFOBJECT as well.)

Other features you may come across for running Java objects include CF_ANYWHERE (an older form of executing servlets) and CFX_J (an older way to run Java objects now superseded by Java CFXs).

Final Thoughts

I've talked about all manner of possibilities and opportunities for using ColdFusion tags. There are a few closing notes to share that may be important.

First, I mentioned previously that you can find many (in fact, hundreds of) prebuilt custom tags at the Allaire Developer's Exchange (formerly known as the Tag Gallery). It is accessible from the Developer section of the Allaire site (www.allaire.com).

When you're working with custom tags, you may encounter an interesting problem that will arise if you move a custom tag from one directory to another (such as moving it from a local directory to the global one). If, before you move it, you execute a template that finds the custom tag in the first location, and then after you move the custom tag you execute that template again, you'll receive an error message indicating that the custom tag cannot be found. CF is still trying to find it in the original location, and even though it's been moved to a location where CF should be able to find it, it doesn't. It seems to be a template code-caching thing. The problem will be corrected if you restart the CF server process. I'm not aware of another way to resolve the problem, though I'm confident there may be one.

Also, as Ben Forta pointed out in his article "Preserve Precious Resources – Recycle" (***CFDJ***, Vol. 2, issue 2), the notion of paired custom tags and nested custom tags has also been enabled in Release 4. They're very useful in some situations, and they work just like the CF_ approach described above. Unfortunately, that means that they're not able to take advantage of some of the control offered by the CFMODULE approaches.

Finally, it's worth noting that with respect to all aspects of referring to file names when using custom tags, case sensitivity is significant in UNIX operating environments. The *Developing Web Applications with ColdFusion* manual indicates that custom tag names must be lower case on UNIX.

So that's it. It has taken me two months' worth of columns…practically a mini-novel! There's probably more to using custom tags than you originally thought. It's really great to know the power and possibilities available with them. These solutions can definitely come in handy in some environments. As always, check out the Allaire reference manuals and user guides to learn more and find real examples. But, most important, try things out yourself. There's no better way to learn!

ABOUT THE AUTHOR
Charles Arehart is an Allaire certified trainer and CTO of SysteManage, an Allaire partner. He contributes to several CF resouces and is a frequent speaker at user groups throughout the country.

CAREHART@SYSTEMANAGE.COM

Career Opportunities

User Group	Contact	e-mail	Phone
NORTH AMERICA			
Alabama-Birmingham	Jeff Bryant	Jeff.Bryant@SouthTrust.com	(205) 667-5204
Alabama-Huntsville	Tasia Malakasis	tasia.t.malakasis@syntegra.com	(256) 971-9104
Arizona-Scottsdale	Sean Tierney	sean@pubcrawl.net	480.421.1270
California-Bay Area	Nathan Dintenfass	nathan@changemedia.com	415 793 9304
California-Sacramento	TJ Downes	tj@jel.net	916-447-5463
California-San Diego	John Morgan	Johnmorgan@Bluestarcorp.com	(619) 622-1201
California-Southern	Leon Chalnick	lchalnick@advantasolutions.com	(562)-491-0212
Canada-Edmonton	Jeff Acker	jeff.acker@stjus.ecs.edmonton.ab.ca	(800) 422-9772
Canada-Ottawa	Shaun St. Louis	shauns@newforce.ca	613-599-5786
Canada-Toronto	Kevin Towes	ktowes@PangaeaNewMedia.ca	(416) 922-1600
Canada-Vancouver B.C.	Cameron Siguenza	cameron@catouzer.com	(604) 662-7551 X303
Colorado -Northern	Robi Sen	robi@granularity.com	(970) 266-9125
Connecticut-Stamford	Barry Hyman	bah@wspthomes.com	(203) 221-0828
Florida-South	Tony Schreiber	dune@cfsf.org	(561) 999-1575
Florida-Orlando		rae@raelee.com	407-658-5045
Florida-Tallahassee	Benjamin Bloodworth	ben@electronet.net	(850) 222-0229
Florida-Tampa	Dale Mulert	dale@eng-tech.com	(888) 382-1900 x58
Georgia-Atlanta	Cameron Childress	cameronc@mcrae.com	(770) 460-7277
Georgia-Atlanta	Seth Land	Sland@figleaf.com	(404) 995-6030
Illinois-Chicago	Bob Burns	bob-burns@mediaone.net	(630) 961-6261
Indiana-Indianapolis	Lisa Sturgeon	ltsturgeon@yahoo.com	(317) 786-7846
Maryland -Baltimore	Steve Drucker	sdrucker@figleaf.com	(202) 797-5477
Maryland-Bethesda	Michael Smith	Michael@teratech.com	(301) 424-3903
Massachusetts-Boston	Steve Casco	steve@ateaze.com	(781) 275-6171
Minnesota-Minneapolis	Dan Chick	dchick@energylevel.com	612.990.5191
Missouri-St. Louis	Mike Stevens	mike@psiwebstudio.com	(618) 624-6694
Montana-Great Falls	Greg Snortland	snorty@unlimiteddata.com	(406) 452-8290
Nebraska-Omaha	Mark Kruger	dtnmarkk@earthlink.net	
Nevada-Las Vegas	Brian Thornton	brian@fiax.net	(702) 310-3600
New Jersey-Newark	Cindy Pomarlen	cindy_pomarlen@hotmail.com	(973) 540-9672
New York-Albany	Thomas McKeon	thomas.mckeon@cgiusa.com	(518) 434-0294 x464
New York-Binghampton	Robert Hanno	robert.hanno@lmco.com	(607)741-3559
New York-NYC (advanced)	Michael Dinowitz	mdinowit@houseoffusion.com	(212) 647-8901
New York-NYC (basic)	Greg Narain	chairman@cfsig.org	(718) 768-3245
New York-Syracuse	Dave Santeramo	dsantera@zdnetmail.com	(607) 756-0393
North Carolina-Charlotte	Glynn Wallace	gwallace@carolina.rr.com	(828) 267-0013
North Carolina-Raleigh	Karen Andrews	kandrews@propoint.com	(919) 329-6434
Ohio-Columbus	Angelo McComis	Angelo@CompuServe.com	(614) 538-4539
Ohio-Cleveland	Jennifer Nagy	JNagy@newchanneltech.com	(330) 220-1558
Ohio-Mid Ohio Valley	Dave Hannum	hannum@ohio.edu	740-597-2524
Oklahoma-Tulsa	David Barnett	dfb@tdis.com	(918) 298-8949
Oregon-Portland	William Cupps	billc@dev-tech.com	(503) 294-4488
Oregon-Roseburg	Sharen Bland	sbland@douglasesd.k12.or.us	(541)957-4822
Pennsylvania-Pittsburgh	Kyle Jenkins	kjenkins@steelcitywebs.com	(724) 846-2250
Pennsylvania-Philadelphia	Chris Swansen	cswansen@etechsolutions.com	(610) 458-2460
Rhode Island-Providence	Michael Rowan	mrowan@castlerea.com	(401) 792-7017
South Carolina-Greenville	Patrick S. Little	patricklittle@home.com	(864) 244-8965
Tennessee-Memphis	Michael Kubicki	mkubicki@univ-solutions.com	(901) 737-3406
Tennessee-Nashville	James Pecora	jpecora@radsoltn.com	(615) 269-4493
Texas-Austin	Dave Williams	david.williams@ci.austin.tx.us	(512) 499-2835
Texas-Dallas	Patrick Steil	pmsteil@imailbox.com	(940) 321-6162
Texas-San Antonio	Chris Montgomery	monty@astutia.com	(210) 490-3249
Virginia-Charlottesville	Steve Nelson	m@secretagents.com	(804) 293-2060
Virginia-Hampton Roads	Raymond Camden	jedimaster@creativeis.com	(757) 318-7042
Virginia-Northern	Steve Drucker	sdrucker@figleaf.com	(202) 797-5477
Washington D.C	Bret Peters	Bpeters@figleaf.com	(202) 797-5477
Washington-Seattle	Shannon Smith	president@cfseattle.org	(206) 343-9445
Wisconsin-Milwaukee	Steve Booth	steve.booth@wepco.com	(414) 221-4303
INTERNATIONAL			
Asia; South East	Roydean Osman	Roydean@gtemail.net	
Auckland	Amanda Irvine	amanda.irvine@delta.co.nz	09-309 9400
Australia-Barton	Peter Tilbrooks	peter@safetyweb.com.au	
Australia-Melbourne	Marc Dimmick	marc@tved.net.au	613-9670-2055
Australia-Sydney	Brent Pearson	b.pearson@morganbanks.com.au	
Belgium	Francois Lamotte	f.lamotte@switchon.be	
Europe (Central)	Sven Slazenger	slazenger@interlake.net	49-7542-1204
Ireland-Belfast	Daryl Fullerton	daryl.fullerton@cfug.ie	0044 1232 223224
Ireland-Dublin	Colm Brazel	cbweb@iol.ie	+353 (01) 4941625
Netherlands	Simon Slooten	slooten@prisma-it.nl	+31 (0)10 458 7148
Singapore	Chou Hao Chan	chiouhao@intracomm.com	
United Kingdom-London	Paul Underwood	paul@cfug.co.uk	44 (0) 171 589 4

LISTING

ADVERTISERS INDEX

ADVERTISER	URL	PH	PG
ABLECOMMERCE	WWW.ABLECOMMERCE.COM	360.253.4142	2,4
ADHOST	WWW.ADHOST.COM	888 ADHOST-1	37
ALLAIRE	WWW.ALLAIRE.COM	888.939.2545	11,27,45
CAREER OPPORTUNITIES			53
CATOUZER	WWW.CATOUZER.COM	604.662.7551	59
COMPUTERJOBS.COM	WWW.COMPUTERJOBS.COM		3
COMPUTERWORK.COM	WWW.COMPUTERWORK.COM		25
CORDA TECHNOLOGIES	WWW.CORDA.COM	888.763.0517	31
DEVELOPERSNETWORK	WWW.DEVELOPERSNETWORK.COM	416.203.3690	15
DIGITALNATION	WWW.DEDICATEDSERVER.COM	703.642.2800	9
EKTRON	WWW.EKTRON.COM	603.594.0249	29
INFOBOARD	WWW.INFOBOARD.COM	800.514.2297	41
INTELIANT	WWW.INTELIANT.COM	800.815.5541	47
INTERMEDIA.NET	WWW.INTERMEDIA.NET	650.424.9935	60
JAVACON 2000	WWW.JAVACON2000.COM		42-43
JAVAONE	WWW.JAVA.SUN.COM/JAVAONE/		33
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	54
ON-LINE DATA SOLUTIONS	WWW.COOLFUSION.COM	516.737.4668	30
PAPERTHIN INC.	WWW.PAPERTHIN.COM	800.940-3087	39
RSW SOFTWARE	WWW.RSWSOFTWARE.COM	508.435-8000	13
SAISOFT	WWW.SAISOFTONLINE.COM	860.793.6681	30
SHIFT4 CORPORATION	WWW.SHIFT4.COM	800.265.5795	21
SITEHOSTING.NET	WWW.SITEHOSTING.NET	888.463.6168	41
SYS-CON MEDIA	WWW.SYS-CON.COM	800.513.7111	51
VIRTUALSCAPE	WWW.VIRTUALSCAPE.COM	212.460.8406	16
WATCHFIRE	WWW.WATCHFIRE.COM	613.599.3888	17
WINMILL SOFTWARE	WWW.WINMILL.COM	888.711.MILL	49
XML DEV CON 2000	WWW.XMLDEVCON2000.COM		22-23

Able Solutions

Enter the realm of browsable store building and administration – from your browser. Build "your_site.com" with secure Merchant Credit Card Processing. Maintain inventory, add discounts and specials to keep your customers coming back. Increase sales with cross selling and membership pricing.

11700 NE 95th Street, Suite 100, Vancouver, WA

www.ablecommerce.com • 360 253-4142

Adhost Internet Advertising

Adhost provides complete web hosting solutions for over twelve hundred business clients. Small firms to multi-nationals, startups to long established companies - every business with which we do business receives the unparalleled level of service and range of products that has set Adhost Internet apart from the pack since 1995.

400 108th Avenue NE, Suite 700, Bellevue, WA 98004

www.adhost.com • (888) ADHOST-1

Catouzer

Catouzer develops web-based intranet and Customer Relationship Management software solutions. With Synergy 2.0, Catouzer continues its lead in providing secure web-based work environments. ColdFusion developers now have the most advanced framework to develop secure web-based projects.

www.catouzer.com • 604 662-7551

Computerjobs.com

ComputerJobs.com is the leading Internet-based job search company in its current markets. ComputerJobs.com is solely dedicated to helping computer and information technology ("IT") professionals find great jobs with companies in need of IT employees. We provide thousands of job listings for candidates seeking jobs in the IT field. Our user-friendly, proprietary Web site provides IT job seekers and hiring companies a convenient, effective way to connect.

3200 Windy Hill Road, Suite 700 West, Atlanta, GA 30339

www.computerjobs.com

ComputerWork.com

ComputerWork.com is a premiere technical job site for computer professionals seeking employment in the IT/IS industry. ComputerWork.com will match your technical skills and career ambitions to our many employers looking to fill their jobs with specialists in computer related fields. You can submit your resume to a specific position on our job board or you can choose to submit your resume to our resume bank, which is accessed by nearly 400 companies nationwide. ComputerWork.com is the FASTEST way to your ideal career!

6620 Southpoint Drive South, Suite 600Jacksonville, FL 32216

www.computerwork.com • 904-296-1993

Corda Technologies

Corda Technologies offers tools to build your own charts and graphs for internal reports, reports on your intranet and Internet sites and for many other applications where fast, high-quality graphs and charts are desirable. Corda also offers an Application Service Provider through PopChart.com which works with high-volume sports web sites to display sports statistics with exciting, interactive charts and graphs. PopChart"! . . .an EXPLOSION of Possibilities!

1425 S. 550 East Orem, UT 84097

www.corda.com • 801-802-0800

DevelopersNetwork.com

Developers Network is the essential online business-to-business resource for new media technology and Internet business solutions. Our Resource, Business and Product channels combine elements of helpware and community in a business setting, successfully reaching those buyers developing and managing Internet strategies.

3007 Kingston Road Toronto, Ontario CANADA M1M 1P1

www.developersnetwork.com • 416-203-3610

digitalNATION - a VERIO company

digitalNATION, VERIO's Advanced Hosting Division, is the world's leading provider of dedicated server hosting, with over 1,650 servers online today. dN's superior connected network and service abilities have earned dN a solid reputation as a first-choice provider of dedicated server solutions (Sun, Windows NT, Linux and Cobalt). digitalNATION has been providing online and network services for over six years. One of the first ISPs to provide dedicated servers running Microsoft Windows NT, the dN staff has unparalleled experience in this industry.

5515 Cherokee Ave, Alexandria, VA 22312-2309

www.dedicatedserver.com • 703 642-2800

Ektron

Ektron supports the next-generation needs of e-businesses by providing dynamic Web infrastructure solution tools designed for use by non-technical staff. Ektron's flagship offering, eContentManager, gives staff members across an organization the hands-on ability to make real-time additions and updates to Web content without requiring knowledge of a programming language -- while still allowing for centralized administrative control and security. With competitive advantages such as ease-of-integration and drag & drop everything, Ektron is looking to provide these empowering products to customers, resellers and integrators.

5 Northern Blvd., Suite 6, Amherst, NH 03031
www.ektron.com • 603-594-0249

EnterAct

EnterAct is the Business Services Group of 21st Century Telecom - Chicago's only single-source, facilities-based provider of bundled telecommunications. Combine this with EnterAct's excellent customer service and technical support and it's easy to see why EnterAct is the premier Internet Service Provider in Illinois.

407 S. Dearborn, 6th Floor, Chicago, IL 60605
www.enteract.com • 312-955-3000

Inteliant

Inteliant Corporation, a leading ColdFusion consulting firm, has an outstanding reputation for providing highly skilled developers for Internet, Intranet, Extranet, Software Development, or any ColdFusion application. Our national practice has emerged to meet the evolving needs of our

clients by providing resources onsite or developing remotely. Our company provides the most cost effective service in the industry and we strive to add value to your projects by minimizing expenses whenever possible. Inteliant…"Delivering Intelligent Solutions

1150 Hancock Street, Suite 4, Quincy, MA 02169
www.inteliant.com • 800-815-5541

Intermedia, Inc.

Our advanced virtual hosting packages (powered by Microsoft Windows NT and Internet Information Server 4.0) offer an environment supporting everything today's advanced Web developer or sophisticated client could ask for. Complete ODBC support is available on plans B and C. We support Microsoft Index Server on all hosting plans.

953 Industrial Avenue, Suite 121, Palo Alto, CA 94303
www.intermedia.net • 650 424-9935

On-Line Data Solutions

CoolFusion.com is dedicated to providing unique and powerful add-on solutions for ColdFusion development and implementation. The site is hosted and maintained by On-Line Data Solutions, Inc. – a leader in ColdFusion integration. Our ColdFusion integration products line is called inFusion – a combination of "infuse" and ColdFusion. For information about our flagship product, inFusion Mail Server, we invite you to read the online information (where you can also download the latest beta) and join the inFusion Mail Server support list.

24 Elm Street, Centereach, NY 11720-1706
www.coolfusion.com • 516 737-4668

PaperThin, Inc.

PaperThin offers a 100% ColdFusion-based packaged solution empowering users with out-of-the box Web publishing and dynamic content management functionality for internets, intranets and extranets. Rich, browser and role-based tools allow users to easily create, update, schedule and personalize content without HTML knowledge, while eliminating the need for client software, technical training or complex scripting. Open and extensible, developers can quite readily integrate custom ColdFusion code and applications.

267 King Caesar Road, Duxbury, Massachusetts 02332
www.paperthin.com • 800-940-3087

SaiSoft

As a recognized Allaire, Microsoft and IBM Solutions Provider, SaiSoft's Strategic focus is to become the most definitive Internet Architect, by building long lasting e-business development partnerships. With development operations in India & the UK, SaiSoft also undertakes off-shore consultancy projects where a 'four-step implementation' model is adopted to meet client needs satisfactorily.

446 East Street, Plainville, CT 06062
www.saisoftonline.com • 860-793-6681

Shift4 Corporation

Shift4 Corporation is a leading provider of credit card and transaction processing software utilized by merchants and application developers in various industries. Shift4 products facilitate the point-of-sale and accounting functions associated with electronic payment media, including credit cards, debit cards, purchase cards, specialty cards, private label cards and electronic check clearing. More than 2,000 customers worldwide utilize Shift4 software to process over 85 million credit card transactions annually.

8691 W. Sahara Ave.Las Vegas, NV 89117-5830
www.shift4.com • 800 265-5795

Sitehosting.NET

Successful electronic commerce starts at SiteHosting.net; a division of Dynatek Infoworld, Inc., which provides total Web development services. We offer personal and efficient customer service with reliability at value prices. All our plans include access to SSL (Secure Socket Layer). We support ColdFusion, Active Server Pages, Real Audio/Video, Netshow Server, and more. Our hosting price starts at $14.95/month.

13200 Crossroads Parkway North, Suite 360, City of Industry, CA 91746
www.sitehosting.net • 877 684-6784

Virtualscape

Why host with Virtualscape? Nobody else on the Internet understands what it takes to host ColdFusion like we do. Virtualscape is the leader in advanced Web site hosting. From Fortune 500 extranets to e-commerce sites and more, developers recognize our speed, stability, reliability and technical support.

215 Park Avenue South, Suite 1905, New York, NY 10003
www.virtualscape.com • 212 460-8406

Watchfire

Watchfire is the leading provider of software solutions that give organizations the power to ensure the quality and usability of their websites. Watchfire was recently voted one of the fastest growing Independent Software Vendors in North America (Information Week) and one of Canada's 25 "Up and Comers" (Financial Post). The Company currently sells into more than 60 countries and has over 50,000+ installed customers including over 50% of Fortune 500 companies and many of the largest e-commerce, government and education sites on the Web

135 Michael Cowpland Dr, Suite 400, Kanata, Ontario K2M 2E9 Canada
www.watchfire.com • 613-599-3888

WinMill Software

WinMill Software is the premier resource for Internet and Intranet development, expert consulting, certified classroom training, Web-based education, e-commerce, development software, and technical support with expertise in site design and development, instructional design and delivery, application development, site hosting, training and site management. WinMill Software was founded on the principle of supporting the entire project lifecycle and our hallmark is a knowledge transfer process that maximizes the skills and intellect of your Internet development associates. We will work with your team to build a stable system architecture.

420 Lexington Avenue, Suite 307, New York, NY 10170
www.winmill.com • 888-711-MILL

CFDJnews

Eye Media Releases Version 2.5 of Virtual Auctioneer

(*Dallas, TX*) – eye media, inc., released Version 2.5 of its Virtual Auctioneer software at "Internet World 2000" in early April in Los Angeles.

Virtual Auctioneer offers middle-market companies a powerful and cost-effective packaged solution to support a variety of e-commerce transactions, including Internet auctions, exchanges and online stores.

www.eyemedia.net

Continuus Releases KnowledgeSynergy

(*Irvine, CA*) – Continuus Software Corporation has released Continuus KnowledgeSynergy, a knowledge management solution that actively provides development organizations with the ability to dramatically improve team productivity by reducing learning curves; reusing components, processes and other intellectual capital; and rapidly evolving best practices within their active development projects.

www.continuus.com

Ektron's eWebEditPro Offers New Features

(*Amherst, NH*) – Ektron, Inc., has introduced the newest version of its eWebEditPro with important new functionality including well-formed HTML and Office 2000 compatibility. eWebEditPro now provides HTML that conforms to the rules of XML and a custom tag for the content management portion of Allaire's Spectra.

www.ektron.com

Allaire Teams with Cisco

(*Cambridge, MA*) – Allaire Corporation has integrated the high-availability clustering features of its Internet business platform with Cisco's LocalDirector load-balancing solutions to deliver increased Web site reliability and performance.

Key benefits include ColdFusion application-based load management, CF application availability awareness, CF Server recovery, automatic e-mail alerts for CF server issues, IP load balancing and single domain name/IP address.www.allaire.com

ServiceConnex.com Transforms Access

(*Washington, DC*) – ServiceConnex.com has developed a new-generation data warehouse to provide enhanced data mining and e-commerce services for automotive businesses and consumers.

Using its proprietary B2B and B2C technology platform, ServiceConnex.com aggregates and standardizes automobile service record information. This information is then integrated into Web portals, auto-related Web sites and used-car marketplaces, increasing the viability and the reliability of online sales.

www.ServiceConnex.com

Letters to the Editor CFDJ

A Disgruntled Reader Writes:

With regard to the article written by Michael MacDonald titled "Create Dynamic Business and Scientific Graphics for the Web" [***CFDJ***, Vol. 1, issue 6], I'm shocked you didn't mention Macromedia Generator 2….What's up with that? You mention the other bases but not MG2. You betta check yourself!

I suggest you do some homework at www.macromedia.com/software/generator/. (Hmmm! but you use Flash4 in your Web site – very interesting ???)

Oh wait, I mistook your advertisement for an article under the heading "CF Tips& Techniques"– I guess that's an editorial issue?

It's fine to promote/showcase your product, but not to disguise it as an article claiming to examine solutions to a problem. Also Java applets are definitely not the way to solve this problem, folks.

Need I remind you this is a developer's magazine? It should have in-depth articles covering possible solutions for developers. It's okay to focus on one product but you must also mention other avenues of exploration so the developer can compare and choose the best product for their needs.

As for editor-in-chief Robert Diamond's comment on page 5, "Michael MacDonald has written a great article on creating dynamic graphics for the Web: I'm sure many of us could use a better method for this process." Well, yes, we still can use a better method after this article, Robert.

I don't mean to be so critical – don't take it personally, I just expect higher standards from "the world's leading ColdFusion resource" (taken from your homepage at www.sys-con.com/coldfusion/).

My two cents! Thank you.

—Steve Killingbeck
skillingbeck@periscope.com

The Author Replies:

Thanks for your candid feedback. The plain fact of the matter is that this article was written in July 1999, Macromedia Generator 2 wasn't even shipped until October, and most likely wouldn't have made review/press time for this article anyway.

So far as product promotion goes, it's my honest belief that the solution I detailed is the easiest and most powerful way to get dynamic charting accomplished. I did reference the other common ways that this problem is addressed (I've been in this field for four years). Even after reviewing the Macromedia product, I would comment that it's weak in terms of available chart types and features that could be rendered.

So far as Java applets on the client are concerned – guess what…a lot of people do it this way, and a lot of products embed Java applets that get downloaded to the client. I think I made the point of why that's not a good idea.

Anyway, I do appreciate the feedback – although I think it's unfair to suggest that it is nothing more than a product promotion. I wrote the article based on my four years in this part of the industry and what I believe to be an ideal solution to the dynamic chart/graph problem.

—Michael R. MacDonald
mikem@visualmining.com

A Code Commenting Story

I was reading the January 2000 issue of ***CFDJ*** (Vol. 2, issue 1) and thought I'd offer a comment about it as per your Editorial on page 5.

I'm familiar with the no-comments method of programming. Years ago when I had my Amiga 500 (1 MB RAM, no hard drive), I coded a text-based RPG that was 12 pages when I printed it out (tightly packed). When I went back to read it later, much of the lengthy calculations made no sense. It didn't help that all variables were two characters long.

Now that I've started coding professionally, I enjoy commenting my code... and adding snide remarks like "This is brilliant." I try to make my comments almost conversational; that way I find them humorous to write and whoever reads them after me will understand what I'm talking about.

Anyway, that's my story for now. If anyone else has something to say I'd be interested in reading it.

—Andrew H.
andrew@earthramp.com

Catouzer

www.catouzer.com