

COLD FUSION Developer's Journal

ColdFusionJournal.com

February 2002 Volume: 4 Issue: 2

web services **EDGE**
world tour 2002

WASHINGTON, DC.....**FEBRUARY 26**

NEW YORK, NY.....**MARCH 19**

SAN FRANCISCO, CA.....**APRIL 22**

Page 48 Register for Web Services
Edge 2002 East Gold Passport,
Attend the World Tour FREE!

Editorial

Work Smarter... Not Harder

Robert Diamond page 5

Product Review

ColdFusion 5 UltraDev Studio

Tom Muck page 30

Book Review

Mastering ColdFusion 5

Mark Cyzyk page 48

What's Online

page 54

Q&A

Bruce Van Horn page 56

CFDJ News

page 58

SYS-CON
MEDIA

Making More of UDFs

Powerful tools
to enhance and
empower your
CF development

by Raymond Camden p. 6

<BF> on <CF>: Faster and Safer Database Queries 12

The benefits of using the <CFQUERYPARAM> tag

Ben Forta

Custom Tags: Convert and Resize Images Using CFXImage 14

Enable faster download of your Web pages

Greg Bell

Custom Tips: A Banner Ad Custom Tag 20

Quick and painless coding to create a banner ad

Eben Hewitt

CFDJ Feature: Smart Tags 24

Implement a simple version of Smart Tags technology using ColdFusion

Andrew Cripps

UDFs: Functional Literacy 28

Create and employ user-defined functions: a step-by-step guide

Kevin Schmidt

CFDJ Feature: Unlocking Restricted Use of CFFILE, CFCONTENT, and More 34

Using restricted tags

Charles Arehart

Foundations: A Fusebox 3 Tutorial – Part 2 42

Build a Fusebox application – a ‘donations’ apparatus

Hal Helms

CF&MS-SQL: Using MS-SQL Stored Procedures with ColdFusion Part 3 50

Improve site performance

Ian Rutherford

Quill Design

www.quilldesign.com

Rackspace
www.rackspace.com

Empirix
www.empirix.com/double/cfm

international advisory board

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

department editors

editor-in-chief

Robert Diamond robert@sys-con.com

vice president, production

Jim Morgan jim@sys-con.com

executive editor

M'lou Pinkham mpinkham@sys-con.com

managing editor

Cheryl Van Sise cheryl@sys-con.com

editor

Nancy Valentine nancy@sys-con.com

associate editor

Jamie Matusow jamie@sys-con.com

associate editor

Gail Schultz gail@sys-con.com

associate editor

Jean Cassidy jean@sys-con.com

online editor

Lin Goetz lin@sys-con.com

product review editor

Tom Taulli

tips & techniques editor

Matt Newberry

writers in this issue

Charles Arehart, Greg Bell, Raymond Camden,
Andrew Cripps, Mark Czyzyk, Robert Diamond,
Ben Forta, Hal Helms, Eben Hewitt, Thomas Muck,
Ian Rutherford, Kevin Schmidt, Bruce Van Horn

subscriptions:

For subscription requests please call
1 800 513-7111 or go to: www.sys-con.com

cover price \$8.99/issue

domestic \$89.99/yr. (12 issues)

canada/mexico \$99.99/yr

overseas \$129.99/yr

back issues \$12 U.S. \$15 all other

editorial offices: SYS-CON MEDIA, INC.

135 Chestnut Ridge Rd., Montvale, NJ 07645

Telephone: 201 802-3000 Fax: 201 782-9600

COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)

is published monthly (12 times a year)

for \$89.99 by SYS-CON Publications, Inc.,

135 Chestnut Ridge Rd., Montvale, NJ 07645

postmaster: send address changes to:

COLDFUSION DEVELOPER'S JOURNAL

SYS-CON MEDIA

135 Chestnut Ridge Rd., Montvale, NJ 07645

copyright © 2002 by SYS-CON MEDIA

All rights reserved. No part of this publication may
be reproduced or transmitted in any form or by any
means, electronic or mechanical, including photocopy
or any information storage and retrieval system,
without written permission.

For promotional reprints, contact reprint coordinator:

Carrie Gebert carrieg@sys-con.com

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

distribution in USA:

by International Periodical Distributors

674 Via De La Valle, Suite 204,

Solana Beach, CA 92075

Phone: 619 481-5928

All brand and product names used on these pages are trade names,
service marks, or trademarks of their respective companies.



Work Smarter...Not Harder

BY ROBERT DIAMOND



In today's tough economic climate, the working world is becoming a more difficult place for everyone, including ColdFusion developers. Budgets are getting leaner, workers are being laid off, and work pressures are increasing as well. A recent discussion with fellow developers revealed that most of them were working longer hours than they used to – to deal with the decrease in staff. Several expressed the belief that they're now doing the jobs of several people, which is probably true, as the majority of companies are "tightening their belts" to ride through some rough economic waters. Piggybacked onto this news is an even worse bit of information – doing the jobs of several people probably won't earn you more money.

The articles in this month's issue might not earn you more money either, but they will help you work smarter and should earn you more time. As a ColdFusion developer you're already on the right track, as CFML has proven itself time and time again to be one of the best ways to hit the ground running on Web development. Now that you're developing in CF, the next step is to be a smarter CF developer. That's what this issue of **CFDJ** is about – working smarter to save time.

There are a few easy ways to do this. One way – custom tags – has been around for a while, and a new way – user-defined functions – was introduced in ColdFusion 5. UDFs, for those of you who haven't yet moved to CF 5.0, is best described as a relative of custom tags, not quite a brother or sister though, more like a second cousin. The most common way to save time as a developer is to reuse code. I do this all the time and I'm sure you do too. Parts of many of the projects I've worked on are similar to many others, and if anyone wanted to do a fascinating case study on the life of a developer, those digital footprints would certainly be the place to start.

An alternate way is to use other developers' code, which you can easily do through the many online directories and developer resources. Custom tags and user-defined functions are, quite conveniently for my ramblings, what this issue is about.

User-defined functions are defined within blocks of CFSCRIPT code. They're then referenced the same as all the other `<cftags>` we know and love. The code that makes UDFs do their thing is CFSCRIPT, which looks quite similar to JavaScript. Once you write a UDF you can then use the function anywhere you use a ColdFusion expression, such as in tag attributes and between the # signs in output.

How do UDFs differ from custom tags? For starters, they don't include access to any native CFML tag functionality. When writing a custom tag, you can use CFQUERY, CFHTTP, CFMAIL, and other CFML tags. On the flip side, user-defined functions provide the convenience of a simple scripting language and a simple return to send values back to the Web application. So, read on and decide for yourself.

Use them both and you'll be saving time, time you can spend enjoying life, or most likely, doing more work.



ABOUT THE AUTHOR

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

Robert Diamond

@ ROBERT@SYS-CON.COM



ColdFusion Feature

By RAYMOND CAMDEN

Making

*U*ser-defined functions, without a doubt, were one of the most asked-for and most appreciated features of ColdFusion 5. Before ColdFusion 5 was even released publicly, lucky developers who had access to the beta versions were playing with UDFs. Rob Brooks-Bilson and I released a site, the ColdFusion Library Project (www.cflib.org), dedicated to creating and sharing libraries of cool UDFs.



Powerful tools to enhance and
empower your CF development

More of UDFs

For those of you who don't know, UDFs are functions, built-in ColdFusion functions like `DateFormat`, that developers can write and use with their applications. Many people see UDFs as a type of custom tag, and in some ways that's correct. Like custom tags, UDFs can be a great way to save development time. For example, you need to write a "Fahrenheit to Celsius" UDF only one time. Once one developer has written this UDF, other developers can include it in their code. Like custom tags, UDFs can "black box" (see sidebar) the data they

use, ensuring they don't overwrite variables in the calling template.

However, the more I work with ColdFusion 5, the more I realize UDFs can be useful in other roles. The purpose of this article is to help developers recognize areas where UDFs can be applied. I'll show how UDFs can be useful for simplifying data entry or working with repeated logic tasks on a page. In general, I'll show how UDFs can be a great development tool in ways you may not have thought of before.

Data Entry Abstraction

Many times we work with scripts that deal with a lot of data. I'm not talking about database data, but configuration style data. For example, imagine a template that displays a form with its particulars configured in the top portion of the script, making it a bit easier to modify the script. For instance, I may want to add a new question to the form. Instead of worrying about the form's layout, I can simply edit the configuration information at the top of the script. Listing 1 shows a simple example of this type of template.

Our template begins with a CFSCRIPT block. We define each of the particular form fields we want to use. We're working with an array of structures, so, as you can see on lines 8, 14, and 20, each field begins with a StructNew() call. Here's an example:

```
Fields[1] = StructNew();
Fields[1].Type="text";
Fields[1].Name="Name";
Fields[1].Display = "Your Name";
Fields[1].Required = True;
```

The details of each structure are pretty self-evident. We set a name and a display name for each form element, along with a type field that determines which type of form element is required. We also have a required field that determines if a particular form field has to be filled out.



Beneath this group of statements is the actual logic of the form handling and display.

As you can imagine, this setup is easy to use. To add a new field, simply add a new structure. However, what happens if you need to remove a field or add a field in the middle? Since we have an array, we would have to renumber all the array indexes. Even adding an element at the end takes quite a few lines of code. We can make this quicker and easier with a simple UDF.

Listing 2 shows how the addition of a new UDF makes the code simpler to work with and easier to modify. We've replaced many of the lines with a call to our new UDF, formField(). Let's take a closer look at it:

```
function formField(name, display) {
    Fields[ArrayLen(Fields)+1] =
    StructNew();
    Fields[ArrayLen(Fields)].Name =
    name;
    Fields[ArrayLen(Fields)].Display =
    display;
    //Required is optional. Default to
    false
    if(ArrayLen(Arguments) GTE 3)
    Fields[ArrayLen(Fields)].Required =
    Arguments[3];
    else
    Fields[ArrayLen(Fields)].Required =
    False;
    //Type is optional. Default to false
    if(ArrayLen(Arguments) GTE 4)
    Fields[ArrayLen(Fields)].Type =
    Arguments[4];
    else Fields[ArrayLen(Fields)].Type =
    "text";
    return true;
}
```

This UDF acts like the abstracted access to the Fields array we used before. This time, however, it handles all the grunt work. It creates the structure, handles the act of adding to the end of the array, and also handles our defaults automatically.

You may notice that the formField UDF accesses the Fields array; we didn't pass it in. Normally, this would be a cardinal sin for a UDF, but the fact that UDFs can access local variables serves our purpose. We don't have to worry about sending in the Fields array or returning the array. (However, all UDFs should return something, so we simply return true.) The point is, there's nothing wrong with UDFs working with variables in the local page scope, as long as you know the data is being manipulated.

"Black Box" Data

While UDFs can "black box" data just like custom tags, don't forget that they can access local variables much easier than custom tags. This means they're a bit more dangerous if you're not careful.

Every variable used in a UDF that's not passed in must use a var statement to declare it or you run the risk of overwriting data in the local page scope. Don't forget to declare variables used in loops as well. UDFs ability to work with local variables can be nice, as long as you know what you're doing.

Compare how we set up and define form fields in Listing 2 to how they're done in Listing 1. As you can see, it's quite a bit easier. We also no longer need to worry about the order of form fields in our configuration data. We can move formField() calls up without any worries.

Logic Abstraction

Let's look at another way that UDFs can benefit our programming. A friend had asked me to develop a navigation system that would be "intelligent." The goal was to take a navigation menu and dynamically remove the link when the navigation item was the same as the current page. For each navigation item we check to see if it pointed to the page we were currently displaying. If it did, we simply output the name of the page bolded with tags. If it didn't, we output a link around the name of the page. Listing 3 displays a snippet of this navigation menu system.

This menu can be included in other files or in the onRequestEnd.cfm template to add navigation to all pages on a Web site. (For those of you who are unfamiliar with onRequestEnd.cfm, this file acts like Application.cfm. For every request ColdFusion handles, it will attempt to find and load onRequestEnd.cfm if it finds it in the current directory. The file is run at the end of the request, unlike Application.cfm, which is run at the beginning of each request.) We begin by assigning the current file name to a variable "CurrentFile." Then for each and every navigation item we use the following logic:

```
<CFIF CurrentFile IS "about.cfm">
    <B>About</B>
<CFELSE>
    <A HREF="about.cfm">About</A>
</CFIF>
```


In Listing 3, we have only four navigation items, but each item uses five lines of code. Adding new items is not a quick and simple process. Also, imagine if you decide not to bold navigation items but to italicize. You'd then need to change multiple lines of code. As you'll see, though, we can use a simple UDF to shorten the navigation template and make adding new items easier.

Listing 4 begins with a UDF called navDisplay:

```
function navDisplay(title,href) {
    if(GetFileFromPath(GetCurrentTemplatePath()) is href) return
    "<b>#title#</b>";
    return "<a href='#href#'>#title#</a>";
}
```

This UDF takes two arguments, title and href. This UDF handles the logic of grabbing the current file name (using the GetCurrentTemplatePath and GetFileFromPath functions), and if it's not equal to the href value passed, returns the title with a link created around it. Looking at the rest of Listing 4, the five lines of code we used before have been replaced with one line:

```
#navDisplay("About","about.cfm")#<BR>
```


Other places where this type of UDF could be useful would be security checks. Imagine a similar navigation system that suppresses links to pages that users don't have access to. For example:

```
<CFIF Request.LoggedIn AND
ListFind(Request.Groups,"Editors")>
<A HREF="editor.cfm">Editor
Commands</A><BR>
</CFIF>
```

This code assumes session information has been copied into the request scope (so we don't have to use CFLOCK). We check to see if the user is logged on and has the value "Editors" in the list of groups he or she belongs in. This could be rewritten in a much shorter, easier to understand line of code:

```
#secureLink("Editor Commands","editor.cfm","Editors")#
```

Conclusion

What have we learned in this article? UDFs are not just "cool" custom tags. By that I mean don't make the mistake of thinking that it's only worthwhile to write a UDF if you think you'll use it again in multiple documents. Used properly, they can be very powerful tools that enhance and empower your ColdFusion development. While you won't see UDFs like these on cflib.org (the site is dedicated to more general purpose UDFs), don't forget that these types of UDFs can be very useful. Not only did we cut down on the size of our templates, we also made them easier to work with and less cluttered. 

About the Author

Raymond Camden is the principal Spectra compliance engineer for Macromedia. He is coauthor of the Allaire Spectra e-Business Construction Kit and Mastering ColdFusion 5. He helps manage the Hampton Roads ColdFusion User Group (www.hrcfug.org) and is the creator and comaintainer of the Common Function Library Project (www.cflib.org).

 JEDIMASTER@MACROMEDIA.COM

SUBSCRIBE AND SAVE

WebServices JOURNAL

Offer subject to change without notice

ANNUAL NEWSSTAND RATE
\$83.88
YOU PAY
\$69.99
YOU SAVE
\$13.89 Off the Newsstand Rate

DON'T MISS AN ISSUE!

Receive 12 issues of **Web Services Journal** for only **\$69.99!** That's a savings of **\$13.89** off the annual newsstand rate.

Sign up online at www.sys-con.com or call 1 800 513-7111 and subscribe today!

In February **WSJ**:

Getting Greater Business Value from Web Services
Thinking of a wider audience

MS Gets It At Last!
Pleasant surprises from the Beta2 of .NET

The Largest Web Application in the World
Is this what .NET My Services will become?

Security and the .NET Framework
Grave danger awaits if your security isn't robust

Peopleclick and HR.NET
How can Web services and .NET work for developers today?



Listing 1

```

<!---
Simple Dynamic Form
--->

<CFSCRIPT>
Fields = ArrayNew(1);

Fields[1] = StructNew();
Fields[1].Type="text";
Fields[1].Name="Name";
Fields[1].Display = "Your Name";
Fields[1].Required = True;

Fields[2] = StructNew();
Fields[2].Type="text";
Fields[2].Name="EmailAddress";
Fields[2].Display = "Your Email Address";
Fields[2].Required = True;

Fields[3] = StructNew();
Fields[3].Type="text";
Fields[3].Name="Color";
Fields[3].Display = "Your Favorite Color";

</CFSCRIPT>

<CFIF IsDefined("Form.Submit")>
<CFSET GoodForm = True>
<CFLOOP INDEX="X" FROM=1 TO="#ArrayLen(Fields)#">
  <CFIF StructKeyExists(Fields[X], "Required")
    AND Fields[X].Required AND
    NOT StructKeyExists(Form, Fields[X].Name)
    OR NOT Len(Trim(Form[Fields[X].Name]))>
    <CFSET GoodForm = False>
    <CFOUTPUT>
    You must fill out field:
    #Fields[X].Display#<BR>
  </CFOUTPUT>
</CFIF>
</CFLOOP>
<CFIF GoodForm>
  <!--- Do something with the results --->
</CFIF>
</CFIF>

<FORM ACTION="listing1.cfm" METHOD="Post">
<TABLE>
<CFLOOP INDEX="X" FROM=1 TO="#ArrayLen(Fields)#">
  <CFPARAM NAME="Form[Fields[X].Name]"
    DEFAULT="">
  <CFOUTPUT>
  <TR>
    <TD>#Fields[X].Display#</TD>
    <TD>
      <CFIF Fields[X].Type IS "text">
        <INPUT TYPE="text"
          NAME="#Fields[X].Name#"
          VALUE="#Form[Fields[X].Name]#">
      </CFIF>
    </TD>
  </TR>
</CFOUTPUT>
</CFLOOP>
<TR>
  <TD>&nbsp;</TD>
  <TD>
    <INPUT TYPE="Submit" NAME="Submit"
      VALUE="Send Data">
  </TD>
</TR>
</TABLE>
</FORM>

```

Listing 2

```

<!---
Simple Dynamic Form
--->

<CFSCRIPT>
Fields = ArrayNew(1);
function formField(name, display) {
  Fields[ArrayLen(Fields)+1] = StructNew();
  Fields[ArrayLen(Fields)].Name = name;
  Fields[ArrayLen(Fields)].Display = display;
  //Required is optional. Default to false
  if(ArrayLen(Arguments) GTE 3)
    Fields[ArrayLen(Fields)].Required =
      Arguments[3];
  else Fields[ArrayLen(Fields)].Required =
    False;
  //Type is optional. Default to false
  if(ArrayLen(Arguments) GTE 4)
    Fields[ArrayLen(Fields)].Type =
      Arguments[4];
  else Fields[ArrayLen(Fields)].Type = "text";
  return true;
}

formField("Name", "Your Name", True);
formField("EmailAddress", "Your Email Address");
formField("Color", "Your Favorite Color");

</CFSCRIPT>

<CFIF IsDefined("Form.Submit")>
<CFSET GoodForm = True>
<CFLOOP INDEX="X" FROM=1 TO="#ArrayLen(Fields)#">
  <CFIF StructKeyExists(Fields[X], "Required")
    AND Fields[X].Required AND
    NOT StructKeyExists(Form, Fields[X].Name)
    OR NOT Len(Trim(Form[Fields[X].Name]))>
    <CFSET GoodForm = False>
    <CFOUTPUT>
    You must fill out field:
    #Fields[X].Display#<BR>
  </CFOUTPUT>
</CFIF>
</CFLOOP>
<CFIF GoodForm>
  <!--- Do something with the results --->
</CFIF>
</CFIF>

<FORM ACTION="listing1.cfm" METHOD="Post">
<TABLE>
<CFLOOP INDEX="X" FROM=1 TO="#ArrayLen(Fields)#">
  <CFPARAM NAME="Form[Fields[X].Name]"
    DEFAULT="">
  <CFOUTPUT>
  <TR>
    <TD>#Fields[X].Display#</TD>
    <TD>
      <CFIF Fields[X].Type IS "text">
        <INPUT TYPE="text"
          NAME="#Fields[X].Name#"
          VALUE="#Form[Fields[X].Name]#">
      </CFIF>
    </TD>
  </TR>
</CFOUTPUT>
</CFLOOP>
<TR>
  <TD>&nbsp;</TD>
  <TD>
    <INPUT TYPE="Submit" NAME="Submit"
      VALUE="Send Data">
  </TD>
</TR>
</TABLE>
</FORM>

```

```

!---
Nav menu
--->

<CFSET CurrentFile =
    GetFileFromPath(GetCurrentTemplatePath())>

<CFIF CurrentFile IS "about.cfm">
    <B>About</B>
<CFELSE>
    <A HREF="about.cfm">About</A>
</CFIF>

<BR>

<CFIF CurrentFile IS "news.cfm">
    <B>News</B>
<CFELSE>
    <A HREF="news.cfm">News</A>
</CFIF>

<BR>

<CFIF CurrentFile IS "listing3.cfm">
    <B>Listing3</B>
<CFELSE>
    <A HREF="listing3.cfm">Listing3</A>
</CFIF>

<BR>

<CFIF CurrentFile IS "search.cfm">
    <B>Search</B>
<CFELSE>
    <A HREF="search.cfm">Search</A>
</CFIF>

<BR>

```

```
<!--  
Nav menu  
-->  
  
<CFSCRIPT>  
  
function navDisplay(title,href) {  
    if(GetFileFromPath(GetCurrentTemplatePath()) is href)  
        return "<b>#title#</b>";  
    return "<a href=\"#href#\">#title#</a>";  
}  
  
</CFSCRIPT>  
  
<CFOUTPUT>  
  
#navDisplay("About","about.cfm")#<BR>  
#navDisplay("News","news.cfm")#<BR>  
#navDisplay("Listing4","listing4.cfm")#<BR>  
#navDisplay("Search","search.cfm")#<BR>  
  
</CFOUTPUT>
```

FEBRUARY CFDJ 11

BY
BEN
FORTA

Faster and Safer Database Queries

Using the <CFQUERYPARAM> Tag

Databases and database access are fundamental elements of just about every ColdFusion application ever created. Database access makes applications real and live and dynamic and valuable, but it's also a major source of performance problems and a primary potential security target. In this article I discuss an oft overlooked tag, <CFQUERYPARAM>, designed to help address both potential problems.

Understanding <CFQUERYPARAM>

Database access (queries, insertions, updates, etc.) occurs by issuing SQL statements – plain text strings containing combinations of special statements, keywords, clauses, operators, and more. These strings are passed to the DBMS for processing (usually via a database driver) and are sometimes referred to as being *ad hoc*.

So far so good. But this is where it gets a little complicated. There are an infinite number of possible SQL statements (when you take into account the values passed to them), so once a DBMS receives a SQL statement, the DBMS must first analyze it

to determine if it's valid, and then work out the best way to process it. Often, the process of parsing, validating, and analyzing the query (the term *query* refers to all SQL queries, even if they're not SELECT query operations) takes longer than processing the query itself.

Once the query is validated and analyzed, the DBMS tries to cache it so that the process won't need to be repeated unnecessarily. Simple queries without dynamic elements, like the following one, can be cached safely and easily:

```
SELECT *
FROM Customers
```

However, queries containing other clauses, like the one that follows, are a little harder to cache. After all, with an infinite number of variations it wouldn't be practical for the DBMS to cache them all:

```
SELECT *
FROM Customers
WHERE CustID=100
```

To cache this type of query the DBMS needs to know which part of the statement is dynamic and which is not, and that's where <CFQUERYPARAM> comes into play.

<CFQUERYPARAM> is a tag used within SQL statements (inside your <CFQUERY>). It replaces parameters (or any passed values) with placeholders that indicate the dynamic portions of a statement. For example:

```
<CFQUERY ...>
SELECT *
FROM Customers
WHERE CustID=<CFQUERYPARAM

CFSQLTYPE="CF_SQL_INTEGER"
VALUE="100">
</CFQUERY>
```

The <CFQUERYPARAM> tag replaces the passed value (100), and does two things: it defines the type of data that will be passed here and then passes the value to be used.

The actual SQL generated by this query will differ from DBMS to DBMS, but that's somewhat irrelevant. The important thing is that the DBMS knows that the query containing this block of text won't change (and can therefore be cached):

```
SELECT *
FROM Customers
WHERE CustID=
```

In DBMSes these variable portions of SQL statements, the ones bookmarked with <CFQUERYPARAM>, are known as *bind parameters*.

Specifying SQL Types

The data type passed to CFSQLTYPE tells the DBMS to expect data of a particular type. The supported types are:

- CF_SQL_BIGINT
- CF_SQL_BIT
- CF_SQL_CHAR
- CF_SQL_DATE
- CF_SQL_DECIMAL
- CF_SQL_DOUBLE
- CF_SQL_FLOAT
- CF_SQL_IDSTAMP
- CF_SQL_INTEGER
- CF_SQL_LONGVARCHAR
- CF_SQL_MONEY
- CF_SQL_MONEY4
- CF_SQL_NUMERIC
- CF_SQL_REAL
- CF_SQL_REFCURSOR
- CF_SQL_SMALLINT
- CF_SQL_TIME
- CF_SQL_TIMESTAMP
- CF_SQL_TINYINT
- CF_SQL_VARCHAR

CFSQLTYPE is an optional attribute, but as a rule you should

The first benefit
of using
<CFQUERYPARAM>
is increased
performance"

always specify an explicit type. The default value is CF_SQL_CHAR (a string) and is frequently not what you need.

Increased Performance

As already implied, the first benefit of using <CFQUERYPARAM> is increased performance. If the DBMS doesn't have to parse, analyze, and validate as much text, it'll be able to respond to requests quicker and more efficiently.

It's common knowledge that stored procedures execute faster than ad hoc SQL statements. Part of the reason for this is that DBMSs cache the compiled stored procedures so they won't need to be processed repeatedly and unnecessarily. By using <CFQUERYPARAM> you get some of the benefits of stored procedures without having to write one.

If you're interested in determining what the actual performance benefit is, turn on ColdFusion debugging output and try the queries both ways. As the DBMS load increases, so should the benefit of using bind parameters.

Protecting from Malicious URL Tampering

Note: As a rule I'm staunchly opposed to documenting security issues and how they're used, but as this particular vulnerability has been documented extensively (there are even Knowledge Base articles about it at www.macromedia.com) I'm making an exception this time.

Another benefit to using <CFQUERYPARAM> is security against malicious URL tampering. Consider the following example, an adaptation of the one used previously:

```
SELECT *
FROM Customers
WHERE CustID=#URL.custid#
```

Here a WHERE clause is populated dynamically using a URL parameter. This type of code is common and popular, and is used any time data drilldown of any kind is used. If the URL was:

```
http://domain/path/file.cfm?custid=100
```

the resulting SQL statement would be:

```
SELECT *
FROM Customers
WHERE CustID=100
```

But what if someone tampered with that URL so that it read:

```
http://domain/path/file.cfm?custid=100;DELETE+Customers
```

The resulting SQL would be:

```
SELECT *
FROM Customers
WHERE CustID=100;
DELETE Customers
```

Depending on the DBMS being used, the <CFQUERY> could end up executing two statements – the SELECT and then DELETE Customers (which would promptly delete all data from the Customers table).

Scared? You should be. As I mentioned earlier SQL statements are not just used for queries. They're also used to create and drop tables, create user logins, change DBMS passwords, set security levels, manage scheduled events, even create and drop entire databases. All features supported by your DBMS may be accessible this way.

Before I go further I must point out that this is not a ColdFusion vulnerability; it's not even a bug or a hole. This is truly a feature – many DBMSs do indeed allow queries to contain more than a single operation; this is legal and by design.

Of course, you should always be checking parameters before passing them to your DBMS. Passing URL parameters through unchecked is programmatic suicide. As such, you should already be using code like this:

```
<CFIF IsDefined("URL.CustID")
AND NOT
IsNumeric(URL.CustID)>
... throw an error or something
...
</CFIF>
```

But <CFQUERYPARAM> provides one extra line of defense. If

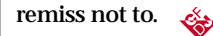
Passing URL parameters through unchecked is programmatic suicide"

the previous tampered URL was passed to the following query, the value would be rejected and an error thrown. The CFSQLTYPE value also doubles as a datatype validation check, and values that don't match the type are rejected. Using <CFQUERYPARAM> you can secure yourself against one of the oldest and best-known hacks in the book.

Note: This type of hack is one of the reasons you should never configure ColdFusion to use administrative accounts for database access (the SA account, for example). As a rule, the login specified in the data source definition should have just the access it needs and nothing more. If administrative access is not available to a login, then some of the more dangerous commands wouldn't be executable.

Summary

<CFQUERYPARAM> is not a new tag; it's been in ColdFusion since version 4.5. However, many developers have yet to use <CFQUERYPARAM>, primarily because they don't understand it. Most good DBMSs (including SQL Server and Oracle) support the use of bind parameters, but some don't. Be sure to read the docs on this tag, and then start plugging it into any new code and even existing apps. With increased performance and improved security you'd be remiss not to.



BEN@FORTA.COM

ABOUT THE AUTHOR

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

Convert and Resize Images Using CFXImage

Enable faster download of your Web pages

BY
GREG
BELL



Many of today's Web pages take too long for users to download. We've all landed on a page, stared impatiently at a blank screen for a minute or two, and become so frustrated that we found another activity to occupy our time.

After responding to the 25 e-mails left minimized on the desktop, we check the page again only to experience a very large image struggling to open before a page times out.

We all know there's a correlation between page load time and page size: an increase in page size means an increase in page load time. Unfortunately, users don't increase the time they spend waiting on your site. Several studies show that most users won't wait longer than 20 seconds for a page to load before leaving the site. Keep the page size less than 50KB in order to keep a user interested in your site.

Even though we know a smaller page size will retain more users, we often come across images larger than 200KB on our own sites. The problem stems from images being posted to the site either directly or through a content-providing template, unknown to the Web administrators. If the latter case is true, you may be able to use a custom tag called CFXImage to control image size and format.

Introducing CFXImage

CFXImage is a DLL that runs only on a Windows 32-bit operating system. The tag uses Microsoft Foundation Classes as a shared library, so "mfc42.dll" and "msvcrt.dll" need to be located in the system path. CFXImage is available for free at www.gafware.com.

Here are some things the tag can do through the actions invoked by the ACTION attribute as taken from the CFXImage documentation:

- **ACQUIRE:** Acquire an image from a TWAIN device
- **ANIMATE:** Add frames to create an animated GIF
- **ARC:** Draw an arc on the image
- **CAPTURE:** Capture a "still frame" from a video capture device
- **CHART:** BAR, LINE, PIE, DOT or CROSS charts or any combination of those
- **COLORBALANCE:** Adjust the color balance of an image
- **CONTRAST:** Adjust the contrast of an image
- **CONVERT:** Convert formats
- **COPY:** Copy an image from one location to another
- **CREATE:** Create a blank image
- **ELLIPSE:** Draw an ellipse on the image
- **EXISTS:** Test for the existence of an image
- **FLIP:** Flip an image vertically or horizontally
- **FLOOD:** Perform a flood fill on the image
- **IPTC:** Insert IPTC-NAA tags into jpegs
- **LINE:** Draw a line on the image
- **OVAL:** Draw an oval on the image
- **PASTE:** Paste image to image; transparent pasting is supported
- **PASTE COMMENT:** Paste an invisible comment in the image
- **PASTE TEXT:** Paste text to an image
- **PIXEL:** Set the color of one pixel in an image
- **READ:** Read image information, width, height, size, time created, etc.
- **RECTANGLE:** Draw a rectangle on the image
- **REDIMENSION:** Proportionately resize the image canvas
- **REMOVE:** Delete an image file from the disk
- **RENAME:** Rename an image file
- **RESIZE:** Resize images to new dimensions or thumbnails
- **ROTATE:** Rotate image by 90, 180, or 270 degrees
- **XML:** Use an XML file to perform batch operations

The tag also supports many file types:

- Jpeg (JPEG)
- Graphics Interchange Format v87a (GIF87A)
- Graphics Interchange Format v89a (GIF89A)
- Windows Bitmap (BMP)
- OS/2 Bitmap (OS2)
- X-Bitmap (XBM)
- Portable Network Graphics (PNG)
- PortableImageMapPlus (PBMPLUS)
- Targa (TARGA)
- Gd (GD)
- Gd2 (GD2)
- Tagged Image File Format (TIFF)
- Zsoft PC Paintbrush (PCX)
- Borland Graphics Interface (BGI)
- Windows Icon (ICON)
- Windows Cursor (CURSOR)
- Mac Paint (MAC)
- Quicktime Picture (PICT)
- Class 3 Fax (FAX)
- Gem (GEM)
- Red Storm Bitmap (RSB)
- Summus Wavelet (WAVELET)
- Black Isle Bitmap (MOS)
- Wireless Bitmap (WBMP)

MACROMEDIA

www.macromedia.com/go/mastering

president and ceo

Fuat Kircaali fuat@sys-con.com

vp, business development

Grisha Davida grisha@sys-con.com

advertising

senior vp, sales & marketing

Carmen Gonzalez carmen@sys-con.com

vp, sales & marketing

Miles Silverman miles@sys-con.com

advertising director

Robyn Forma robyn@sys-con.com

advertising account manager

Megan Ring megan@sys-con.com

associate sales managers

Carrie Gebert carrie@sys-con.com

Alisa Catalano alisa@sys-con.com

Kristen Kuhnle kristin@sys-con.com

editorial

executive editor

M'lou Pinkham mpinkham@sys-con.com

editor

Nancy Valentine nancy@sys-con.com

managing editor

Cheryl Van Sise cheryl@sys-con.com

associate editors

Jamie Matusow jamie@sys-con.com

Gail Schultz gail@sys-con.com

Jean Cassidy jean@sys-con.com

online editor

Lin Goetz lin@sys-con.com

production

vice president, production & design

Jim Morgan jim@sys-con.com

art director

Alex Botero alex@sys-con.com

assistant art directors

Cathryn Burak cathyb@sys-con.com

Louis F. Cuffari louis@sys-con.com

Richard Silverberg richards@sys-con.com

graphic designer

Aarathi Venkataraman aarathi@sys-con.com

sys-con events

vice president, events

Cathy Walters cathyw@sys-con.com

conference manager

Michael Lynch mike@sys-con.com

sales executives, exhibits

Richard Anderson richard@sys-con.com

Michael Pesnick michael@sys-con.com

show assistant

Niki Panagopoulos niki@sys-con.com

customer relations/JDJ store

manager, customer relations /JDJ store

Anthony D. Spitzer tony@sys-con.com

customer service liason

Patti Del Vecchio patti@sys-con.com

web services

web designers

Stephen Kilmurray stephen@sys-con.com

Christopher Croce chris@sys-con.com

accounting

chief financial officer

Bruce Kanner bruce@sys-con.com

assistant controller

Judith Calnan judith@sys-con.com

accounts payable

Joan LaRose joan@sys-con.com

accounts receivable

Jan Braidech jan@sys-con.com

accounting clerk

Betty White betty@sys-con.com

Many users submitted incorrect image sizes and formats, creating an oversized and unattractive Web page"

How I Used CFXImage

I recently completed an application that allows a Motorola Computer Group Embedded Connections Partner to upload a company logo and other company information (see Figure 1). Even though I provided guidelines for submission (including image size and type), I found that many users submitted incorrect image sizes and formats, creating an oversized and unattractive Web page.

After weeks of painstakingly fixing the images, I found a solution using CFXImage. CFXImage provides the functionality to resize and convert image formats as well as basically manipulate almost any image to your liking.

Implementing CFXImage into my code was as painless as installing the custom tag on a ColdFusion Server. I first call the custom tag to read all the file attributes of the uploaded file through the READ action:

```
<CFIF #UploadedFile# NEQ ''>
<CFX_Image FILE="#UploadedFile#"
ACTION="READ">
```

The READ action will get many image attributes such as height, width, type, and size.

Next, in Listing 1 I check if the image is already in GIF format. I also check if the image width is greater than 300 pixels. If both cases are true, the image is resized to a width of 300 pixels through the RESIZE action. If an image is the correct size and format, it's copied to the directory through the COPY action.

To produce a sharper 8-bit image, set the BITCOUNT to 24. This allows the image to be proportionately resized back to 8 bits before it's saved back to a GIF format. Using this method, an 8-bit image will have increased colors resulting in a sharper output around the edges. However, this method doesn't work well with transparent images.

Last, in Listing 2 a size check is performed on an image in the wrong format. The image is then converted into GIF format through the CONVERT action and resized through the RESIZE action if it's too large.

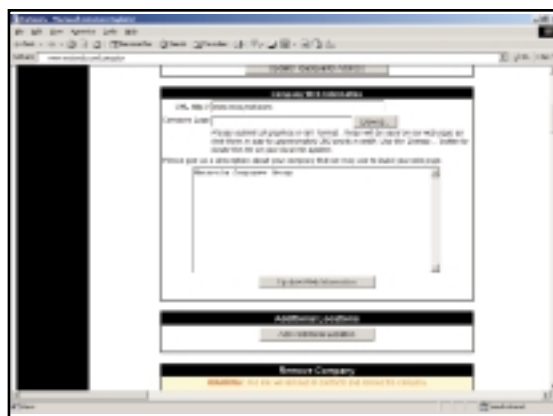


FIGURE 1: MCG Embedded Connections Partner application

Notice how easy the syntax is to convert an image into another format. Basically, it's a matter of making the file extension in the OUTPUT path the desired format and specifying an image type. You can use several different formats, such as GIF, JPEG, BMP, TIFF, and PNG.

That's it. The images on Motorola Computer Group's Partner Web pages are now automatically controlled upon upload. As a result, the pages download faster and look cleaner, which can mean increased traffic throughout the site. Customers can spend more time viewing images on your site rather than checking their e-mail while your graphics load. The difficult task of resizing and converting files is just the tip of the iceberg for CFXImage. I encourage you to explore the options in this custom tag and see what it has to offer your site.

@GREG.BELL@MOTOROLA.COM

ABOUT THE AUTHOR

Greg Bell is a Web developer for the Motorola Computer Group. He has a BS in accounting and computer information systems and is currently working on an MBA from the University of Phoenix.

Listing 1

```
<CFIF #IMG_TYPE# IS "GIF89A" OR #IMG_TYPE# IS "GIF87A">
<CFIF #IMG_WIDTH# GT 300>
    <CFX_Image ACTION="RESIZE"
        FILE="#UploadedFile#"
        OUTPUT="#LibraryDirectory#\Logo-
#FORM.OnLineAccount#.gif"
        BITCOUNT="24"
        QUALITY="100"
        X="300">
<CFELSE>
    <CFX_Image ACTION="COPY"
        FILE="#FORM.UploadedFile#"
        OUTPUT="#LibraryDirectory#\Logo-
#FORM.OnLineAccount#.gif">
</CFIF>
```

Listing 2

```
<CFELSE>

    <CFIF #IMG_WIDTH# GT 300>

        <CFX_IMAGE ACTION="CONVERT"

            FILE="#FORM.UploadedFile#"
```

```

OUTPUT="#LibraryDirectory#\Logo-
#FORM.OnLineAccount#.gif"

    QUALITY="100">

<CFX_IMAGE ACTION="RESIZE"

    FILE="#LibraryDirectory#\Logo-
#FORM.OnLineAccount#.gif"

    OUTPUT="#LibraryDirectory#\Logo-
#FORM.OnLineAccount#.gif"

    BITCOUNT="24"

    QUALITY="100"

    X="300">

<CFELSE>

    <CFX_IMAGE ACTION="CONVERT"

        FILE="#FORM.UploadedFile#"

        OUTPUT="#LibraryDirectory#\Logo-
#FORM.OnLineAccount#.gif"


        TYPE="GIF89A">

        QUALITY="100">

</CFIF>

</CFIF>

```

CODE
LISTING


CODE
LISTING

HOSTMYSITE.COM
www.hostmysite.com

STRATEGIES & SOLUTIONS FOR UNWIRING THE ENTERPRISE

International Wireless Business & Technology Conference & Expo

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

Who Should Attend

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

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

Plan to Exhibit

Provide the Resources To Implement Wireless Strategy

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

C O N F E R E N C E T R A C K S

TRACK 1: DEVELOPMENT

- WAP
- i-Mode
- Bluetooth / 802.11
- Short Messaging
- Interactive Gaming
- GPS / Location-Based
- Wireless Java
- XML & Wireless technologies

TRACK 2: CONNECTIVITY

- Smart Cards
- Wireless LANs incl. Bluetooth
- UMTS/3G Networks
- Satellite Broadband

TRACK 3: WIRELESS APPS

- Education
- Healthcare
- Entertainment
- Transport
- Financial Services
- Supply Chain Management

TRACK 4: HARDWARE

- Cell Phones/WorldPhones
- PDAs
- Headphones / Keyboards / Peripherals
- Transmitters / Base stations
- Tablets

TRACK 5: MANAGEMENT

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

**FOR EXHIBIT & SPONSORSHIP
INFORMATION PLEASE CALL
201 802-3004**



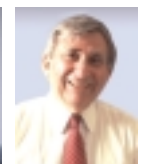
Arie Mazur
President, Chief Executive Officer, and Chairman, Slangsoft



Chris Bennett
Architect, Freedom Technologies



Daniel Elliott
Senior Vice President, Mobile Business, CompuCom



Douglas Lamont
Professor of Marketing, DePaul University



Kenneth Leung
Business Development Manager, Retail and Emerging Technologies, IBM



Keith McIntyre
Vice President and Chief Technologist, Stelcom



Steve Milroy
Wireless Technologist, Immedient



Tony Wasserman
Director, Mobile Middleware Lab, Hewlett-Packard Company

Exclusive Sponsorship Available

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



ONLINE REGISTRATION NOW OPEN

WWW.SYS-CON.COM/WIRELESSEGE2002

Or, To Register By Phone

Call: (201) 802-3069



	Development	Connectivity	Wireless Applications	Hardware	Management
M A Y 8 , 2 0 0 2					
8:30-9:45	Embedded Java (Bill Ray, Network 23 Limited)	A Real Time Model of (3G) UMTS Access Stratum (Niloy Mukherjee, MIT Media Laboratory)	Using Mobility to Streamline Traditional Business Processes (Dan Elliott, CompuCom)	Java Software Performance On Wireless Devices: Myths and Realities (Ron Stein, Nazomi)	LMDS - The Last Mile Enabler of Class 5 Revenues with VoIP (Ed Peters, Ensemble Communications, Inc.)
10:00-11:15	KEYNOTE				
11:30 A.M.	EXPO FLOOR OPEN				
	LUNCH BREAK				
1:30-2:30	XML & Wireless Technologies (Karl Best,OASIS)	Securing Wireless Data Via Smart Card (Joseph Smith, New Dominion Software)	Collaboration for Wireless Warriors (Timothy Butler, SiteScape, Inc)	User Interactivity for Information Appliances (Arie Mazur, Slangsoft)	Leveraging Wireless In Customer Acquisition and Retention (Kenneth Leung, IBM)
2:45-3:45	KEYNOTE				
3:45-4:45	Developing Mobile Web Applications (Tony Wasserman, Hewlett-Packard Company)	In-building Wireless, the Next Frontier (Mary Jesse, RadioFrame Networks)	Turbocharge Mobile Applications with J2EE (Dr. Jeff Capone, Aligo, Inc.)	Cell Phones/ WorldPhones (Speaker TBA)	Mobilize Your Enterprise (Chris Bennett, Freedom Technologies)
5:00-6:00	VoiceXML Workshop (Bryan Michael, BeVocal)	Satellite Broadband (Speaker TBA)	Mobile Portals - The First Step Towards the Mobile Enterprise (Tony Wasserman, Hewlett-Packard Company)	PDA's (Speaker TBA)	Wireless Solution Return On Investment (ROI) In the Real World (Steve Milroy, Immedient)
M A Y 9 , 2 0 0 2					
8:30-9:45	Brew, I-Mode, WAP, and J2ME: How the Battlefield Is Shaping Up at the Start of the Mobile War (Reza B'Far, eBuilt, Inc.)	The Future of IP Mobility (Antti Eravaara, NetSeal, Inc.)	Multimodality: Revolutionizing Our Wireless Lifestyles (Arvind Rao, OnMobile Systems)	Tablets (Speaker TBA)	Marketing Value in Automotive Telematics Through Mobile Communications (Douglas Lamont, DePaul University)
10:00-11:15	KEYNOTE				
11:30 A.M.	Guide to Developing Applications with Micro Java (Kurt Baker, Kada Systems)	EXPO FLOOR OPEN			
	LUNCH BREAK				
1:45-3:00	Building Secure Mobile Solutions (Keith McIntyre, Stelcom)	Smart Card Communications (Bill Ray, Network 23 Limited)	Developing New Applications Using VoiceXML (Jonathan Taylor, Voxeo)	PDA's (Speaker TBA)	Policies and Profiles: The Key to Mobile Data Services (Doug Somers, Bridgewater Systems)
3:15-4:30	Bluetooth™ Wireless Technology and Java™ Technology (Michael Portwood, Exuberance, LLC)	UMTS/3G Networks (Speaker TBA)	Instant Messaging and Wireless Computing Collide (JP Morgenthal, IKimbo)	Transmitters / Base Stations (Speaker TBA)	Total Business Solutions B2E (Speaker TBA)
4:45-6:00	Creating Carrier Optimized Wireless Internet Applications (David Young & Victor Brilon, Lutris)	Wireless LANs/Bluetooth (Speaker TBA)	Rapid Development of Successful Wireless Applications (Rod Montrose, AVIDWireless)	Headphones / Keyboards / Peripherals (Speaker TBA)	Wireless Sales & Marketing (Speaker TBA)

REGISTER BEFORE

March 1

And

Save

Up To

\$400

Plan to Attend the
3-DAY
Conference

PRODUCED BY



May 7 will feature full-day tutorials. Consult www.sys-con.com/wirelessedge2002 for up to-the-minute conference news.

A Banner Ad Custom Tag

BY
EBEN
HEWITT



Quick and painless coding

The banner ad is a bit of a curious creature.

It wasn't so long ago (say, 1998) that the pundits who had for so long decried the gluttonous, indelicate banner ad were immersed in self-congratulation. Click-through rates were dropping below 1% down to .5% by the start of 2000.

Users don't even see them, the pundits claimed. They've learned to block out the banners and scroll past them; people will turn to other forms of advertising, such as the content ad. And perhaps, to an extent, they have.

Consider the number of Web sites you viewed today. How many of those sites feature banner ads? A large percentage I wager.

The curious thing about banner ads is that everybody hates them, everybody "ignores" them, and yet they continue to exist with Terminator-like persistence. The bottom line is this: as long as the marketing people are expecting us developers to schlep banner ads from site to site, we might as well make it quick and painless. As you know, the best way to make code portable and to hide its implementation is to write it as a custom tag. Hence, `<cf_BannerAds>`.

In this article we write a custom tag that does the basic things a banner ad needs to do. We need to generate a random banner ad from the database that stores the ads, display it back to the page at the point the custom tag was called, and then update the number of times that particular banner has been displayed. Finally, we need a way to record a click-through and redirect the user to the target site.

We can fulfill all these requirements in fewer than 50 lines of code in a custom tag. We'll use a Microsoft SQL Server database to hold our data, though an Access

database could be easily substituted here. You should come away with a custom tag that you can use without hassles in a production environment.

Defining the Application

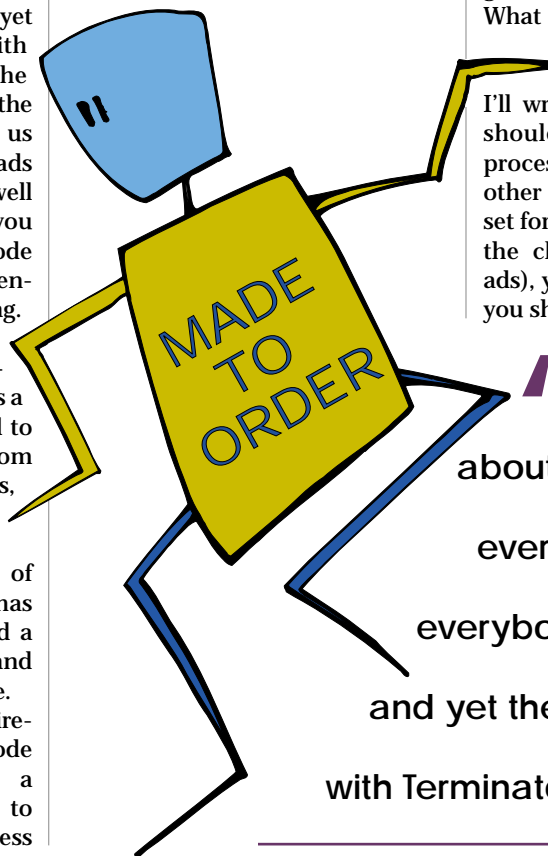
We begin writing our tag (see Listing 1) by defining a banner ad in specific terms. It's comprised of an image and a URL to open if the user clicks on the ad and is often a constant width and height (468 x 60). A company name is associated with it, which we can use in the Alt attribute of the HTML `` tag.

It's probably a good idea for each banner ad to have an associated unique identifier. In the real world,

somebody in accounting is making sure these ads are paid for. Although here we're simply making the banner ad rotator work, a unique ID allows each banner to be associated with a customer account.

What we've done so far allows the banner ad to be displayed. That's only one possible action, however, it can also be clicked on. When that happens, we need to update the number of times the ad has been clicked on and then redirect the user to the URL associated with the ad.

Because this month's issue of *CFDJ* focuses on custom tags, let's detour for a moment to examine why this required functionality is a good candidate for a custom tag. What do you do when you work on a Web site? You don't say to yourself, "Hey, Earl, I think I'll write a custom tag. Now what should that sucker do...?" The process generally starts from the other end. Upon defining a feature set for some required function (e.g., the client wants rotating banner ads), you determine whether or not you should make it a custom tag.



The curious thing about banner ads is that everybody hates them, everybody "ignores" them, and yet they continue to exist with Terminator-like persistence"

BannerID	int	PK
ImagePath	varchar	50
AltText	varchar	50
WebsiteURL	varchar	75
TimesDisplayed	int	
ClickThrus	int	

TABLE 1 Banner ads database table

There are a number of good ways to make this determination. I'm sure you've read about the general criteria: will you want to reuse this code, encrypt it, conceal complexity from less experienced developers, conceal variables from exposure on the calling page, and so on? If you've worked with Java, you might ask yourself: If I were writing in Java, would I make this an object? A banner ad is fairly discrete. It has a couple of different modes: it can be displayed or it can be clicked on, and it behaves differently in each instance. It can have methods (getBanner, setClickThroughs, etc.). You might ask, "Will I need to reuse this code frequently and be able to port it easily to different applications?" As it happens, a banner ad rotator is a good example of when to incur extra application overhead (and possibly extra development time) and write it as a custom tag.

Building the Application

Now that we've determined what needs to be done and that it's best implemented as a custom tag, we can figure out how to write it. Let's start by writing the banner ad into a database table. This table (see Table 1) will be implemented in Microsoft SQL Server, though it should run with little or no modification on other systems.

Once we have our database table, we can enter a quick row of dummy data and then write the query to return the result set. This makes sure that everything is working properly, and it throws us impatient, instant-reward types a bone. Let's walk through the code and see how it works.

Displaying a Random Banner

First, we have to determine what action should be performed. We create a regular local variable called banner.action that we'll use to switch against in the main body of the tag. Possible values are display (the default) and click. We use the Immediate If (IIF) function to determine the value of banner.action. If the variable URL.AdAction is defined, the user has clicked the banner ad; if it isn't defined, then we need to retrieve a new ad and display it.

Certainly the most complicated aspect of the tag is retrieving a banner randomly. Here's the code that does it:

```
<cfset RandomID = RandRange(1,
getBannerIDs.RecordCount)>
```

```
<cfset thisBannerID =
ListGetAt((ValueList(getBannerID
s.BannerID)),RandomID)>
```

First, create a local variable called RandomID that will hold a random number between one and the number of banners currently in the database. Next, create a local variable called thisBannerID to hold the banner we'll display in this call to the tag. As you may know, the ValueList function returns a comma-separated list of the values of each record returned from an executed query (in this case, "getBannerIDs"). We then pass the result of the ValueList as an argument to the ListGetAt function, which retrieves the element in a list from the given position (in this case, the value of the RandomID variable).

Now we've got a primary key, which we use in another query to retrieve all the information required to display an ad: the name of the image, the URL of the advertiser's Web site, and the AltText. Then we simply write out the HTML with the appropriate tag attributes populated with the results of our query. Note that we're storing the name of the image in the database – you need to have the actual image stored somewhere on the Web server (here images/banners/).

SAVE 30% off the annual newsstand rate

JAVA DEVELOPER'S JOURNAL

Offer subject to change without notice

ANNUAL NEWSSTAND RATE

~~\$71.88~~

YOU PAY

\$49.99

YOU SAVE

30% Off the Newsstand Rate

DON'T MISS AN ISSUE!

Receive 12 issues of **Java Developer's Journal** for only **\$49.99!** That's a savings of **\$21.89** off the cover price. Sign up online at www.sys-con.com or call 1 800 513-7111 and subscribe today!

In February JDJ:

Getting Started with Java on PDAs

Build and deploy Java apps
by Rob Tiffany

Pattern Foundations: The Open-Closed Principle

Create systems that are easily maintained and flexible
by Kirk Knoernschild

Coming Out of the JDO Closet

The JDO specification is a promising newcomer in the object persistency arena – for J2EE projects large and small
by Yaron Telem and Shay Litvak

Use XML Inside Existing Applications...

...without learning the XML native programming models
by John Goodson



It's probably a good idea for each banner ad to have an associated unique identifier. In the real world, somebody in accounting is making sure these ads are paid for"

ABOUT THE AUTHOR
Eben Hewitt is a Macromedia Certified ColdFusion developer and author of Core ColdFusion 5 and the interactive CD-ROM Eben Hewitt's ColdFusion Training Course. He writes applications for Arizona's largest Internet service company.

When we create the ad for display, we include three parameters in the construction of the link: AdAction tells the custom tag that we need to run the code inside the

"click" case; the "thisURL" parameter will tell the tag the address of the Web resource to display when the user clicks on the ad; finally, the "BannerID" parameter is passed to tell the tag which banner to update the number of click-throughs for.

Now we need to update the number of times this particular ad was displayed, so we hit the database again and set the TimesDisplayed column for the ad with the current BannerID to TimesDisplayed+1.

As these are all the requirements for displaying an ad, we exit the case, and thereby the switch and the tag.


Now we need to create the code that runs in the event that a user clicks on a banner.

ClickThroughs

All that remains in the "click" case is to handle two events: update the number of click-throughs for this particular ad and relocate the user to display the Web resource associated with the ad. We update the

ClickThroughs in much the same way we updated TimesDisplayed. Using the "thisURL" parameter we set up in displaying the ad, we simply pass it to a <cflocation> tag and we're done.

To use the tag on your site, set up the database and populate a few rows, then call <cf_banner>. Listing 2 shows a simple table and how to call the custom tag for use on your site.

As you can see, this setup is compact and easy to use. You can expand and improve on this basic tag a few ways. For instance, instead of hitting the database so much, try using the <cfsavecontent> tag, new with ColdFusion 5, to store your banner info. Also, you might create a "data source" attribute of the tag to make it more flexible. Last, if your system requires it, add a column to the database called "weight", and use it to calculate which banners should be shown more frequently. 

[@EBEN@CYBERTRAILS.COM](mailto:EBEN@CYBERTRAILS.COM)

Listing 1

```
<!------
template: <cf_BannerAds>
use: called from index.cfm
purpose: display and record hits and click thrus of banner
ads
notes:
author: e hewitt, eben@corecoldfusion.com
created date: 12/10/2002 1:01:29 PM
last modified:
----->

<cfset banner.action = #iif(IsDefined("URL.AdAction"),
DE("click"), DE("display"))#>

<cfswitch expression="#banner.action#">
<cfcase value="display">
<!--get all the banner ids----->
<cfquery name="getBannerIDs"
datasource="#Request.App.Datasource#">
```

```
SELECT BannerID
FROM BannerAds;

</cfquery>

<!--choose a random banner from known ids ----->
<cfset VARIABLES.RandomID = RandRange(1,
getBannerIDs.RecordCount)>

<cfset VARIABLES.thisBannerID =
ListGetAt((ValueList(getBannerIDs.BannerID)),
VARIABLES.RandomID)>

<!--Get a banner ad with that ID --->
<cfquery name="getOneBanner"
datasource="#Request.App.Datasource#">

SELECT BannerID, ImagePath, AltText, WebsiteURL
FROM BannerAds
WHERE BannerID = #VARIABLES.thisBannerID#;

</cfquery>

<cfoutput>
<a href="index.cfm?AdAction=Click&thisURL=#
```

```

    SET ClickThrus = ClickThrus + 1

    WHERE BannerID = #URL.BannerID#;

</cfquery>

<!--and redirect----->

<cflocation url="#URL.thisURL#">

<cfabort>

</cfcase>

</cfswitch>

```

Listing 2

```

<table border="0">

    <tr>

        <td colspan="2" align="center">

            <!--call the tag----->

            <cf_bannerads>

                </td>

            </tr>

        </table>

```

CODE
LISTING

FEBRUARY CFDJ 23

ColdFusion Feature

By Andrew Cripps

Smart Tags

Microsoft invented a technology called Smart Tags. Smart Tags allow text to be marked up with actions, such as hyperlinks, when the text is displayed.

They planned to introduce the technology with Windows XP. However, since public opinion was strongly against it, Microsoft decided to de-emphasize it. You can still create and use Smart Tags, but they're not prominent in Windows XP.

This article provides an overview of the technology and then shows how you can use ColdFusion to implement a simple version of Smart Tags. You can download the custom tag, `cf_smarttag`, from the **CFDJ** Web site (www.sys-con.com/coldfusion/sourcecf.cfm).

Overview of Microsoft Smart Tags

A typical Web page has text that includes hyperlinks to other Web sites or resources on the World Wide Web. The author usually creates the hyperlinks when the document is written, and those hyperlinks are part of the authored Web page. However, Microsoft thought it would be interesting to take the hyperlink concept one step further: the author of the document creates some text, then someone (possibly the author, but perhaps not) can create a set of keywords that are associated with shortcuts in Windows that could provide a hyperlink to another Web site, open another application, or perform some custom action.

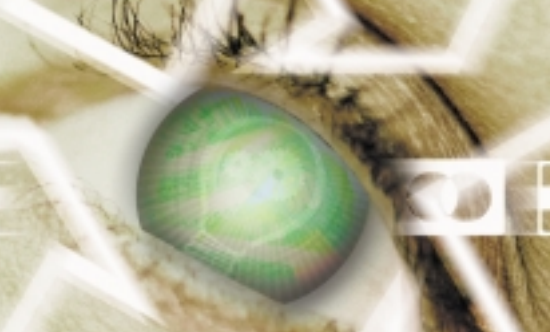
For example, if an author writes "The most interesting news last month was....," Smart Tags technology allows you to associate the word "news" with a hyperlink to `msn.com`, or some other Web site. When a user goes to a site that's Smart - Tags enabled and opens a Web page, the user sees squiggly purple lines under the keywords (like the lines in Microsoft Office documents if you've made a spelling mistake). When the user hovers over a keyword with a squiggly underline, a Smart Tags icon appears with a dropdown selection defined for the particular Smart Tag associated with the keyword. Most typically, the action is to hyperlink to one or more (Microsoft) Web sites. Note that the author

Implement a simple version of Smart Tags using ColdFusion

of the document may not have intended this hyperlink when writing the document; the hyperlinks and other actions are added when the page is displayed, depending on the definitions of the Smart Tags on the computer the page has been requested from.

To use Smart Tags, define the Smart Tag and what it does, then create a list that relates the Smart Tag to a keyword. When a page is rendered that contains those keywords, the user can click on an item in the Smart Tag dropdown menu to take one of the actions defined by the Smart Tag.

For more information, visit www.microsoft.com/office/dev/xp_06_2001/odc_st/webpages.htm.



Tags. The text on the CFML template that will be affected by Smart Tags would be marked up as follows:

```
<cfsmarttags>
  text to change goes here. This
  could be a ColdFusion #variable#
</cfsmarttags>
```

Why Smart Tags Stalled

Smart Tags received a rough reception from the public. Although many people thought the technology was interesting, they believed Microsoft would use it to increase traffic to their own Web sites and servers. For example, keywords such as “news,” “sports,” and “servers,” would all link to Microsoft’s Web servers.

Many critics thought Smart Tags would primarily allow Microsoft to re-edit a Web site without the author’s knowledge or permission.

There are, in fact, some legitimate uses for Smart Tags technology. Imagine a large company with offices worldwide, and each office has a local intranet. With Smart Tags technology, whenever the word “intranet” appears in documents posted on any of the company Web sites, Smart Tags technology would allow each office to associate a hyperlink with that keyword so that it pointed to their local intranet. This is clearly an improvement for each office. The head office sends documents that can be hosted on several corporate intranet sites, and the hyperlinks are now separately maintained and consistent across all documents. Wherever the word “intranet” occurs, the same hyperlink would be in place.

A Simple Smart Tags Implementation

After that summary of Microsoft Smart Tags technology, I will now show you how to build a simple version using ColdFusion. In my implementation we will define keywords and just one action: a hyperlink. When a page is displayed that contains one of the defined keywords, a hyperlink is added to the keyword. In this section I describe the basic implementation; later I’ll provide some ideas about how you might extend it.

The solution breaks down into the following parts:

- Wrapping text: paired custom tags
- Storing keywords and their associated hyperlinks
- Generating content
- Exiting gracefully

The first part of the solution is to design a Smart Tag custom tag that can wrap the text we want to use with Smart

Tags. The Smart Tag custom tag could be written at the start and end of each page, or perhaps into header and footer files that are included on each CFML template on a site.

The second part of the solution is to design a way to store the keywords and their associated Smart Tag action – a hyperlink. One convenient way to store this association is in an XML file. The format I chose is shown in Listing 1.

The XML document has a single root element, “<smarttags>” in this case, and that element has any number of “<smart-tag>” elements that relate keywords (the field elements) to Smart Tag actions (the URL element). I’ll show ways to extend this XML definition later in the article.

The Smart Tag custom tag uses a default XML file, smarttags.xml, but you can also pass in the location of the XML file through the “smarttagsfile” attribute. For example, you could call the Smart Tag custom tag like this:

```
<cfsmarttags smarttagsfile="c:/inet-
pub/wwwroot/news/sport_smarttags.xml"
>
  text to change goes here. This
  could be a ColdFusion #variable#
</cfsmarttags>
```

and the sport_smarttags.xml file containing the keyword/hyperlink pairs would be used to mark up the text. (You could even use the smarttagsfile attribute in combination with a ColdFusion variable – perhaps the location of the smarttagsfile to use in a particular section of a large Web site is retrieved from a database.)

- The third part of the solution is generating the modified content. The Smart Tag tag has to be a paired tag – with a begin tag and a matching end tag, just like <cfquery> or <cfhttp> – since it will be used to wrap text that will be changed by the Smart Tags. The custom tag has to:
- Take a copy of the content the ColdFusion server has generated so far
 - Wipe out whatever was in the generated content variable
 - Read the XML file and modify the copied content according to the keyword/hyperlink pairs
 - Output the modified content

Macromedia has built into CFML the ability to create paired custom tags and to manipulate whatever it generates (most often HTML) for display in the browser. In CFML you can make some final changes to the output before the ColdFusion server sends it off to the browser to be displayed. You can access the content the ColdFusion server sends to the browser through the variable “thistag-generatedcontent”.

Once a copy of the thistag.generated-content variable has been taken:

```
<cfset mytext = thistag.generatedcon-
tent>
```

the tag looks for keywords defined in the XML Smart Tags file, and replaces each keyword it finds with the associated URL. Lines 60–69 of smarttag.cfm use the mid-string function that’s user-defined and declared in the file functionlibrary.cfm. (A valuable re-source for user-defined functions that you can download and use can be found at www.cflib.org.)

You may think that it would be easier to define the midstring function in the Smart Tag custom tag. Unfortunately, this won’t work because a paired custom tag executes twice: once in start mode and once in end mode. To ColdFusion it appears as though the user-defined function midstring has been declared twice, and you get this error:

```
The routine midstring has been
declared twice in different tem-
plates.
```

To avoid this problem, use cfinclude to include a CFML template that contains your user-defined function definition. In the smarttags.cfm custom tag, nothing is required in start mode (lines 43–44); however, if you wanted to do something the first time the tag runs, this is where your code would go. All the processing occurs when the closing “</cfsmarttag>” is encountered on the CFML template. The custom tag is executed in end mode, and lines 45–71 copy the generated content, make changes, and write out the modified content.

The last thing to note in this solution is the use of <CFEXIT method=“ExitTag”>. The CFEXIT tag allows a custom tag to pass control back to the caller of the custom tag. It’s used in the Smart Tag custom tag if any errors are caught while trying to read the XML settings file. If the XML settings file is not readable, the tag will exit cleanly without making any changes to the generated content.

Ideas for Extensions

The ColdFusion implementation is very simple right now. All it does is gain access to the generated content and, before that content is sent back to the browser, it makes some substitutions in the generated content variable.

Microsoft's implementation is much more complex. They've allowed for the fact that you might want to use several sets of Smart Tag definitions that might contain the same keyword. They use namespaces to avoid confusion about what to do when that keyword is encountered. Microsoft's implementation also shows a squiggly line under each keyword when the page is displayed in Internet Explorer: no action is taken automatically – you have the option to take an action such as hyperlinking to another location when you hover over the keyword. How might you extend and improve on this CFML implementation?

You could extend the XML file with additional actions. In my simple example the action is a hyperlink to the provided URL. But you could create an XML definition that associates keywords with CFML templates that are executed when one of the keywords is encountered. Another example might be to create an XML defi-

nition that associates keywords with CFML templates that take actions such as sending e-mail when one of the keywords is encountered.


You could extend the Smart Tag to work through a set of definition files instead of one. Rather than have the XML definition file stored locally on the ColdFusion server that returns the page for display, the definitions could live anywhere on the Web. You could use an attribute similar to the `smarttagsfile` attribute to find the location of the settings file on the Web and perhaps retrieve the file using CFHTTP (although this may raise performance issues).

You could also extend the XML definition to replace keywords, not with a hyperlink but with almost any HTML or dynamic HTML you like. You might use this to create a simple menu scheme, replacing some keywords specifically chosen to identify menu choices, and then allowing the Smart Tag custom tag to substitute your keywords with HTML or JavaScript. You could almost create an XML definitions file that would act as a set of macros for a Web site, consistently changing text and HTML. To make this work, you would also need to extend the basic `midstring` user-defined function.

To create something similar in appearance to Microsoft Smart Tags, you need to embed objects using the HTML element `<object classid="">`. You'd want to access the same object that Microsoft uses in Smart Tags so that when you hover over a link, you get the icon and the options dropdown box appears.

The Code

The files included in the ColdFusion implementation on the *CFDJ* Web site are:

- **`callsmarttags.cfm`**: An example page showing how the `cfsmarttags` tag is used
- **`smarttags.cfm`**: The custom tag itself
- **`smarttags.xml`**: An example XML definitions file consisting of keyword/hyperlink pairs
- **`functionlibrary.cfm`**: This tag uses the `midstring` function available from cflib.org 

About the Author

Andrew Cripps is a development manager at Macromedia in Boston where he works on ColdFusion and Flash technologies. He has over 13 years of experience in the computing industry. Andrew holds an MS in computer science and an MA in philosophy from Canadian universities.

 ANDREW@CRIPPS.NET

Listing 1

```
<?xml version="1.0"?>
<smarttags>
  <smarttag>
    <field>microsoft</field>
    <url>www.microsoft.com</url>
  </smarttag>
  <smarttag>
    <field>apple</field>
    <url>www.apple.com</url>
  </smarttag>
  <smarttag>
    <field>java</field>
    <url>sun.java.com</url>
  </smarttag>
  <smarttag>
    <field>cripps</field>
    <url>www.cripps.net</url>
  </smarttag>
</smarttags>
```

CODE
LISTING



The code listing for
this article is also located at

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

WWW.EKTRON.COM
www.ektron.com/demo

Functional Literacy

BY
KEVIN
SCHMIDT



Now that some of the hype surrounding ColdFusion 5 has begun to die down, you may find yourself ready to tackle one of its new features – user-defined functions.

Step-by-step with user-defined functions

User-defined functions (we'll call them UDFs) are one of the most talked about features of ColdFusion 5 and certainly one of its best. UDFs are powerful, yet simple to create. They're also fast and highly flexible. (For more detailed information about UDFs, check out the article by Raymond Camden in this issue of **CFDJ**.) In the meantime, let's create our own.

Replacing Custom Tags

Last April I wrote an article focusing on the creation of a custom tag to calculate a mortgage payment (*CFDJ*, Vol. 3, issue 4). Now it's time to turn that custom tag into a UDF. This is an ideal condition for a UDF, because we're performing a simple calculation on several values that are passed in and the UDF is able to do this much faster than a custom tag. Not to say that UDFs are always a better alternative to custom tags, however, in this case they offer a clear advantage.

<CFSCRIPT>

The `<cfscript>` tag is the key to UDFs, as all UDFs must be written in `<cfscript>`. (For those of you who aren't familiar with `<cfscript>`, it's ColdFusion's scripting language and is very close in syntax to JavaScript.) Knowing that `<cfscript>` is the key, the first thing we'll do is create a new file and drop in the `<cfscript>` beginning and end tags.

```
<cfscript>
//A user defined function for
calculating mortgage payments

</cfscript>
```

You'll notice that the comment also follows the syntax of JavaScript,

not ColdFusion. (All the code for the Mortgage Payment UDF is available in Listing 1.)



Creating the Function

Now that we have our `<cfscript>` tags, we need to give our function a name. This is done by using the function keyword, followed by the name for the function, a pair of parentheses, and a left-hand brace. Here's how we proceed to name the function mortgagepayment:

```
function mortgagepayment() {
```

This sets up the function and gives it the name mortgagepayment. When we call this function, we call it just like any other function in ColdFusion.

Passing in Parameters

To do the calculation, we'll need to pass in three values: the balance of the loan, the term of the loan, and the interest rate. To do this, we must let the function know that it requires these three parameters. We

accomplish this by specifying the names of the parameters in the parentheses after the function name. The code below illustrates this:

```
function mortgagepayment
(balance, term, rate) {
```

Your function expects to receive these three parameters. All three must be passed in, as these are required parameters. If they're not passed in, we'll receive an error. We can also pass in optional parameters. All parameters passed into the function are stored in an array called Arguments, so to access optional parameters we'd reference their position in the array.

Creating Local Variables

Once we have the function defined and the parameters set, we'll have to create a variable inside the function to hold the payment value. Variables are created inside UDFs using the "var" keyword and are only available to the functions they're created in. Thus, to create a variable to hold the value of the mortgage payment, we use the following code:

```
var monthlypayment = 0;
```

With this, we now have a variable local function to store our payment value.

Doing the Calculations

To arrive at the monthly payment we need to do a few calculations, the first being the term. The term is passed in years and needs to be converted to months. The next piece of our code takes care of this:

```
term = term * 12;
```

ABOUT THE AUTHOR

Kevin Schmidt is a Macromedia-certified ColdFusion developer and author of Macromedia ColdFusion 5: Training from the Source. He is the manager of the Des Moines ColdFusion User Group.

ColdFusion 5 UltraDev Studio from Macromedia

REVIEWED BY
TOM MUCK



The merger of Allaire and Macromedia created much speculation in development circles as to the future of ColdFusion and ColdFusion Studio.

With the release of ColdFusion 5 UltraDev 4 Studio, Macromedia has committed to providing the best possible development environment for the ColdFusion programmer.

The package isn't an integration of the two products, but a packaging of two solid programs that can be used together to develop dynamic Web sites. ColdFusion Studio 5 is the latest release of the classic CF development environment. UltraDev 4, which came out in December 2000, takes the core functionality of Dreamweaver and adds server-side functionality to the design environment. This release coincides with the release of ColdFusion Studio 5.

ColdFusion Studio 5 vs UltraDev 4

The two programs work well together to provide the best of both worlds when building your CF pages. ColdFusion Studio takes a more code-centric approach, whereas UltraDev features a design environment while still allowing access to the code.

ColdFusion Studio is a robust, mature program that's on the shelf of every ColdFusion developer who is serious about his or her work. Each version of Studio is better than the previous one. Version 4.5 had some

issues with memory leaks that affected many users, but those issues seem to have been resolved. CF Studio 5.0 is a stable, lean program that occupies a small footprint in memory and allows easy access to multiple documents at any time.

UltraDev 4, despite its version number, is only a second-generation program. Six months after the release of UltraDev 1, UltraDev 4 hit the shelves with much better support for ColdFusion. Still, as a development environment for ColdFusion, it has some issues. Certain combinations of tags and/or expressions can cause the program to rewrite your code; however, through careful coding you can eliminate these problems. Once you've mastered the limitations of working in the UltraDev environment, the speed at which you can deliver completed Web applications dramatically increases.

UltraDev has three views – design, code, and split – that are synchronized while you work. For example, when you select a table row in design view, the code is highlighted in code view (see Figure 1). This makes it easy to manipulate your design in ways you can't easily do by hand coding. For example, if you wanted to make that table row wider, simply drag the table border in design view. The appropriate width attributes are written to the <tr> and <td> tags automatically. Dreamweaver is known for generating clean HTML code.

ColdFusion Studio can be set up as UltraDev's external editor, but it's a manual step: after loading UltraDev, set your external editor preference to CF Studio. This is one area that I expect to see improved in future editions of the package and future versions of UltraDev. CF Studio does this already by integrating a

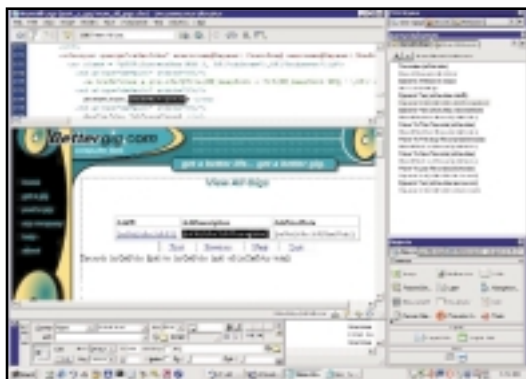


FIGURE 1: UltraDev features a split code/design view

VITALS

ColdFusion 5 UltraDev 4 Studio:
Macromedia

Address: 600 Townsend Street
San Francisco, CA 94103

Phone: 415 252-2000

Web: www.macromedia.com

Environment:

Windows 98/2000/Me/NT 4/XP

Pricing: \$599 for the full version. Upgrades from UltraDev (any version) or CF Studio (any version) are \$299. Upgrades from Dreamweaver (any version) are \$399. Also, CF Studio is available as a standalone product (without UltraDev) for \$495 for the full version or \$169 as an upgrade from a previous version of CF Studio.

Dreamweaver/UltraDev button with the CF Studio interface. A simple click of the button launches the current page into the UltraDev environment. It would be nice to see a ColdFusion Studio button on the UltraDev toolbar as well.

New Features

CF Studio 5 looks remarkably like version 4.5. The upgrade can be considered more of an enhancement to an already great program than a version upgrade. If you're accustomed to working with CF Studio 4.5, you won't have to change your workflow to use the new version.

The new features of ColdFusion Studio are few, but substantial. XHTML 1.0 support based on the W3C specifications is finally built into the program. This was one of the top requests for enhancements. Also, the group of Resource Windows has a new addition – a secondary Files window so you can have two directories open at once with the ability to drag-and-drop files between the two tabs (see Figure 2).

More new features include a Custom Tag Wizard to ease in the cre-

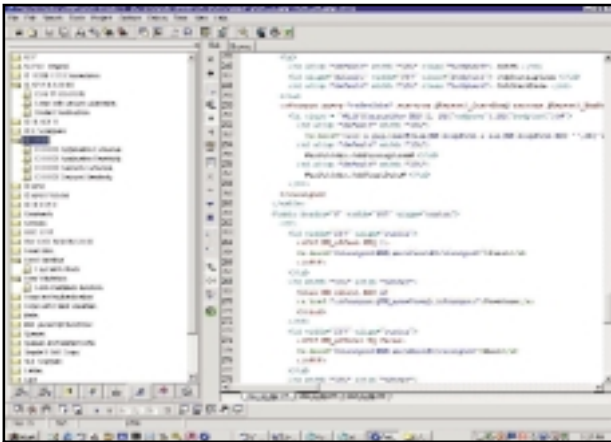



FIGURE 2: CF Studio 5

ation of custom tags, a Configuration Wizard that eases the configuration of the Studio environment, and folder deployment. A new feature that I particularly like is the ability to right-click on an include file reference and choose Edit Include File; this is a real time-saver when editing pages that have lots of includes. Also, Auto Backup allows you to specify a backup directory and a particular time when your files will be backed up – the default is every 10 minutes. Backup files are stored for 10 days, or whatever you state in the Settings >> Editor configuration. Also, Fireworks integration is new with a contextual menu item that allows you to edit the current image in Fireworks. Fireworks is not part of the package, however, and must be purchased separately.

Of course, the new CF Studio also has full support for all the new CF Server 5.0 tags. These were available as an add-on for CF Studio 4.5 as well.

UltraDev doesn't sport any new features in this combined release. However, since it's been around for over a year there are plenty of third-party add-ons available on the Web that can greatly enhance the integration of the two programs. Dreamweaver has always had a rich tradition of third-party add-ons (called extensions in DW lingo) and UltraDev takes this a step further by allowing the automatic creation of custom extensions with the Server Behavior Builder. Some extensions of interest include a Snippets extension that works in tandem with CF Studio's Snippets resource window, and a set of ColdFusion objects similar to the CF objects available for Dreamweaver 2, only more refined. Both are available from www.massimocorner.com.

Summing Up

ColdFusion developers have always been ahead of the game with the development environment of CF Studio. It's a feature-rich, dedicated environment that can dramatically speed up the coding process if used properly. By combining this tool with UltraDev, Macromedia speeds up the process even more. Whatever CF Studio is lacking is included in UltraDev, and what UltraDev is lacking is included in CF Studio. It's a combination that works, and a worthy upgrade if you have a previous version of either program. 

ABOUT THE AUTHOR

Tom Muck is a senior applications developer for Ingram in northern Virginia. He develops back-end applications for expedited, electronic communications. Tom is the coauthor of three UltraDev books including Dreamweaver UltraDev 4: The Complete Reference.

@ TOMMUCK@BASIC-DRUMBEAT.COM

SAVE 16% off
the annual cover price

ColdFusion Developer's Journal

ANNUAL NEWSSTAND RATE

~~\$107.88~~

YOU PAY

\$89.99

YOU SAVE

\$17.89 Off the Annual Newsstand Rate

Receive 12 issues of
ColdFusion Developer's Journal
for only \$89.99! That's a savings of
\$17.89 off the annual newsstand rate.

Visit our site at www.sys-con.com

or call 1-800-513-7111

and subscribe today!

Here's what you'll find in every issue of ColdFusion Developer's Journal

- Exclusive feature articles
- Interviews with the hottest names in ColdFusion
- Latest CFDJ product reviews
- Code examples you can use in your applications
- CF tips and techniques





INTERNATIONAL WEB SERVICES CONFERENCE & EXPO



INTERNATIONAL JAVA DEVELOPER CONFERENCE & EXPO



INTERNATIONAL XML CONFERENCE & EXPO



FUNDAMENTALLY IMPROVING THE SPEED, COST AND FLEXIBILITY OF BUSINESS INTEGRATION

Sponsored by:



Sponsored by:



Produced by:



Web Services – Skills, Strategy, and Vision

For the developer, the latest tools and techniques...

For the architect, the latest designs....

For the VP/CIO, technology management issues

- Invaluable information in the form of discussions, presentations, tutorials, and case studies
- Unmatched Keynotes and Faculty - gurus in the Java, XML, .NET, and Web Services world
- The largest independent Web Services, Java, and XML Expos
- An unparalleled opportunity to network with over 5,000 i-technology professionals

New in 2002!

Each track will feature "Hot Breaking" Sessions to keep attendees up-to-the-minute on Emerging Technologies and Strategies!

Featuring 2 Keynote Panels and Industry Perspectives from the Visionaries shaping Next Generation Technologies and Business Strategies

2002

ONLINE
EARLY BIRD REGISTRATION
NOW OPEN!
SAVE \$400

JUNE 24-27
JACOB JAVITS
CONVENTION
CENTER
NEW YORK, NY

OCTOBER 1-3
SAN JOSE
CONVENTION
CENTER
SAN JOSE, CA

FOR MORE INFORMATION

SYS-CON EVENTS, INC
 135 CHESTNUT RIDGE RD.
 MONTVALE, NJ 07645

201-802-3069
WWW.SYS-CON.COM

Who Should Attend...

- Developers, Programmers, Engineers
- Senior Business Management
- Senior IT/IS Management
- i-Technology Professionals
- Analysts, Consultants

Who Should Exhibit...

Java, XML, Web Services, and .NET Technology vendors, staking their claim to this fast-evolving marketplace



Web Services Edge Conference Committee



Alan Williamson
 Java Track Chair
 Editor-in-Chief
Java Developer's Journal

Java Track

- Java 1.4: What's New?
- Building Truly Portable J2EE Applications
- J2ME: MIDlets for the masses
- WebScripting Technologies: JSP/CFML/Velocify
- Where is Swing in this New Web Services World?



Ajit Sagar
 XML Track Chair
 Editor-in-Chief
XML-Journal

XML Track

- XML Web Services: Standards Update
- Using XML for Rapid Application Development and Deployment with Web Services
- Unlocking the Promise of XML: From Hype to How-To
- The Use of XML Technologies to Enhance Security
- Content Management with XML



Sean Rhody
 Conference Tech Chair
 Web Services Track Chair
 Editor-in-Chief
Web Services Journal

Web Services Track

- Starting Out in Web Services: Fundamentals of Web Services (WSDL, UDDI, SOAP)
- Exploring .NET myServices Initiative
- Standards Watch: Reviews and Discussions of the Interactions of All the Relevant Standards
- Guarding the Castle: Security and Web Services
- The Microsoft Way: .NET, Passport, and other MS Technologies for Web Services
- Laying Down the Rules: Web Services and Business Process Engines
- Practical Experiences with Web Services and J2EE
- The Odd Couple: Making .NET and J2EE Work Together



Andy Astor
 IT Strategy Track Chair
 International Advisory Board
Web Services Journal

IT Strategy Track

- Key Architectural Issues in a World of Web Services
- .NET vs J2EE – From Religious Wars to Fact-Based Decision Making
- Getting Started with Web Services
- Selecting a Framework: Toolkit, Platform, or Roll Your Own?
- Infrastructure Vendors—A Map of the World
- Extreme Programming and Web Services Projects: Defining ROI
- Standards You Need to Know About
- Best Practices You Need to Insist On
- Introduction to Business Rules and Rules Engines
- Panel Discussion- Web Service Business/Economic Models



Jim Milbery
 Vendor Track Chair
 Product Review Editor
Java Developer's Journal

Vendor Track


- .NET, J2EE, JMS and other Messaging
- Case and Development Tools
- The Use of Testing Tools
- How-to Demonstrations



ColdFusion Feature

By CHARLES AREHART

Unlocking Restricted Use of **CFFILE, CFCONTENT, and More**



Many developers know that the CF administrator can restrict your ability to use several tags that not only provide very useful capabilities, but can also be used to provide unauthorized access to server resources. What many don't know is that they can arrange to use those restricted tags on a case-by-case basis with the administrator's permission.

In this article, I introduce the Unsecured Tags Directory feature, which most developers don't know about and many administrators don't understand. This feature can be the key to unlocking tags that might otherwise render a developer's intended application design difficult or impossible. I'll also explain the ability to restrict tags, for those unfamiliar with that feature.

The Closed Door to CFFILE, CFCCONTENT, etc.

Have you ever wanted to use a tag such as CFFILE, but when you tried, you received the following message:

Error Diagnostic Information

The CFFile tag is disabled.

The administrator has chosen to disable this functionality on this server unless executed from a specified directory.

The problem, as the message clearly states, is that the administrator has chosen to disable or restrict use of this tag (and likely others), which might be used in an unacceptable and security-compromising way. The reasoning behind the ability to restrict such tags, and the choice by an administrator to do so, is logical and defensible.

I'll explain in more detail. What many developers and administrators don't know is that there's a way to permit access to these otherwise restricted tags. It can be done in a controlled manner that requires the administrator's complicity. The end of the error message even offers a clue as to how, but those few who do notice it still may not understand what it's implying.

Why the Closed Door?

The problem with permitting unfettered use of the CFFILE tag is that it's quite powerful and can be used for many things, including deleting files on the server or moving them around. Since the CF Server generally has rights to almost all directories on a server, it's naturally a scary prospect for a server administrator to let all CF developers access whatever files they want.

Using restricted tags

The administrator can use the “Basic Security” panel in the Administrator to select which of a handful of listed (and potentially dangerous) tags should be restricted so that no CF developer can use them. At least that’s the way it appears, but this article will demonstrate that it’s not entirely true.

The screen in the Administrator used to set these restrictions is shown in Figure 1. As of CF5, the Basic Security features are accessed through the new “Security” link listed below the “Home” and “Logout” links at the top left of the screen. The “Tag

at the top of the screen. The [Tag Restrictions](#) link shows this page, which indicates the tags that can be restricted. It also offers the relatively new Unsecured Tags Directory. (In CF 4.5, this page is under the “Basic Security” link in the Administrator’s left nav bar and appears pretty much the same.)

By default, all the tags are unrestricted (all checked). Anyone reading the screen or the CF Administration manual will likely think that turning these tags off is the “safest thing to do.” Most hosted sites run by CF administrators with at least some knowledge of the restriction option will likely disable the tags.

Sadly, it's a bit like throwing the baby out with the bathwater. There is indeed reason to fear that some might misuse a tag, but one sore point is that it's an all or nothing proposition. Once disabled, that tag can't be used at all by any developer on the server (unless you know the trick that I'll demonstrate in this article).

It's certainly reasonable to fear that someone may misuse the CFFILE Action="Delete" or Action="Move" attributes, but what if you want to use the useAction="Upload" option to allow your users to upload a file to your server? Perhaps your application needs to enable uploads of résumés, product photos, or legal documents. With the CFFILE tag disabled, your application is dead in the water. It would be nice if we could restrict certain attributes or attribute/value pairs rather than the entire tag.

Another Potentially Challenged Application

Another good example where this “all or nothing” approach can be a problem is when a developer wants to use CF to create an application that serves a repository of files available for download. Again, it could be photos, executable programs, documents, etc. He or she may want to permit end users to access these files only if they’ve logged in or have perhaps paid a fee. Creating a

What About Advanced Security?

You may be wondering why CF can't be a little smarter about the restriction of tags and simply limit the files or directories your program is allowed to access. If you could identify an application user or the owner of the application, why not just restrict which files that user or application owner can access? It involves setting up Advanced Security where you map users to resources, so people see only what they should. Because it's a bit harder to set up and that feature came out only in release 4, the simpler "Basic Security" mechanism of restricting tags is more prevalent. That basic approach needn't be a showstopper if you want to allow limited rather than total restriction of the tags, as this article shows.

Web interface to perform authentication or accept payment is beyond the scope of this article, but well documented in other articles and books.

The issue here is how do you make those files available to the end user? While some would say, "Give them a user ID and let them FTP into a server," others would argue for a cleaner, more integrated approach. They may want the Web application to provide the file for download upon request. For this type of application, CFCONTENT comes to the rescue with its File attribute that allows you to send a given file's contents to the browser user (and even set its MIME type with the TYPE attribute).

The problem is that, again, the CFCONTENT FILE attribute would allow you to access any file and directory that the CF Server has access to (assuming you're not working under CF's Advanced Security approach). That's a problem, so many administrators restrict the tag. Again, if you want to do any sort of file serving, as described in this example application, it appears you're out of luck.

CFCONTENT is another example of a tag that should be restricted by attribute. While the aforementioned FILE attribute is risky and the DELETEFILE attribute is even riskier, the TYPE attribute can be used without either of those for an entirely different purpose. It simply sets the MIME type for the content of a page being created using CFML. This tag/attribute pair is especially critical for developers serving Wireless Markup Language applications or creating data to be read by a client as a spreadsheet. I do wish Macromedia would allow that CFCONTENT TYPE attribute to be permitted while restricting the other two.

The Tags – and Attributes – That Can Be Restricted

We've mentioned that CFFILE and CFCONTENT are tags that can be restricted. What are the rest? Prior to ColdFusion 5, the tags that could be restricted were CFCONTENT, CFDIRECTORY, CFFILE, CFOBJECT, CFREGISTRY, CFADMINSECURITY, and CFEXECUTE.

As you can see in Figure 1, release 5 added CFFTP, CFMAIL, and the new tag CFLOG to that list. It also added the ability to restrict not only tags but specific attributes of tags, namely DBTYPE=DYNAMIC and CONNECTSTRING, which are new attributes that can be used on any of several tags related to database processing (such as CFQUERY and CFSTOREDPROC). For the sake of this article, however, the interesting new feature is that we can now restrict by attribute rather than by tag only, as discussed in the previous sections.

Perhaps a future release might allow restriction of just CFFILE's Action="Move" or Action="Delete" attributes, while permitting Action="Upload"; I've lodged that as an enhancement request for the next release of CF code-named Neo. (Then again,

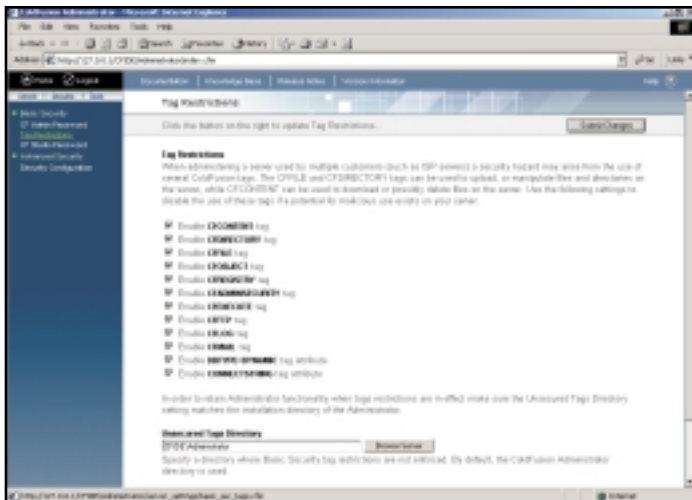


FIGURE 1: Basic Security “Tag Restrictions” page

MACROMEDIA

www.macromedia.com/go/usergroups

permitting uploads has its own security risks, as described in Knowledge Base Article 17407, "Security Best Practice: Evaluating the Risks of Allowing Uploading and of Attached Files on Your Server.")

Going back to the new list of restricted items in CF5, you may wonder why CFMAIL is listed. Though it pains me to point this out for fear it might be abused in pre-release 5 servers, the MIMEATTACH attribute of CFMAIL has always been available to send virtually any file off the server as an attachment in an e-mail sent via CFMAIL. Shocking news though that may be, it's also scary to think that a server administrator might disable CFMAIL entirely, now that it's possible. Again, I've requested an enhancement for Neo that this tag attribute be separately restricted, so that other more reasonable uses of CFMAIL can still be permitted.

It's always a dicey proposition explaining how security works (or doesn't), even if it helps solve problems. That's the nature of security, however. There's no sense living with your head in the sand. Folks who abuse systems often learn of things before those who control them, and they tend to keep their discoveries to themselves. As I so often write in this column, forewarned is forearmed.

Controlled Circumvention of the Restrictions

If I'm adverse to sharing security risks, why am I offering an article called "Unlocking Restricted Use of CFFILE, CFCONTENT, and More." I'll now show how to get around these sort of "global" tag restrictions. Am I again giving away the keys to the kingdom? No. In this case I'm sharing an ability to use a feature that can generally be enabled only with the concurrence and approval of the CF Administrator.

While an administrator *can* restrict various tags, and perhaps should – as in the case of a server shared by multiple, unrelated users – there are times, as we've seen, when this restriction can severely hamper a developer's intention to create some application feature.

Fortunately, release 4.5 and forward of ColdFusion provides an out for us in the form of the Unrestricted Tags Directory. Here, finally, is the key feature being introduced in this article. It's likely big news to many: any templates run in that directory (whether run directly, or called as a custom tag or by way of CFINCLUDE) will indeed be able to run any tag, regardless of whether it's restricted.

The option for setting this special directory is controlled by a text box appearing at the bottom of that Basic

Security "Tag Restrictions" page shown in Figure 1. Many don't even notice the option. The Unrestricted Tags Directory form field is used to specify which directory on the server holds templates that are allowed to use the otherwise restricted tags. That's pretty much it. You may want to try it out.

Some additional background may be useful when considering the value to be chosen for this setting. As you can see in the figure, the value is set to "CFIDE\Administrator," which is where the CF code supporting the ColdFusion Administrator itself is located. Certainly, the CF Administrator templates (which show you the Administrator interface) need to be able to execute any of these tags, whether they're restricted for all other programmers or not.

The value for this Unsecured Tags Directory is set by default to point to the directory where the Administrator code is stored. The value "CFIDE\Administrator" is the default you'll see indicated in the CF5 Server. In 4.5, the value will by default be set to the same directory but use the full path (in my case, <e:\inetpub\wwwroot\CFIDE\Administrator>). The difference between using a full path and a relative one is the only difference between the two releases. In both releases, you could change it to point to any full path.

Now in More than 5,000 bookstores worldwide

FORFAST DELIVERY

subscribe Now!

The World's leading Independent WebLogic Developer Resource

FOR DEVELOPERS • BY DEVELOPERS

WebLogic

BEA

DEVELOPER'S JOURNAL

Go Online and Subscribe Today!

Helping you enable inter-company collaboration on a global scale

- Product Reviews
- Case Studies
- Tips, Tricks and more!

WebLogic Developers Journal .com

SYS-CON Media, the world's leading publisher of *i*-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of BEA's WebLogic.

*Only \$149 for 1 year (12 issues) regular price \$180.

SPECIAL INTRODUCTORY OFFER HURRY, DON'T DELAY! OFFER EXPIRES APRIL 30, 2002

SAVE \$31*

SYS-CON MEDIA

Indeed, in both releases you could point it to something other than the Administrator directory location, but you shouldn't need to and it's probably not a good idea (unless your Administrator has been placed in some other directory intentionally, perhaps for security reasons, and you need to correctly point to that). If you change it to something other than where the Administrator code is located and you disable the CFREGISTRY tag, you'll find that you won't be able to do anything else in the Administrator anymore. To regain access to the Administrator, you'll have to modify the registry to reset this key (HKEY_LOCAL_MACHINE\SOFTWARE\Allaire\ColdFusion\CurrentVersion\Server\UnsecuredTagsDirectory) to point to the Administrator path and restart CF before being able to get into the Administrator again.

Using the Feature

If you shouldn't change the Unsecured Tags Directory to point anywhere other than the Administrator, how can you leverage this available feature as a developer? If you're not in control of the Administrator, how will any of this knowledge help?

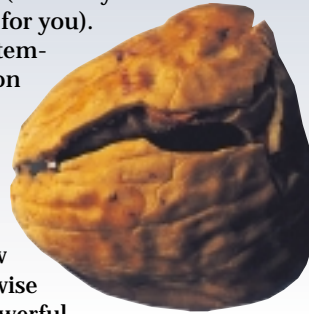
I indicated that any template placed in this "unsecured" directory could execute

any tags, regardless of restrictions. Therefore, to get around the restriction on the tags, simply place the needed code in the "unsecured" directory (or ask your administrator to do this for you). Again, any use of that template from anywhere on the server, whether called directly or by a custom tag call (with CFMODULE, to point to that directory) or a CFINCLUDE, will allow execution of the otherwise restricted tags. That's powerful.

Given the possibilities, it's probably all the more appropriate that the Unsecured Tags Directory value be left to point to the Administrator directory, since that will likely have security placed on it so browser users can't request a directory listing to see any file names that might allow unrestricted access to a tag or tags. It might even be appropriate to restrict access to that directory from the operating system level so that people on the same network can't even see what files are in that directory.

Indeed, the availability of this Unsecured Tag Directory option, and the fact that it's set by default to the Administrator directory, indicates that it's critical that

// There is indeed reason to fear that some might misuse a tag, but one sore point is that it's an all or nothing proposition"



this directory be secured at the operating system level so that no one but authorized users can put files into the directory; otherwise developers would be able to grant themselves unfettered access to the otherwise restricted tags by placing templates there that use them.

(You may wonder if you can specify multiple values for the Unsecured Tag Directory. You can't. At least, it's not documented that you can, and I tried unsuccessfully to use semicolons, commas, and spaces to indicate multiple directory paths.)

SAVE UP TO \$100 ON MULTIPLE SUBSCRIPTIONS



Special online offers

Pick 4 or 5 and Subscribe for one special low price

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTION

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

WWW.SYS-CON.COM/SUBOFFER.CFM



DATA SUBJECT TO CHANGE WITHOUT NOTICE

A Simple Example: Enabling CFFILE Uploads

Admittedly, some servers will be run by tightfisted monarchs who won't be interested in considering the possibilities this enables. Maybe you still don't see the point. Here's a real example.

What if you wanted to get the administrator to allow you to use CFFILE even though it's restricted for the whole server? If some custom tag could be written to allow just the CFFILE Action="Upload", and that was given to the admin for approval and then placed by the admin in that Unsecured Directory, then that specific code could be called as needed by developers.

I offer an example of such a custom tag in Listing 1. Notice that it has to do a bit more than simply run the tag for us. First, it needs to account for the fact that we don't know if the user will provide all the attributes of the CFFILE action="Upload" tag; some of them are optional. Further, for some of the attributes we can't just default them to being an empty string if not provided, so you'll see the use of some CFPARAM tags to set the defaults to whatever the tag documentation says is the default value for that attribute (such as "delete" for NAMECONFLICT).

A couple of attributes (“mode” and “attributes”) are needed only on Unix machines, but again we need to account for the fact that a value may or may not be passed in by setting a suitable default value. The default may not make sense for your installation (though these are ignored on Windows machines anyway), so feel free to change them if needed.

Finally, one other challenge is that the CFFILE tag returns several “file”-prefixed

result variables, such as “file.ClientFile”. Since a custom tag’s variables are local to that tag and the caller will certainly need access to those variables, you’ll see that I also set all the needed result variables to the caller scope, making them available to the caller.

Sadly, I couldn't do something simple like `<CFSET caller.file=file>` or even `CFSET it to duplicate(file)`. The `file` scope isn't considered a structure. Although the release 5 docs say we should use `"cfile"` as the prefix over the older `"file"` prefix (as in `cfile.ClientFile`), the `cfile` scope reacts the same way, so I had to simply assign each expected return value to the caller scoped variable of the same name. (I used `CFSCRIPT` but it could just be 21 `CFSET`s [see Listing 1].)

If that file were saved as `cfile_upload.cfm` and placed in the Unsecured Tags Directory (such as `CFIDE\ Administrator`), then we could call it from anywhere on the server using:

```
<cfmodule
template="/CFIDE/Administrator/cffile
_upload.cfm" filefield="photo" desti-
nation="d:\temp" nameconflict="over-
write" >
```

Notice that it's using CFMODULE's Template attribute to say "run cffile_upload.cfm in the /CFIDE/ Administrator" directory. Of course, this would need to be called in the context of a form action page where the form has passed in an INPUT TYPE="file" field called "photo", and it's saying to store the file on the server in D:\temp, etc. It's a simple replacement for a CFFILE call that would have done the same thing:

```
<CFFILE ACTION="UPLOAD"
FILEFIELD="photo"
DESTINATION="d:\temp"
NAMECONFLICT="OVERWRITE">
```

All that's changed is the CFFILE Action="Upload". Everything else is the same. The power of all this is that the administrator can still prevent anyone from using CFFILE's more dangerous attributes. All they've opened up is the action="Upload".

If the admin still doesn't want to make such an extensible and reusable tag, they could just hard-code the values for the "destination" or even "filefield" attributes to make it useful for a single application only.

Another idea is to limit it to writing only to the directory of the caller by changing the `DESTINATION` attribute of the `CFFILE` tag within the custom tag to be:

```
Destination="#getdirectoryfrompath(CG  
i.CF_TEMPLATE_PATH)#"
```

You'd also then want to remove the CFPARAM tag at the top of the custom tag for attributes.destination, since the attribute would no longer be used. Of course, it may seem to some that such complications would make the whole concept too much bother, but I don't think so. It's still better than not being able to use the tag at all.

Finally, there's still another approach to limiting what the user can write to. You could enable the destination attribute again, but use the CFIMPERSONATE tag and its TYPE="os" attribute so users could

Listing 1: cffile_upload.cfm

```
< CFPARAM name="attributes.filefield">
< CFPARAM name="attributes.destination">
< CFPARAM name="attributes.accept" default="*/*>
< CFPARAM name="attributes.mode" default="0">
< CFPARAM name="attributes.attributes" default="normal">
<CFPARAM name="attributes.nameconflict" default="error">
```

```
<CFFILE action="UPLOAD" destination="#attributes.destination#"
filefield="#attributes.filefield#"
nameconflict="#attributes.nameconflict#" accept="#attributes.accept#"
mode="#attributes.mode#" attributes="#attributes.attributes#">
```

```
<CFSCRIPT>
caller.file.attemptedServerFile = file.attemptedServerFile;
caller.file.clientDirectory = clientDirectory;
caller.file.clientFile = clientFile;
caller.file.clientFileExt = clientFileExt;
caller.file.clientFileName = clientFileName;
caller.file.contentSubType = contentSubType;
```

```

caller.file.contentType = contentType;
caller.file.dateLastAccessed = dateLastAccessed;
caller.file.fileExisted = fileExisted;
caller.file.fileSize = fileSize;
caller.file.fileWasAppended = fileWasAppended;
caller.file.fileWasOverwritten = fileWasOverwritten;
caller.file.fileWasRenamed = fileWasRenamed;
caller.file.fileWasSaved = fileWasSaved;
caller.file.oldFileSize = oldFileSize;
caller.file.serverDirectory = serverDirectory;
caller.file.serverFile = serverFile;
caller.file.serverFileExt = serverFileExt;
caller.file.serverFileName = serverFileName;
caller.file.timeCreated = timeCreated;
caller.file.timeLastModified = timeLastModified;
</CFSCRIPT>

```

CODE
LISTING

The code listing for this article is also located at

write only to what they're authorized for. Even without Advanced Security installed, you could specify your NT domain in the tag's SecurityContext (I found I could leave it blank since my Win2k server doesn't run under the control of an NT domain server) and then pass in a username and password as attributes on the call to the custom tag. You'd wrap the CFFILE tag within this tag, as in:

```
<CFIMPERSONATE username="#attributes.username#"  
password="#attributes.password#"  
securitycontext="" type="OS">  
<CFFILE...>  
</CFIMPERSONATE>
```

Thanks to Raymond Camden for suggesting that last solution.

Summary

The ability to call upon restricted tags (indeed to restrict them in the first place) may be something that's not well understood by many CF developers and administrators. I hope this article provides information and ideas, as well as justification to approach your administrator to propose some alternatives if you work in an environment where some needed tag is restricted.


Administrators, if you've set tags to be restricted but haven't locked down that cfide\administrator directory to keep developers from freely placing files there, you've left the barn door open. I hope, too, that you'll consider using advanced security to restrict who can access what, but that's a subject for a future article (or two).

• • •

Here's one last bonus tip, since this issue of *CFDJ* is about using custom tags. Did you know that CF 5 introduced a means to name more than one location for globally accessible, server-wide custom tags? (This has nothing to do with the Unsecured Tags Directory.) Previously, the default location was cfusion\customtags (i.e., wherever you installed CF under the cfusion directory, such as c:\cfusion). Now you can indicate more than one location using the "custom tag paths" link in the Administrator's "Extensions" section on the main "Server" page (the first one you normally see after logging into the Administrator). While the interface for adding multiple directories is new, you could do it in previous releases by providing a comma-separated list of directories in HKEY_LOCAL_MACHINE\SOFTWARE\Allaire\ColdFusion\CurrentVersion\CustomTags\CFMLTagSearchPath.

Don't forget, this isn't the only way to create alternatives to the default cfusion\customtags directory for placement of globally accessible custom tags. You've always been able to create any number of subdirectories under that directory. The normal call of a custom tag as <cf_yourtagname> will look for yourtagname.cfm first in the current directory where the caller is located, then in cfusion\customtags, then in any of its subdirectories. If you wanted to run a specific tag in a specific subdirectory of cfusion\customtags, you could also use CFMODULE and its NAME attribute. See the CF docs for more information on that tag.

• • •

Some of you may be looking for Part 2 of my "Toward Better CF Administration" article that I started last month (*CFDJ*, Vol. 4, issue 1). Since the focus of this issue is custom tags (and it's got plenty of content already), we've decided to push that off until next month. 

About the Author

Charles Arehart is a certified Macromedia advanced developer/instructor and CTO of SysteManage. He contributes to several CF resources, is a frequent speaker at user groups throughout the country, and provides training, coaching, and consultation services. Charles is also a columnist for *Java Developer's Journal*.

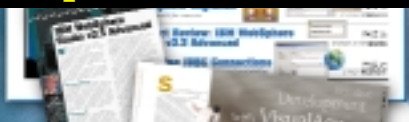
 CAREHART@SYSTEMANAGE.COM

SYS-CON Media, the world's leading publisher of i-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of WebSphere.



WebSphereDevelopersJournal.com

WebSphere
DEVELOPER'S JOURNAL



Introductory Charter Subscription

**SUBSCRIBE NOW AND SAVE \$31.00
OFF THE ANNUAL NEWSSTAND RATE**

ONLY \$149 FOR 1 YEAR (12 ISSUES) REGULAR RATE \$180

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Do You Have Access to the Internet?

The
World's
Leading
Independent
WebSphere
Developer
Resource

Then Subscribe Online and Save \$31! It's that easy

A Fusebox 3 Tutorial – Part 2

BY
HAL
HELMS



Building a 'donations' apparatus

Last month in *CFDJ* (Vol. 4, issue 1) we began building a Fusebox 3 application – a game played at a social club run by Vinny.

The game is simple: registered users log in, pick a number, and then place a bet...

Vinny: Make a donation

Me: Huh?

Vinny: You said, "Place a bet." There's no betting going on. People choose numbers and make donations.

Me: But Vinny, people are picking numbers and then putting money down. Surely, that's a bet.

Vinny: Let me reiterate: the donations people place are not related to the numbers they pick in order to bet...

Me: Ha! You said it: "bet"!

Vinny: ...to better enjoy the game.

Me: I gotta tell you, Vinny. I think somebody's pulling a fast one on you. I know you would never countenance such a thing, but it appears quite clear to me that what we're building here is a betting apparatus, an illegal betting apparatus.

Vinny: That kind of talk is...dangerous. I think you better stick to building the application.

Me: I don't know, Vinny. Well, for now, let's continue. Between last month and this month I wrote all the fuses.

Vinny: Don't these people need to know how to write these fuses?

Me: Well, it's just normal ColdFusion code. The only thing slightly different is that all action returns to the main index.cfm page, or whatever your default page is. Here's some sample code to make this clear:

- **Using a form:**

```
<form action="#self#?fuseaction=
#XFA.home#" method="post">
  <input type="submit" value="Go
  home">
</form>
```

- **Using <cflocation>:**

```
<cflocation url="#self#?fuseac-
tion="#XFA.makeDonation#" addto-
ken="yes">
```

- **Using an href link:**

```
<a
href="#self#?fuseaction="#XFA.
buyProbableWinners#">Look at num-
bers that are "due" to win!</a>
```

Vinny: Yeah, that's clear, clear as mud. What's with this "self" and "XFA." stuff?

Me: Self is a variable that's set to the home circuit's default page.

Vinny: So, why don't you just say that: "index.cfm"?

Me: What happens if your code is deployed on a server that uses "default.cfm"? By using a variable, the code is more portable.

Vinny: What about these XFA things?

Me: XFAs stand for eXit FuseActions. They're also variables.

Vinny: Why do we need those?

Me: Well, two of the design goals of Fusebox 3.0 are to make code more reusable and more maintainable. Imagine that you wrote some code that performs a user login and validation.

Vinny: Yeah, we're gonna need that.

Me: Sure. But lots of applications need the same basic functionality. However, where you head to after validation differs for each application. In some applications, after the user is validated, we want them to go to a catalog. Other applications might send them to a personalized home page, or maybe to an admin page.

Vinny: Okay. But what's that got to do with XFAs?

Me: Well, look at this code that validates a user. I wrote it for another application several months ago. I'll take out the XFAs:

```
<cfif UserInfo.recordCount>
  <cflocation url="#self#?fuse-
  action=catalog.personalized">
<cfelse>
  <cflocation url="#self#?fuse-
  action=login.badLogin">
</cfif>
```

Vinny: If the "UserInfo"...what do you call it?

Me: Recordset?

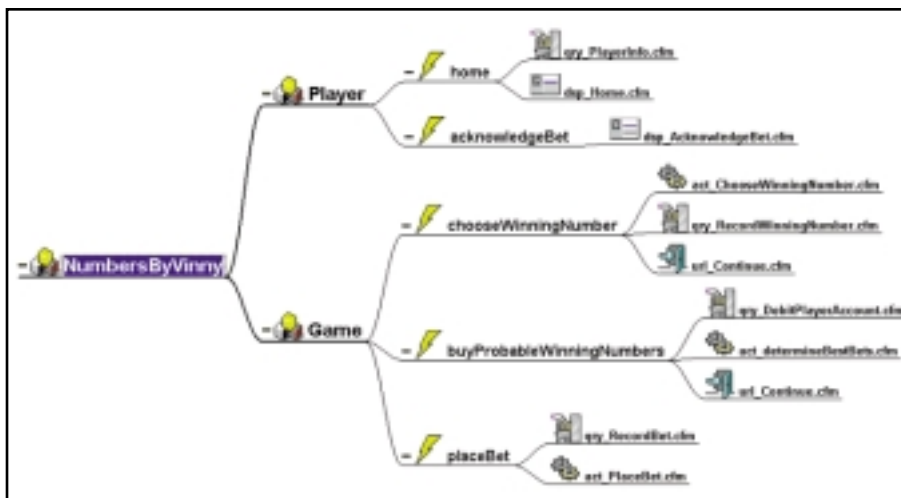


FIGURE 1: Fusebox schematic

Vinny: Yeah, recordset has any rows, then we'll go to the..., help me out here.

Me: To the home circuit's default page.

Vinny: Right. And we send a fuseaction called "catalog.personalized". By the way, what's with the dot stuff in the fuseaction name?

Me: We'll get to that in a minute. I figure I can reuse this code and save you some money. Sound good?

Vinny: Yeah, but...

Me: But what?

Vinny: But, we ain't got a catalog to send people to.

Me: That's just where XFAs come in handy! Take a look at the same code but with XFAs:

```
<cfif UserInfo.recordCount>
<cflocation url="#self#?fuse-
action=#XFA.success#">
<cfelse>
<cflocation url="#self#?fuse-
action=#XFA.failure#">
</cfif>
```

Vinny: But I ain't going to "XFA.success". I'm going to the player's main page.

Me: Right, and the project I wrote that code for was headed off to "catalog.personalized". But by using a variable, we can use the same code for both.

Vinny: How?

Me: By setting the value of "XFA.success" and "XFA.failure" before the fuse is called. That way, it can be specific to each application.

Vinny: Oh...okay. That does kind of make sense. Where would we set this variable?

Me: In a file called "FBX_Switch.cfm".

Vinny: Oh, now you're losing me,

and you never answered my question about the dots in the fuseaction name.

Me: All right. Let's back up. Last month, I used a tool called Visual Mind to create a Fusebox "schematic." It looked like Figure 1.

Vinny: Yeah, I remember.

Me: Now, let's build a Fusebox application around it. I think you'll see how it fits together. The first thing is to create a directory structure that mirrors the schematic, like Figure 2.

Me: There are Fusebox core files – files that should appear in each circuit.

Vinny: What's a circuit again?

Me: We call each directory in Fusebox that handles fuseactions a "circuit."

Vinny: Okay.

Me: You can get the core Fusebox files at www.fusebox.org. They are:

- FBX_Fusebox30_CF50.cfm
- FBX_Fusebox30_CF50_nix.cfm
- FBX_Fusebox30_CF45.cfm
- FBX_Fusebox30_CF45_nix.cfm
- FBX_Fusebox30_CF40.cfm

Vinny: Wait, those all seem like pretty much the same file, no?

Me: Yes, they are. This is the true core Fusebox file and it comes in different flavors, depending on the version of ColdFusion server you're running.

Vinny: But you said that this "self" thing is where everything goes back to, so isn't that really the true core file?

Me: No, all that does is call the core Fusebox file. In fact, "self" can be as simple as this:



```
<cfinclude
template="FBX_Fusebox30_CF50.cfm"
">
```

Vinny: Okay, go on with the other core files.

Me: There's a file called FBX_Circuits.cfm. This lets you define aliases for directory names (or circuits).

Vinny: Why would you want to do that?

Me: Because you may have more than one circuit with the same name. Look at the directory structure in Figure 3.

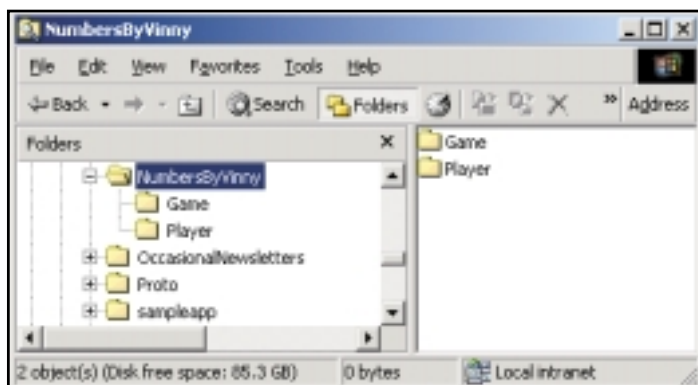


FIGURE 2: Directory structure

PACIFIC ONLINE

www.paconline.net

Here you have two circuits with the same name, "Users." How will the core FBX_Fusebox file know which one you mean?

Vinny: That FBX_Circuits file?

Me: Right. You provide aliases for the directories. Here's the FBX_Circuits.cfm file for the "sampleapp" application:

```
<cfset fusebox.Circuits.home="
    sampleapp">
<cfset fusebox.Circuits.admin =
    "sampleapp/Admin">
<cfset fusebox.Circuits.inventory
    = "sampleapp/Admin/Inventory">
<cfset fusebox.Circuits.users =
    "sampleapp/Admin/Users">
<cfset fusebox.Circuits.main =
    "sampleapp/Users">
```

The FBX_Circuits.cfm file is needed only in the home circuit, by the way.

On the left side of each definition is an entry into a structure called "fusebox.Circuits". That structure is automatically created by the core file.

Vinny: FBX_Fusebox_thingamajig.

Me: Right. The alias to be used is really a key in this circuit. On the right side of the equals sign is the path to the appropriate circuit beginning with the home circuit.

Now when you make a call back to the default page – remember I showed you several examples? – you provide the circuit alias name, followed by a dot separator, and then the name of the fuseaction you want performed, like this:

```
<cflocation url="#self#?fuseac-
tion=users.newUser">
```

Vinny: That answers my question about the dot stuff. But I thought

you said we don't use the fuseaction name itself, that we use an XFA?

Me: That's right. It would really look more like:

```
<cflocation url="#self#?fuseac-
tion=#XFA.addUser#">
```

Vinny: Two questions: Where did you get that name "XFA.addUser"? There's no "addUser" in the schematic you did.

Me: That's easy: I made it up. I make up XFAs to describe as best I can the condition under which an XFA would be called. They don't have to correspond with the actual fuseaction name. Remember, that's just a variable name.

Vinny: Okay, but it looks like a circuit alias – dot – fuseaction thing, and you don't have a circuit called "XFA".

Me: No, "XFA" is a structure that's set in the FBX_Fusebox file. I'm just using a key to that structure.

Vinny: I got it. But where does the value of that key get set?

Me: Good question. For that we have to go to the FBX_Switch.cfm file. First, you need to know a little about what the FBX_Fusebox file is doing.

Vinny: Okay.

Me: When the core file gets a request for a compound fuseaction, something like "Users.newUser" (and for now, just trust that XFA.addUser gets resolved to that), it uses the FBX_Circuits file.

The core file says in effect, "You said 'Users.newUser', but you really mean 'sampleApp/Admin/Users'."

Vinny: This it gets from FBX_Circuits?

Me: Right! With that knowledge it calls the file FBX_Switch.cfm that's found in the target directory.

Vinny: The target directory being "sampleApp/Admin/Users"?

Me: Right again. It also strips off the circuit alias and the dot separator and presents what's left as the variable "fusebox.fuseaction".

Vinny: When it gets to the FBX_Switch file in sample/App/Admin/Users, that file sees a variable called "fusebox.fuseaction"?

Me: Yes. Let's look at that file. Here's a portion of it:

```
<cfswitch expression = "#fuse
    box.fuseaction#">
<cfcase value="newUser">
<cfset XFA.submitForm =
    "Users.addUser">
<cfset XFA.cancelForm =
    "Home.main">
<cfinclude template="dsp_New-
    UserForm.cfm">
</cfcase>
</cfswitch>
```

What's happening here? First, notice that our "<cfcase>" statement is looking for a simple fuseaction. The circuit alias and dot separator have been stripped away. Also, notice that we're setting the values of two keys in the XFA structure, "submitForm" and "cancelForm".

Vinny: Yeah, so that's where they get set.

Me: Yes. You can look at XFAs as "candidate fuseactions." When the fuse returns to self, it does so with only one fuseaction, right?

Vinny: Yeah. That's the <cflocation url="#self#?fuseaction=#XFA.some-thing#">.

Me: Right, and the value of the fuseaction depends on the exit path chosen. If the user clicks the submit button on the form, we'll use the value of "XFA.submitForm" as the fuseaction's value.

Vinny: What would that look like? The code, I mean.

Me: Like this:

```
<form action="#self#"
    method="post">
<input type="hidden"
    name="fuseaction"
    value="#XFA.submitForm#">
...
<input type="submit" value=" OK ">
</form>
```

Vinny: This time you created a hidden field called "fuseaction".

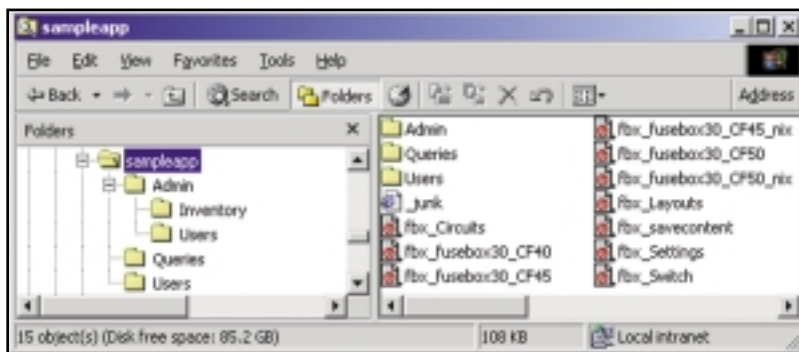


FIGURE 3: Duplicate circuit names

Me: Yes, the core file will sort it out whether a fuseaction is sent as a URL or a form variable. I did this so that if the user clicks on a "Cancel" button, I can use a little JavaScript to reset the value of that form field. Make sense?

Vinny: I think so...

Me: If we wanted to reuse the fuse "dsp_NewUserForm.cfm" in another application, we'd never have to touch that code. We would just set the values of "XFA.submitForm" and "XFA.cancelForm" to whatever would be appropriate for that application.

Vinny: That ain't so hard!

Me: I tell you, Vinny, you should get out of this betting stuff and get an honest job, like programming.

Vinny: Watch what you're saying. I specifically said there wasn't no betting going on.

Me: I know, Vinny, but you have to admit...

Vinny: I ain't admittin' nuthin'! You got any more to say about this Fusebox stuff?

Me: Okay, Vinny...have it your way.

After we set the values of the XFAs, we include whatever fuses are needed to perform the task at hand. Here, we just call "dsp_NewUserForm.cfm".

Vinny: Where would you ever need more than one?

Me: Well, you might need to include a query file prior to calling another display file, that's one example.

Vinny: Yeah, okay.

Me: Now we have to deal with the question of where we set common variables.

Vinny: Like what?

Me: Like "self," for example. Or maybe the data source for a database used throughout the application. We've got another core file for that, FBX_Settings.cfm.

Vinny: Is that just in the home circuit?

Me: No, it goes in every circuit. This enables Fusebox to implement a simple inheritance mechanism, while allowing children to override their parents' settings.

Vinny: It just has "self" and the data source?

Me: No, it can have anything you want. For example, you could put the "<cfapplication>" tag in the home circuit's FBX_Settings.cfm file. If you had common XFAs across



FIGURE 4: Nested layouts

an application, you could put those in FBX_Settings.

Vinny: When do I call them – the FBX_Settings files?

Me: They're called automatically by the core Fusebox file. In fact, the core file uses the directory path for the target circuit...

Vinny: Wait, what's that mean again?

Me: The directory path is the part on the right side of the equal sign in FBX Circuits. This code in FBX_Circuits

```
<cfset  
fusebox.Circuits.users =  
"sampleapp/Admin/Users">
```

is used by the core file to create a list of circuits whose FBX_Settings.cfm files it will include.

Vinny: It will first include the FBX_Settings file in "sampleApp", then in "Admin", and then in "Users"?

Me: Right, and it does it in that order. That lets the coder set variables higher "up" the tree if he or she wants them to be inherited by children circuits. Those variables can always be overridden by one of the children, if needed.

Vinny: Huh, that makes sense.

Me: There's just one other thing you would need to know. Fusebox 3 has the concept of nested layouts.

Vinny: How does that work?

Me: Well, it lets you "wrap" the content generated by your fuses in a layout file.

Figure 4 shows an example of one I did for teaching purposes. See how many layouts you can spot.

I'll give you a couple of hints. Figure 5 provides the circuit structure for the application. I'll also tell you that the fuseaction being called was "grandson.sayHi".

Vinny: Hmm...well, I'm thinking that "Hi there!" is what the fuse in the grandson circuit did. Is that right?

Me: So far, so good.

Vinny: Then I'm thinking that each box is a layout applied to that content. That right?

Me: Right again.

Vinny: There would be four levels of layout applied: grandson, son, father, and home. But we don't have a "home" circuit.

Me: Well, those are the alias names, and if you looked at that application's FBX_Circuit's file, you'd see that I aliased the grandparent circuit as "home".

Vinny: Hey, I like that!

Me: Me, too. Nested layouts are immensely powerful stuff.

Vinny: You just gotta remember to call them in the right order.

Once you're in it...

- Wireless Business & Technology
- Java Developer's Journal
- XML Journal
- ColdFusion Developer's Journal
- PowerBuilder Developer's Journal

reprint it...

Contact Carrie Gebert
201 802-3026
carrieg@sys-con.com

Reprints
SYS-CON MEDIA



FIGURE 5: Circuit structure

Me: No, the core Fusebox file takes care of that for you. It uses that same “fuseaction path”, sample-app/Admin/Users in our example, and reverses it to Users/Admin/sampleApp. Then it walks “up” that tree, including FBX_Layouts.cfm for each circuit.

Vinny: Yeah, that’s cool.

Me: See, Vinny? Face it, you were born to be a programmer. Ditch this gangster stuff and...

Vinny: Yeah, I wanna discuss that very shortly. For now, though, is there anything else I need to know to get my application completed?

Me: Well, the FBX_Layouts.cfm file sets two variables: “fusebox.layout-file” and “fusebox.layoutdir”. Unless you want to have a separate directory for layout files, you can set “fusebox.layoutdir” to an empty string:

```
<cfset fusebox.layoutdir = "">
```

Vinny: What about “fusebox.layout-file”?

Me: That points to a file that, well, take a look at the one I used for that sample application I just showed you:

```
<cfoutput>
<style>
.#Fusebox.thisCircuit# {
border : double Silver;
}
</style>
<table
class="#Fusebox.thisCircuit#">
<tr>
<td>
<font size="-2">Don't mind
```

```
me, I'm just the
#Fusebox.thisCircuit#</font>
</td>
</tr>
<tr>
<td>
#Fusebox.layout#
</td>
</tr>
</table>

</cfoutput>
```

Vinny: Hmmm...you lost me a little.

Me: All right. The first part of the file is just declaring a style; it’s an HTML thing. I’m using a variable name to declare it.

Vinny: Yeah, I see that, “fusebox.-thisCircuit”. Where did that get set?

Me: That’s part of the API (application programming interface) of Fusebox. There are several variables you can access – things like “fusebox.thisCircuit”, “fusebox.rootpath”, “fusebox.targetCircuit” – a bunch of others.

Vinny: Okay, and I can find that info where?

Me: Look in the Fusedoc of the core file and you’ll see the info on it. But don’t worry, Vinny, I’m not going to leave you stranded until we finish your app.

Vinny: Yeah, I was just thinkin’ if anything was to happen...

Me: Well, contingency plans are always a good idea! Okay, let’s continue with the layout file. I create a table that applies the style I just created, then it outputs the variable “fusebox.layout”.

Vinny: What is that?

Me: That’s another part of the API. Essentially, whatever is done in

each circuit is wrapped into a variable called “fusebox.layout”. Then the programmer can decide what to do with it. Here, I’m wrapping the results of what went before in a table.

Vinny: Wait, if “fusebox.layout” wraps “whatever is done in each circuit,” then won’t that code you just wrote be wrapped into this “fusebox.layout” variable?

Me: Exactly! You’re really getting it now. It’s the programmer’s job to do whatever he or she wishes with that variable. Eventually, though, it’s the core file’s job to output “fusebox.-layout”.

Vinny: In that example, what you did was just keep wrapping it in layout files?

Me: Yes, the layout files were identical, with the exception of the actual definition of the style.

Vinny: Well, I think I understand a lot more now.

Me: I think you’re making fantastic progress! Hey, maybe you should come to one of my classes as a student, instead of as the client!

Vinny: Yeah, well, all this talk of yours about how you’re convinced I’m runnin’ an illegal bettin’ operation has made me...reexamine things. Say, I know a nice little place down by the East River where we could talk about this in greater privacy.

Me: That sounds great, Vinny. Let me post this last bit to my editor and, hey, is that your car? Nice car! Wow, look at those guys. Are those your customer service reps?

Vinny: Yeah. Let me introduce you to them...

• • •



Editor’s Note: The last section planned for this article, “Final Thoughts,” had to be omitted since, at this point, we stopped receiving updates from the author. More information may be found at www.halhelms.com.

@ HAL.HELMS@TEAMALLAIRE.COM

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

CTIA WIRELESS 2002

www.ctiashow.com

Mastering ColdFusion 5

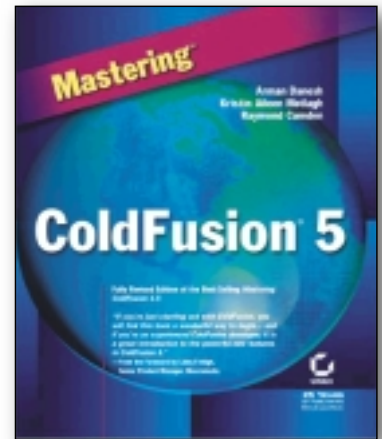
REVIEWED BY
MARK
CYZYK

Mastering ColdFusion 5

By Arman Danesh, Kristin Aileen
Motlagh and Raymond Camden

Sybex

1,263 pages, \$49.95



I received *Mastering ColdFusion 5* two months ago and began using it immediately. Since I had recently upgraded to ColdFusion 5.0, I used this book to learn about some of its new features; I used three of them (query-of-a-query, CFFLUSH, and CFSAVECONTENT) in a single template the first day. This book is an excellent reference to the new features of CF 5.0.

That's not all it's good for. Danesh, Motlagh, and Camden have written a fine book suitable for both novice and expert ColdFusion developers. It's clear that their intention was to produce a comprehensive work that serves not only as an introductory through advanced text but also as a reference to all things ColdFusion. Its whopping 1,263 pages and mammoth size attest to this.

The book is well organized into four parts and five appendices. Part 1 covers the basics, such as "Creating Your First ColdFusion Template," "Passing Data Between ColdFusion

Templates," and "Retrieving Data from a Database." It introduces the beginner to the rudiments of ColdFusion development and offers a solid overview of the architecture of any database-backed CF application.

Part 2 provides what every other programming book offers: an explanation of the meaning and syntax of variables, operators, data structures, flow control, and more. It illustrates a how-to program, specifically, how to program in ColdFusion Markup Language (CFML). This section also explains ColdFusion-specific conventions, techniques, and technologies. For instance, it discusses and illustrates the "ColdFusion application framework," i.e., how to use the built-in capabilities of the ColdFusion server to maintain client state within an application, as well as graphing in ColdFusion, variable locking, form validation, error control, and the myriad other functions that CF provides.

Two chapters within Part 2 are worth mentioning. First, "Using Advanced Query Techniques" provides the best introduction to advanced SQL that I've seen. This chapter focuses on JOIN techniques and syntax, then addresses subqueries and table creation before proceeding to excellent coverage of advanced CFQUERY topics like caching strategies and transaction processing. Second, the chapter "Using ColdFusion Studio" is not something I remember seeing in other books. It provides a brief, useful introduction to the Studio interface and utilities, which should interest beginning CF programmers.

RECEIVE \$150
DISCOUNT OFF FULL CONFERENCE
WEB SERVICES EDGE REGISTRATION

web
services **EDGE**
conference & expo

2002 WORLD TOUR

Learn How to Develop SOAP Web Services NOW!

at a One-Day Tutorial...Coming to a City Near You!

Part 3 focuses on the various technologies built into the ColdFusion platform that enable communication with other popular Internet servers and services via such protocols as SMTP, HTTP, FTP, and LDAP. This section focuses on the CFMAIL tag since e-mail is the most complicated Internet service. Examples of how to send and receive e-mail from within ColdFusion are provided as well as good examples of generating message content from a query, generating a recipient list from a query, and so on.

I eagerly skipped straight to the chapter on CF and LDAP, mainly because I'm having a problem SSL-encrypting communications between my CF server and LDAP server. Unfortunately, this section doesn't cover SSL-encryption in any depth. This is puzzling, since I'd assume the one thing most developers would be using LDAP for would be authentication, and if you're authenticating across a network you'd want to ensure the credentials you're sending are properly encrypted. (The authors do address this topic in slightly more detail in the ColdFusion Tag Reference entry for CFLDAP at the back of the book. At least there it unveils the magic term – cert7.db – that's the key to getting SSL-encryption between CF and LDAP to work at all.)

Part 4 covers such advanced topics as query-of-queries, ColdFusion scripting, using the Verity search engine, CF custom tags, user-defined functions, and WDDX.

“ Their intention was to produce a comprehensive work that serves not only as an introductory through advanced text but also as a reference to all things ColdFusion ”

Being new to CF 5.0, I found the chapters on query-of-queries and user-defined functions particularly enlightening and useful. The sections on the CF Administrator and administering a CF server are welcome additions as well.

Finally, the most important of the six appendices are the ColdFusion Tag Reference and Function Reference; they're what you'd expect: long lists of tags and functions with corresponding explanatory text. I'm a little disappointed with these two sections. The two-column layout is visually confusing – a single-column layout would have been much easier to browse. Also, the function reference is organized alphabetically, not grouped by function type (e.g., string functions, date functions, list functions). There's an index of functions grouped by type at the beginning of this reference, but using it would require a separate lookup – one for the name of the function, the other for the function itself – something the harried developer doesn't want to do

every time he or she needs a quick reference. Grouping the entire function reference by type, then alphabetically, would have been a lot better.

Since *Mastering ColdFusion 5* comprehensively covers almost everything from introductory material to the nitty-gritty of the ColdFusion Markup Language and SQL, advanced application development techniques, and server administration, it could be the one book you use for learning and reference. This book is not something you'd lug around in your backpack (although an 11MB PDF of the entire book is included on the accompanying CD, and is certainly worthy of disk space on a laptop). It is, however, the kind of volume you'd want by your workstation. It's a valuable introduction to new features of ColdFusion, as well as a solid, comprehensive reference to CFML and its built-in functions.



MCYZYK@JHU.EDU

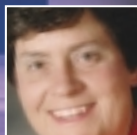
ABOUT THE AUTHOR

Mark Cyzyk is the Web architect at Johns Hopkins University.

Jump-start your Web Services knowledge Get ready for Web Services Edge East and West!

AIMED AT THE JAVA DEVELOPER COMMUNITY AND DESIGNED TO EQUIP ATTENDEES WITH ALL THE TOOLS AND INFORMATION TO BEGIN IMMEDIATELY CREATING, DEPLOYING, AND USING WEB SERVICES.

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



PRESENTERS...

Anne Thomas Manes, Systinet CTO, is a widely recognized industry expert who has published extensively on Web Services and service-based computing. She is a participant on standards development efforts at JCP, W3C, and UDDI, and was recently listed among the Power 100 IT Leaders by Enterprise Systems, which praised her "uncanny ability to apply technology to create new solutions."



Zdenek Svoboda is a Lead Architect for Systinet's WASP Web Services platform and has worked for various companies designing and developing Java and XML-based products.

EXCLUSIVELY SPONSORED BY



BOSTON, MA (Boston Marriott Newton) **JANUARY 29**
WASHINGTON, DC (Tysons Corner Marriott) **FEBRUARY 26**
NEW YORK, NY (Doubletree Guest Suites) **MARCH 19**
SAN FRANCISCO, CA (Marriott San Francisco) **APRIL 22**

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE \$495 REGISTRATION FEE.

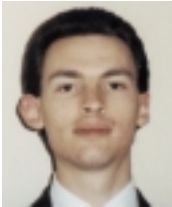
Register at www.sys-con.com or Call 201 802-3069



Using MS-SQL Stored Procedures with ColdFusion

Part 3 of 3

Improve site performance



BY IAN
RUTHERFORD

Now that we've gone over stored-procedure integration with ColdFusion and looked at some programming basics in Parts 1 and 2 of this article (*CFDJ*, Vol. 3, issues 10 and 12), let's look at some other useful and more complicated functions using MS-SQL2000.

First, however, I need to correct a statement I made in Part 1. I mentioned that stored procedures weren't good if you needed to cache a query on the Web server. However, I've since found out from Ben Forta's *CF 4.0 Web Application Construction Kit* that you can cache stored-procedure results and, further, run almost any T-SQL functions within `<CFQUERY>` as long as you're performing only one action. In other words, you can't run a `SELECT` and an `UPDATE` function within one `<CFQUERY>` tag.

Fortunately, when you need to cache a query it's usually a simple `SELECT` statement. Listing 1 shows a simple stored procedure that returns a list of states; Listing 2 shows the CF code to cache this stored procedure. This feature is limited to returning a single result set and can't return values like `<CFPROCParam>` can.

getDate() and dateAdd()

I frequently read about problems with passing dates into the database in the CF support forum. As long as you're simply inserting the date as a

timestamp or need to do calculations on today's date, it's far easier to let the database do the processing through stored procedures than trying to format the date on the Web server into something the database likes. I'll be referring to this date field as a timestamp but it shouldn't be confused with the "timestamp" value type in T-SQL.

If you try to pass a timestamp into the database using the `<CFPROCParam>` tag with the type set to "datetime," you'll probably get a nice error message that says "Feature Not Implemented." Don't worry about the date on the Web server. T-SQL has a feature called "getDate()". It works the same way as "now()" in CF.

Each column in the database can have a default value. The value doesn't have to be a constant; it can be a function. If you use a date as a timestamp for entering records, simply set that "datetime" field to "getDate()" as the default. Then when you do your insert, ignore that column altogether and the database will put in the correct date in the correct format. Listing 3 shows how to do an insert statement using "getDate()" with `<CFQUERY>`, in case you don't trust the database to set the date automatically.

If you need to do addition or subtraction on dates, it works almost the same way as in CF. Even the syntax ("dateAdd(datepart, number, date)") is the same. The dateparts are also the same in T-SQL as in CF5 except for hour ("h"), which is "hh" in T-SQL. To subtract dates, use a negative number in the number field.

Triggers

What if you have a timestamp field that needs to be updated whenever a record is edited? T-SQL has a feature called *triggers* that lets the database take actions when rows are inserted, updated, or deleted. Triggers can be as complex as necessary, but I recommend keeping them simple because tracking errors can be very frustrating otherwise.

Let's look at a trigger for updating the timestamp when a row in a table is edited (see Listing 4).

```
CREATE TRIGGER tr_updateItemTS
ON Items
FOR UPDATE
AS
```

Each trigger must have a unique name. I prefix mine with "tr" for easy identification. The second line tells what type of action the trigger fires on. You can use "Update," "Insert," or "Delete" or any combination separated by commas.

```
DECLARE
    @Item_Pk int

IF Update(ItemMin)
BEGIN
```

I'm declaring a local variable because we'll need to get the primary key of the item to update the row if an update has been done. The "IF" statement isn't necessary if you want to update the timestamp when anything changes in the row. However, if you want to update the timestamp only if a specific column (or columns) has changed, list the column(s) to check in the "IF Update()" section.

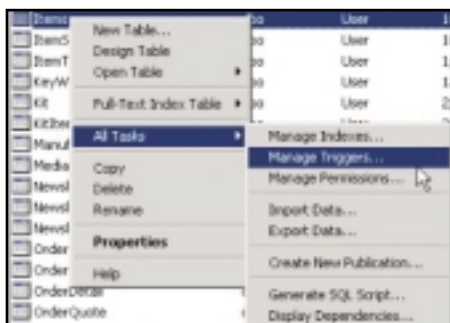


FIGURE 1: Adding a trigger to the database

```
SELECT
    @Item_Pk = Item_Pk
FROM
    Deleted
```

“Deleted” and “Inserted” are two system tables that temporarily contain all the data that was manipulated in the last T-SQL command. For an update, “Deleted” contains all the original data and “Inserted” contains the changed data. Get the primary key out of the “Deleted” table so you know which row’s timestamp to update.

```
UPDATE
    Items
SET
    ItemEditDate = getDate()
WHERE
    Item_Pk = @Item_Pk
END
```

Now update the row in the original table with the new timestamp using the “getDate()” function and stick on the “END” statement if you used an “IF” statement at the beginning.

Triggers can also be used to prevent specific data from being removed from the database. This is very handy when you don’t want someone to inadvertently remove an important row from a table. For example, at our store we have an order table and a gift certificate table in our system. We want the order table to contain the key from the gift certificate table, but we also want to make sure that the key does exist, so we establish a relationship between these tables. However, when a gift certificate isn’t used on an order, we still need to insert some valid value so the relationship doesn’t throw an error. We use the first row in the gift certificate table as the default key to go in the orders table. We don’t want this row to be deleted because it would cause errors in our program. We could use CF to always check this but what if a command isn’t sent through CF? It’s much more reliable to let the database check this itself. See Listing 5 for the code. Notice the “Rollback Transaction” line. This reverses everything that the previous SQL statement tried to do.

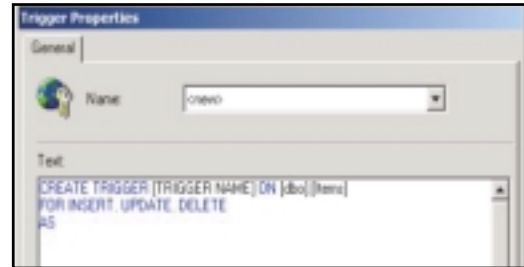


FIGURE 2: Trigger properties

To add a trigger to the database, either take the code in Listing 4 and run it through the Query Analyzer tool or use Enterprise Manager by right-clicking on the table name as illustrated in Figure 1; you’ll get the screen shown in Figure 2.

User-Defined Functions (UDFs)

UDFs made their debut in CF 5 and MS-SQL2000. I’ll demonstrate a simple T-SQL UDF that’s already found in CF but lacking in T-SQL’s built-in functions: “Trim()”.

T-SQL contains “Left()” and “Right()” functions but lacks a “Trim()” function. With UDFs it’s simple to create your own. Here’s the code for the “Trim()” function:

ORDER ONLINE AND GET 10% DISCOUNT GO TO WWW.JDJSTORE.COM TO ORDER

only \$79⁹⁹

3 years' worth of...

- Features & Articles
- Product Reviews
- Case Studies
- Tips & Tricks
- Source Codes
- Interviews
- Editorials & more!

CFDJ the complete works

1999-2001

www.ColdFusionJournal.com

easily searchable HTML FORMAT

THE COMPLETE WORKS

Check out over 250 articles covering topics such as...

- Custom Tags, ColdFusion and Java, Finding a Web Host, Conference Reports, Server Stability, Site Performance, SYS-CON Radio Interviews, ColdFusion Tips and Techniques, Using XML and XSLT with ColdFusion, Fusebox, Building E-Business Apps, Application Frameworks, Error Handling, Wireless ColdFusion, Product Reviews, Ask the Training Staff, Unlocking Verity's Potential, Authentication, Database Tips and Tricks, Monitoring, Smart Objects, A Beginner's Guide to CF, Safe Scripting, JavaScript, Load-Balanced Servers and more...

Questions?
E-mail CFDJCD@SYS-CON.COM

ORDER ONLINE AND GET 10% DISCOUNT GO TO WWW.JDJSTORE.COM TO ORDER

```
CREATE FUNCTION Trim
    (@String varchar(200))
RETURNS varchar (200)
AS
BEGIN
    RETURN LTrim(RTrim(@String))
END
```

The line under the “CREATE...” statement tells what type of data will be given to the function. “RETURNS...” tells what type of data will be returned from the function. The actual processing occurs between the “BEGIN” and “END” statements.

To call a UDF in a T-SQL statement you need to prefix it with “dbo.” This lets the database know that it should look for an object in the database with your function’s name.

```
INSERT INTO
    Names
    (FirstName,
    LastName)
VALUES
    dbo.Trim(@FirstName),
    dbo.Trim(@LastName)
```

Functions can be created through the Query Analyzer or Enterprise Manager (see Figure 3).

Cursors

Sometimes it’s necessary to copy data directly from one table to

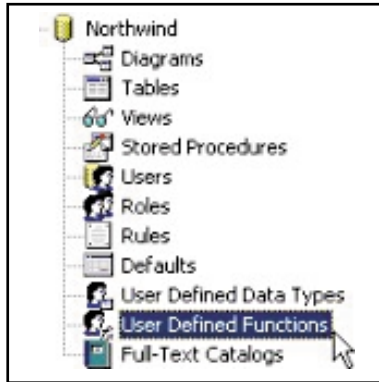


FIGURE 3: Creating functions

another. You could do it using CF as in Listing 6, using <CFQUERY> as in Listing 7, or using the same T-SQL code in Listing 7 in a stored procedure.

What if you need to add more data or manipulate the data before it’s inserted? Cursors are the answer. They allow you to take the result set from one select statement, loop over it, and change data line by line.

Cursor syntax is a little tricky.

```
DECLARE Cart2Order CURSOR
    FAST_FORWARD
    FOR
    SELECT
        CartDetail_Pk,
        ItemNumber,
        Qty,
        UUID
    FROM
        CartDetail
```

The first line of the code gives the cursor a name. The second line tells what type of cursor it is. “FAST_FORWARD” means the cursor can only scroll from the first to the last record and that the data within the cursor can’t be updated. For a full list of cursor options, check the “cursor, types” section of the SQL documentation. “FOR SELECT...” defines the recordset retrieved by the cursor.

```
OPEN Cart2Order
```

To use the cursor you need to open it.

```
DECLARE
    @CartDetail_Pk [int],
    @ItemNumber [int],
    @Qty [int],
    @UUID char(35)
```

Declare all your local variables at the beginning of a stored procedure or as needed. I define local variables that will be used only within a cursor at the beginning of the cursor so the rest of the code is cleaner.

```
FETCH NEXT FROM Cart2Order

INTO
    @CartDetail_Pk,
    @ItemNumber,
    @Qty,
    @UUID
```

Listing 1

```
CREATE PROCEDURE pr_getStates
AS
    SELECT
        StateName
    FROM
        States
    ORDER BY
        StateName
```

Listing 2

```
<CFQUERY NAME="getStates"
    DATASOURCE="#REQUEST.DS#"
    CACHEDWITHIN="#CreateTimeSpan(0,0,60,0)#">
    EXEC pr_getStates
</CFQUERY>
```

Listing 3

```
<CFQUERY NAME="insertUser"
    DATASOURCE="#REQUEST.DS#">
    INSERT INTO
        Users
        (FirstName,
        LastName,
        User_TS)
    VALUES
```

```
'#Variables.FirstName#',
'#Variables.LastName#',
getDate()
</CFQUERY>
```

Listing 4

```
CREATE TRIGGER tr_updateItemTS ON Items
FOR UPDATE
AS
    DECLARE
        @Item_Pk int

    IF Update(ItemMin)
    BEGIN
        SELECT
            @Item_Pk = Item_Pk
        FROM
            Deleted

        UPDATE
            Items
        SET
            ItemEditDate = getDate()
        WHERE
            Item_Pk = @Item_Pk
    END
```



```
WHILE @@fetch_status <> -1
BEGIN
```

When you create a cursor, the “@@Fetch_Status” variable provides information about it. By checking the status after each loop, we can tell when to quit looping. The cursor returns a “-1” value when the cursor is closed. A “FAST_FORWARD” cursor automatically closes itself when it reaches the last row, so this check will stop attempts to loop beyond the result set.

```
SELECT
    @User_Pk = @User_Pk
FROM
    Users
WHERE
    UserID = @UserID
```

This is the first part of the processing. Check if this user, as identified by a unique UUID, exists. If the user does, go ahead and move the item to the order. If not, create the user and then add the item to the order. In real life I'd recommend creating the user before starting work on the cart. I'm doing it this way to demonstrate using "IS NULL" instead of "@@Rowcount". Instead of getting a recordset, set a local variable. This is useful and can

```
IF @User_Pk IS NULL
BEGIN
    (Create the user)

    SET @User_Pk = SCOPE_IDENTITY()
END
```

Rather than seeing if you received any rows using “@@Rowcount”, check if “@User_Pk” has a value. If it does, you already have the data you need. If it doesn’t, skip creating a user. “Scope_Identity()” provides us with the key generated by our insert statement.

```
INSERT INTO
    OrderDetail
    (ItemNumber,
     Qty,
     User_Pk)
VALUES
    @ItemNumber_Pk,
    @Qty,
    @User_Pk
```

If you used “@@Rowcount” instead of setting the local variable “@User_Pk”, you’d have to make a second request from the database to get the value for @User_Pk.

DELETE
CartDetail

```
WHERE
    CartDetail_Pk =
@CartDetail_Pk

FETCH NEXT FROM Cart2Order
INTO
    @CartDetail_Pk,
    @Qty,
    @User_Pk

END
```

After you finish processing each row, get the next row to run the loop again.

DEALLOCATE Cart2Order

When you're finished with the cursor, the line above removes it from memory. Always end your code this way to avoid cluttering your database memory with unnecessary data.

Summary

In this article I've shown how to expand the use of the database to improve site performance, cut down on network traffic, and safeguard data. In the next article I'll show how to debug stored procedures and cursors, and also explain some gotchas when trying to do math with null values.

 IAN@CATHOLICSTORE.COM

```
CREATE TRIGGER tr_PreventGCDelete ON GiftCertificate
FOR DELETE
AS
DECLARE @GC_Pk [int]
BEGIN
SELECT
    @GC_Pk = GC_Pk
FROM
    Deleted
IF @GC_Pk = 1
BEGIN
    ROLLBACK TRANSACTION
END
END
```

```
<CFQUERY NAME="getName"
    DATASOURCE="#REQUEST.DS#">
    SELECT
        FirstName,
        LastName
    FROM
        Name1
</CFQUERY>

<CFLOOP QUERY="getName">
```

```
<CFQUERY NAME="InsertName"
    DATASOURCE="#REQUEST.DS#">
    INSERT INTO
        Name2
        (FirstName,
         LastName)
    VALUES
        ('#getName.FirstName#',
         '#getName.LastName#')
</CFQUERY>
</CFLOOP>
```

```
<CFQUERY NAME="copyNames"
    DATASOURCE="#REQUEST.DS#">
    INSERT INTO
        Names2
    (FirstName,
        LastName)
    SELECT
        FirstName,
        LastName
    FROM
        Names1
</CFQUERY>
```

Ian Rutherford has been working with CF and SQL for two years and is the designer of CatholicStore.com and CatholicLiturgy.com, both built using Fusebox architecture.

DDDDDDDDDDDD

February 2002

What's Online

www.sys-con.com/coldfusion

CFDJ Online

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

Share Your Expertise and Get a Byline!

Readers often ask if **CFDJ** accepts manuscripts from readers. The answer is an emphatic yes! We're continually on the lookout for emerging issues and ideas, and some of the best suggestions and articles come from readers like you. For author guidelines, go to www.sys-con.com/coldfusion/writers.cfm; this site has the complete rundown on how to submit an article proposal, the types of articles **CFDJ** wants, the editorial calendar, a writer's agreement, format details for text and code listings, copyright and payment information, and even some style pointers to improve your manuscript. Your CF expertise and ideas may get you into print!

ColdFusion Forum

Join ColdFusion List, a new ColdFusion community. Join other IT professionals, industry gurus, and ColdFusion writers for ColdFusion discussions, technical questions, and more. Voice your opinions and assessments on topical issues or hear what others have to say. Join the ColdFusion List now to monitor the pulse of the ColdFusion industry.

Search ColdFusion Jobs

ColdFusion Developer's Journal is proud to offer an employment portal for IT professionals. Get direct access to the best companies in the nation. Find out about the "hidden job market" and how you can find it. As an IT professional curious about the market, this is the site you've been looking for. Simply type in the keyword, job title, and location and get instant results. You can search by salary, company, or industry. Need more help? Our experts can assist you with retirement planning, putting together a resumé, immigration issues, and more.

The Latest in CF at the CFBuyersGuide.com

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



<dot.com>

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

<magazine>

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

<content>

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

<conferences>

- web services conference **web 2.0** june 24-27 2002
- web services conference **web 2.0** oct 1-3 2002

<search cfdj>

<cfdj specials>

Collect All Four!

JDJStore.com \$229.99

JDJ, XML, CFDJ, CFAdvisor - ALL Four!!! The Complete Works

<bestsellers>

1. IBM WebSphere Application Server 2.5 Standard Edition. \$999.99
2. CFDJ and CF Advisor - The Complete Works. \$529.99
3. AllWeb Application Platform Release 2.5. \$3,349.00
4. ColdFusion 5.0 Server Enterprise. \$4,999.99
5. ColdFusion 5.0 Server Professional. \$3,299.99

QUICK POLL

- What is your favorite new feature of ColdFusion 5.0?
- ☐ User-Defined Functions
 - ☐ Query of Queries
 - ☐ Application Deployment
 - ☐ Integrated Clustering
 - ☐ Performance Improvements

INTERMEDIAR.NET

NT web hosting

FREE 30-day trial

CFWEB

Click for a FREE 30-day trial

AutoWeb

click here to DOWNLOAD

TEST SUITS

empirix

AbleCommerce

EXTENSION

CFDYNAMICS

QUALITY COLDFUSION HOSTING

mineerva

OUTSIDE COUNCIL

SOURCE

OR

THE ULTIMATE IN WEB TRADING SYSTEMS

Cold Fusion Hosting

virtuallcape

CFXtras

FREE Domain Names

FREE E-commerce software

Cold Fusion Hosting

virtuallcape

WebServices JOURNAL

Special
Introductory
Offer
SAVE \$13.89*

**The world's leading independent
Web Services information resource**

Get Up to Speed with the Fourth Wave in Software Development

- Real-World Web Services: Is It Really XML's Killer App?
- Demystifying ebXML: A Plain-English Introduction
- Authentication, Authorization and Auditing: Securing Web Services
- Wireless: Enable Your WAP Projects for Web Services
- The Web Services Marketplace: An Overview of Tools, Engines and Servers
- Building Wireless Applications with Web Services
- How to Develop and Market Your Web Services
- Integrating XML in a Web Services Environment
- Real-World UDDI
- WSDL: Definitions and Network Endpoints
- Implementing SOAP-Compliant Apps
- Deploying EJBs in a Web Services Environment
- Swing-Compliant Web Services
- and much, much more!

Only \$69.99 for 1 year (12 issues)
Newsstand price \$83.88 for 1 year



THE BEST SOURCE FOR COVERAGE ON .NET

**SYS-CON
MEDIA**

SYS-CON Media, the world's leading publisher of technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of Web Services.

*Offer subject to change without notice

Ask the Training Staff

BY
BRUCE
VAN HORN



A source for your CF-related questions

We're only two months into 2002 and it's already shaping up to be a very busy year for ColdFusion development. I hope you're experiencing the same thing.

If so, don't forget we're here to help you with all your CF-related issues. This month I have two questions for you.

Q: *I'm writing a CF app that needs to interface with another company's shopping cart. A user adds products to our shopping cart and when ready to pay for the products, we send the user to a payment processor for credit-card processing. When the purchase is approved, the payment processor sends the user back to our site using a URL we specify. The problem is this: when the user first comes back to our site, the cart is empty and the session variables seem to be lost. If the user clicks any links within our site again, the cart contents reappear. Any ideas?*

A: I've encountered this situation before with an application I wrote for a client a few months ago. The problem is that the payment processor, after processing the transaction, is redirecting the user back to your site using a server-side redirect. Because of this, the user's browser is not sending the CFID and CFTOKEN cookies needed for session management to your site. However, once the user clicks again within your site, these cookies are available to your server. To solve this you need to include the values of CFTOKEN and CFID as part of the URL the cart will send the user back to.

For example, when your user clicks the "Pay" button on your site, you're sending the amount of the purchase and the return URL (e.g., "yoursite.com/return.cfm") over to the other site. When the user completes the transaction, the other site sends him or her back to that URL on your site. Once back on your site, your server doesn't recognize the

user because CFID and CFTOKEN are missing. You can fix this by changing the return URL to include that person's session identifiers. Modify your code to create a return URL of "return.cfm?CFID=#Session.CFID#&CFTOKEN=#Session.CFTOKEN#" (all within a CFOUTPUT block, of course). This will put that user's session management values in the URL. When the user returns to your site using that URL, CF will look for these identifiers as cookies or as URL variables. The cookies still won't be there, but the correct values will be in the URL so session management can identify them.

Q: *I have a question about your recommendation against copying shared-scope variables into the Request scope (CFDJ, Vol. 3, issue 12). At an Advanced CF course we were told that the duplicate function is very inefficient and resource-intensive, as is using the IIF function. I'm wondering if you've*

done any testing using the StructCopy function instead. I agree with you that copying information into the Request scope with every page load regardless of whether or not the page requires the data is a poor way to do things. However, why not have two templates, SharedToRequest.cfm and RequestToShared.cfm? They'd contain the code needed to handle the task. You could then include them wherever shared-data access is required. This way you avoid needless memory access and are still able to reap the benefits of accessing variable information without locking.

A: I haven't done any testing on the StructCopy function, but I would avoid using it because it copies only the first level of the structure. Therefore, if you have a structure inside a structure, only the first level is copied. The nested structure is still a pointer.

I like your idea of having SharedToRequest.cfm and RequestToShared.cfm cfincluded only where you need them. However, you'd still have some issues with the size of the data being copied and the number of people accessing these pages. The bottom line is: while it does take more time to place explicit cflocks around shared data requests, your applications will run faster than if you copy those pieces of data to the request scope before accessing them.

• • •

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



@BRUCE@NETSITEDYNAMICS.COM



ABOUT THE AUTHOR

Bruce Van Horn is president of Netsite Dynamics, LLC, an Allaire certified instructor, and a member of the CFDJ International Advisory Board.

Subscribe Now

for Instant Access to

CF Advisor

CFAdvisor.com!

12 months
\$89
24 months
\$169

Now on CD!

THE MOST
COMPLETE LIBRARY
OF EXCLUSIVE
CF ADVISOR
ARTICLES!

3
YEARS
40
ISSUES
200
ARTICLES
ONE CD

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

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

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

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



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

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

"The Complete Works"

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

Easy-to-navigate HTML format!

E-Commerce	CFBasics	Object-Oriented CF
Interviews	Reviews	WDDX
Custom Tags	Scalability	Upgrading CF
Fusebox	Enterprise CF	Programming Tips
Editorials	Wireless	CF Tips & Techniques
Databases	Verity	Programming Techniques
News	Source Code CF	Forms
CF & Java	Applications	

Order Online and Save 10% or More!

www.JDJSTORE.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

**SYS-CON
MEDIA**

135 CHESTNUT RIDGE ROAD
MONTVALE, NJ 07645

WWW.SYS-CON.COM

**ONLY
\$71⁹⁹**

Macromedia Announces Free CF Server 5 Developer Edition

(San Francisco) – Macromedia, Inc., has announced the immediate availability of Macromedia ColdFusion Server 5 Developer Edition. This free, nonexpiring, full-featured edition works on a single computer to enable customers to develop applications for ColdFusion Server 5.

CF Server 5 offers a new

integrated charting engine, advanced full text searching, and new language features.


To download the server, visit www.macromedia.com/go/cfdeved.



ActivSoftware / CFDev.com Launch User Group Initiative

(Utica, NY) – ActivSoftware has launched a new initiative for Web development user groups via CFDev.com. The company will sponsor the groups, provide presentations via  download or in person, and offer special licensing arrangements.

The user group receives a free, enterprise-level ActivEdit license for use in the UG Web site; UG managers and active members receive two free commercial-use licenses; and all UG active members receive special license discounts. Additional giveaways and

 special licensing arrangements will be announced in mid FY 2002.

The company encourages user group managers to sign up at www.cfdev.com/usergroups/.



Ektron Releases eWebEditPro 2.5

(Amherst, NH) – Ektron Inc. has released version 2.5 of its eWebEditPro, a browser-based, multilanguage, business user-focused Web content authoring tool.

eWebEditPro version 2.5 offers new bidirectional editing for Arabic and

Hebrew, and adds traditional Chinese to 10 existing European and Asian menus and dialogs. Additional new features include Section 508 compliance (federally mandated), stylesheet enhancements, Netscape 6.2 support, and various new developer commands.

www.ektron.com

Events

<CF_North> Developers Conference

May 4-5, Toronto
www.cfconf.org/CFNorth/

CF_Cruise

May 19-26
In the Caribbean, departs from Tampa, FL
www.cfconf.org/cf_cruise/

CFDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
BEA Eworld	WWW.BEA.COM/EVENTS/EWORLD/2002	404.240.5506	9
CFADVISOR.COM	WWW.CFADVISOR.COM		47
CFDYNAMICS	WWW.CFDYNAMICS.COM	800.422.7957	31
CFXHOSTING	WWW.CFXHOSTING.COM	866.CFX-HOST	23
COLD FUSION DEVELOPER'S JOURNAL	WWW.SYS-CON.COM	800.513.7111	33
EKTRON	WWW.EKTRON.COM/DEMO	603.594.0249	35
EMPIRIX	WWW.EMPIRIX.COM/DOUBLE/CFM		4
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.HOST	13
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800.379.7729	52
JAVA DEVELOPER'S JOURNAL	WWW.SYS-CON.COM	800.513.7111	27
JAVA ONE	HTTP://JAVA.SUN.COM/JAVAONE	888.886.8769	41
MACROMEDIA	WWW.MACROMEDIA.COM/DOWNLOADS	888.939.2545	51
MACROMEDIA	WWW.VUE.COM/MACROMEDIA	877.460.8679	17
MACROMEDIA	WWW.MACROMEDIA.COM/GO/USERGROUPS	877.460.8679	20
PACIFIC ONLINE	WWW.PACONLINE.NET	877.503.9870	43
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	11
QUILL DESIGN	WWW.QUILLDESIGN.COM		2
RACKSPACE	WWW.RACKSPACE.COM	888.850.1470	3
SYS-CON MEDIA, INC.	WWW.SYS-CON.COM	800.513.7111	48
SYS-CON REPRINTS	WWW.SYS-CON.COM	201.802.3026	49
WEB SERVICES EDGE CONFERENCE & EXPO	WWW.SYS-CON.COM	201.802.3069	38, 39
WEB SERVICES JOURNAL	WWW.WSJ2.COM	800.513.7111	25, 45
WEB SPHERE DEVELOPER'S JOURNAL	WWW.SYS-CON.COM	800.513.7111	49
WEBLOGIC DEVELOPERS JOURNAL	WWW.SYS-CON.COM/WEBLOGIC	800.513.7111	42
WIRELESS EDGE CONFERENCE & EXPO	WWW.SYS-CON.COM	201.802.3069	37
XML JOURNAL	WWW.SYS-CON.COM	800.513.7111	15

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

Next Month...

*A Case for Methodologies
Who needs a methodology?
by Hal Helms*

*Microsoft Access Ain't That Bad!
Reduce development time using ODBC
with Access and SQL server
by Adam Howitt*

*Pre-Fab ColdFusion Construction
Build your ColdFusion applications
modularly with prefabricated parts
by Joshua Oster-Morris*

*A Cold Cup O' Joe
Extending Java CFX tags – Part 7
by Guy Rish*

COLD FUSION Developer's Journal

Don't miss the March issue!

MACROMEDIA
www.macromedia.com/downloads

Intermedia.Net
www.intermedia.net